ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ. И. Сәтбаев атындағы Қазақ ұлттық зерттеу техникалық университеті

Б.А. Ахметов, А.Г. Корченко,
В.П. Сиденко, Ю.А. Дрейс, Ж.К. Алимсеитова

ҚОЛДАНБАЛЫ КРИПТОЛОГИЯ:

шифрлау әдістері

Университеттің Ғылыми-әдістемелік кеңесі оқулық ретінде ұсынған

Алматы 2016

ӘОЖ 003.26(075)
ББК 32.973.26-018.1я7
Қ 62

Пікір жазғандар:
Джурунтаев Д.З., тех. ғыл. д-ры, проф., Қ. И. Сәтбаев атындағы ҚазҰТЗУ
Тукеев У.А., тех. ғыл. д-ры, проф., Әль-Фараби атындағы ҚазҰУ
Ускенбаева Р.К., тех. ғыл. д-ры, проф., ХАТУ оқу проректоры
Утепбергенов Е.Т., техн. ғыл. д-ры, профессор, АЭжБУ

Қазақстан Республикасы Білім және ғылым министрлігінің 2016 жылғы басылым жоспары бойынша басылды

Қ 62 Қолданбалы криптология: шифрлау әдістері: Оқулық. – Алматы: Қ. И. Сәтбаев атындағы ҚазҰТЗУ, 2016. – 500 б.
Сурет 175. Кесте 69. Библиогр. – 47 атау.
ISBN 978 – 601 – 228 – 890 – 2

Оқулыққа жалпы мәліметтер кіреді: криптология, криптография және криптографиялық талдау негіздері; дәстүрлі тарихи ауыстыру және алмастыру шифрлары; блокты және құрамдас шифрлар және оларға шабуылдар; ағынды шифрлар және псевдокездейсоқ сандар генераторлары; криптографиялық түрлендіру және шифрлау стандарттары; блокты симметриялық криптографиялық алгоритмдер; ақпараттық-коммуникациялық жүйелерде деректердің конфиденциалдығын камтамасыз ету үшін қолданатын шифрлаудың симметриялық және асимметриялық криптографиялық жүйелерін құрастыру принциптері, сонымен қатар шешімдері бар мысалдар, бақылау тапсырмалары жауаптармен.

Оқулық "Ақпараттық қауіпсіздік" білімдер аймағында оқитын жоғары оқу орындарының бакалаврлар, магистранттар, докоранттарына арналған.

Оқулық Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университетінің және Украинаның Ұлттық авиациялық университетінің авторлық ұжымымен ынтымақтастық туралы Мемарандумға сәйкес дайындалған.

ӘОЖ 003.26(075)
ББК 32.973.26-018.1я7

ISBN 978 – 601 – 228 – 890 – 2

МАЗМҰНЫ

ҚЫСҚАРТУЛАР ТІЗІМІ

AES	-	Advanced Encryption Standard (Жетілдірілген шифрлау стандарты)
ASCII	-	American Standard Code for Information Interchange (Ақпаратпен алмасу үшін кодтардың американдық стандарты)
ATM	-	Asynchronous Transfer Mode (Тасымалдаудың асинхронды әдісі)
BBS	-	Algorithm Blum – Blum – Shub (Блюм-Блюма-Шуба алгоритмі)
CBC	-	Cipher Block Chaining (“шифрланған деректер блоктарын жалғау” режимі)
CFB	-	Cipher Feedback (“шифрланған деректер бойынша кері байланыс” режимі)
CTR	-	Counter Mode (“санағыш” режимі)
DES	-	Data Encryption Standard (Деректерді шифрлау стандарты)
ECB	-	Electronic Code Book (“электронды кодты кітап” режимі)
FIPS	-	Federal Information Processing Standards (Деректерді өңдеудің федералды стандарты)
GSM	-	Global System for Mobile Communications (Мобильды ұялы байланыстың ауқымды цифрлық стандарты)
IBM	-	International Business Machines (Халықаралық бизнес-машиналар корпорациясы)
IDEA	-	International Data Encryption Algorithm (Халықаралық деректерді шифрлау алгоритмі)
IPES	-	Improved Proposed Encryption Standard (Жетілдірілген ұсынылған шифрлау стандарты)
IST	-	Information Societies Technology (Ақпараттық қоғамдық технологиялар)
IV	-	Initialization Vector (Инициализация векторы)
KSA	-	Key-Scheduling Algorithm (кілттік кесте алгоритмі)
LFSR	-	linear feedback shift register (сызықты кері байланысы бар жылжымалы регистр)
MA	-	Multiplication/Addition (көбейту/қосу)

MAC	-	Message Authentication Code (хабардың аутентификация коды)
MDC	-	Modification/Manipulation Detection Code (өзгерістерді/манипуляцияларды табу коды)
MIT	-	Massachusetts Institute of Technology (Массачусетстік технологиялық нститут)
NFS	-	Number Field Sieve (Сандық өрістің елеудің жалпы әдісі – тұтас сандарды факторизациялау әдісі)
NIST	-	National Institute of Standards and Technology (Стандарттар және технологиялардың ұлттық институты)
OFB	-	Output Feedback (“шығыс бойынша кері байланыс” режимі)
PES	-	Proposed Encryption Standard (Ұсынылған шифрлау стандарты)
RSA	-	Rivest, Shamir і Adleman (ашық кілтті криптографиялық алгоритм)
SPA	-	Software Publishers Association (Бағдарламалық қамтаманы құрушылар ассоциациясы)
TCP	-	Transmission Control Protocol (Жіберуді басқару хаттамасы)
ААШ	-	асинхронды ағынды шифрлар
АЖ		автоматтандырылған жүйе
ПКСГ	-	псевдокездейсоқ сандар генераторы
ПКСТГ	-	псевдокездейсоқ сандар тізбегінің генераторы
ДКТ	-	дифференциалды криптографиялық талдау
ЭЕМ	-	электронды есептеу машина
ЭЦҚ	-	Электронды цифрлық қолтаңба
АТЖ	-	ақпараттық-телекоммуникациялық жүйе
КСҚ	-	кілттік сақтау құрылғы
СКТ	-	сызықты криптографиялық талдау
ПКСТ	-	псевдокездейсоқ сандар тізбегі
ПКТ	-	псевдокездейсоқ тізбегі
ӘШҚҚ	-	әуе шабуылына қарсы қорғаныс
САШ	-	синхронды ағынды шифрлар
КСРО	-	Кеңестік социалистік республикаларының одағы
АҰБФА	-	РФ Президент аймағында ақпарат және үкіметтік байланыстың федералдық агенттік

КІРІСПЕ

Бүгінгі күнге ұлттық қауіпсіздіктің саясаттық және экономикалық құрамына әсер ететін алғашқы фактор ақпарат пен ақпараттық ортақтың қорғалғандық деңгейі болып табылады. Сондықтан, әртүрлі қызмет аясында ақпаратты өңдеу кезінде автоматтандырылған (ақпараттық) және телекоммуникациялық жүйелерде ақпаратты қорғауды қамтамасыз ету сұрақтары маңызды мәнге ие болуда.

Жүйеде ақпаратты өңдеу деп келесі операциялардың біреуін немесе бірнешесін орындауды түсінеді: жүйеде техникалық және бағдарламалық құралдар көмегімен орындалатын жинау, енгізу, жазу, түрлендіру, оқу, сақтау, жою, тіркеу, қабылдау, жіберу. Өңделетін ақпарат сияқты, осындай бағдарламалық құралдар (немесе бағдарламалық қамтама) қорғау объектілері болып табылады. Жалпы қорғауды қатынас құруы шектелген ақпарат (сынды ақпарат) талап етеді, нақтырақ: конфиденциалдық, қызметтік немесе құпия ақпарат (мысалы, мемлекеттік құпия, банктік құпия немесе басқа құпия). Қатынас құруы шектелген ақпаратты өңдеу кезінде оны рұқсатсыз және бақылаусыз танысудан, түрлендіруден, жоюдан, көшіруден, таратудан қорғау қамтамасыз етілу керек, яғни оның қорғалғандығының негізгі қасиеттерін қамтамасыз ету – конфиденциалдықты, тұтастықты, қол жетімділікті және байқаушылықты. Соның ішінде қызметтік және құпия ақпаратты бір жүйеден екіншісіне *техникалық* және *ақпаратты криптографиялық қорғау* көмегімен шифрланған түрде немесе қорғалған байланыс арналары арқылы жіберіледі. Криптографиялық қорғау тәсілі бағдарламалық, бағдарламалық-аппараттық және аппараттық құралдар арқылы арнайы (кілттік) деректерді қолданып ақпаратты түрлендіру жолымен ақпараттың мазмұнын жасыру/қалпына келтіру, оның шынайлығын, тұтастығын, авторлығын және т.б. растау мақсатымен жүзеге

асырылады. Өңделетін, сақталатын және/немесе жіберілетін ақпараттың қорғалғандығының керекті деңгейін қамтамасыз ету үшін қолданатын криптографиялық ақпаратты қорғау құралдарының, керекті кілттік, нормативтік, пайдаланушылық және басқа құжаттар (соның ішінде қауіпсіздік шараларын анықтайды) жинағы *криптографиялық жүйе* (криптожүйе) деп аталады. Криптожүйелерді және ақпаратты қорғаудың криптографиялық құралдарын құратын, өндіретін, пайдаланатын және/немесе тарататын ұйымдардың, мекемелердің, кәсіпорынның ақпаратын криптографиялық қорғауын қамтамасыз етуге жұмысы бағытталған мүшелер, бөлімшелер, топтардың жиыны – *ақпаратты қорғаудың криптографиялық жүйесі* деп аталады.

Ақпараттық-телекоммуникациялық жүйеде (АТЖ) деректердің конфиденциалдығын қамтамасыз ету үшін заманауи симметриялық және асимметриялық алгоритмдерін қолдану негізінде ақпаратты қорғаудың криптографиялық жүйелерін құрудың жалпы принциптерін қалыптастыру *осы оқу құралының мақсаты* болып табылады, ол келесі тараулардан тұрады:

бірінші тарау криптография, криптология және криптографиялық талдау туралы жалпы мәліметтерді мазмұндайды. Бұл жерде *АТЖ* ақпаратты қорғау негіздері, ақпараттық қауіпсіздікті қамтамасыз етудің жалпы мақсатында криптографиялық хаттамалардың рөлі, *К. Шеннонның* құпияланған байланыс теория негіздері ашылған, хабарларды шифрлау әдістерінің жіктеліу келтірілген;

екінші тарау дәстүрлі тарихи ауыстыру (біралфавитты, көпалфавитты) және алмастыру шифрларына арналған;

үшінші тарау блокты шифрларға криптографиялық талдаумен блокты және құрамдас шифрларды қолданатын заманауи симметриялық криптографиялық жүйелерді құру принциптерін анықтайды;

төртінші тарауда ағынды шифрлар (*A5, RC4*) және *BBS* алгоритмінде, кері байланысы бар жылжымалы регистрлерге негізделген псевдокездейсоқ сандар генераторлары ашылған;

бесінші тарау Data Encription Standard (*DES*) деректерді блокты симметриялық шифрлау стандартының мәнін ашады: құру тарихы, құрастыру принциптері, құрылымы, шифрлау мысалдары және мүмкінді шабуылдардың криптографиялық талдауы;

алтыншы тарау деректерді блокты симметриялық шифрлаудың криптографиялық алгоритмдерін орындау режимдерін мазмұндайды;

жетінші тарауда ГОСТ 28147-89 криптографиялық түрлендіру стандарты қарапайым ауыстыру режимінде, гаммалау режимінде, кері байланысы бар гаммалау режимінде және имитоендірме қалыптастыру режимінде толық қарастырылады;

сегізінші тарау Rijndael және *Advanced Encryption Standard* (*AES*) блокты симметриялық криптографиялық алгоримдеріне арналған: стандарт құрылымы, түрлендіру раунды және т.б.;

тоғызыншы тарау International Data Encryption Algorithm (*IDEA*) және *Camelia* деректерді шифрлаудың халықаралық стандарттарының мәнін ашады: құру тарихы, құрастыру принциптері, құрылымы, шифрлау мысалдары және мүмкінді шабуылдрдың криптографиялық талдауы;

онынші тарауда асимметриялық криптографиялық жүйелер қарастырылған: *Диффи-Хеллман, Шамир, Эль-Гамаль, RSA, Рабин*; дискреттік логарифмдеуге негізделген криптожүйелерді бұзу әдістері.

Әр тарауға *бақылау сұрақтары* және игерілген білімдерді өзіндік бақылау үшін *тапсырмалар*, сонымен қоса бақылау тапсырмаларға оларды орындаудың дұрыстығын тексеру үшін жауаптар қосылған.

Оқулықтың құрылымы мен мазмұны «Ақпаратты қорғаудың математикалық негіздері», «Ақпаратты қорғаудың теориялық негіздері», «Криптография және шифрлау математикасы» пәндерінің мазмұндарымен, авторларының теориялық және тәжірибелік тәжірибесімен анықталған.

Оқулық «Ақпараттық қауіпсіздік» білім бағытында оқу үрдісіне, бірінші болып дайындаудың 5В100200, 6М100200, 6DB100200 - «Ақпараттық қауіпсіздік жүйелері» бағыты үшін арналған. Осыған қоса, криптографиялық жүйелерді, *АТЖ* ақпаратты қорғау құралдары мен кешендерді қолданумен байланысты мамандарға, ғылыми зерттеулер жүргізу кезінде магистранттар, докторанттар және ғалымдарға қолдануға арналған.

1 тарау

КРИПТОЛОГИЯ, КРИПТОГРАФИЯ ЖӘНЕ КРИПТОГРАФИЯЛЫҚ ТАЛДАУ НЕГІЗДЕРІ

1.1. АҚПАРАТТЫҚ-ТЕЛЕКОММУНИКАЦИЯЛЫҚ ЖҮЙЕЛЕРДЕ АҚПАРАТТЫ ҚОРҒАУ ТУРАЛЫ ЖАЛПЫ МӘЛІМЕТТЕР

Есептеу техника құралдарының және деректерді жіберудің ашық желілерінің тез дамуы күнделікті өмірде және кәсіпкерлік қызметте олардың кең тарауына әкелді. Құатты есептеу мүмкіндіктері және ақпаратты жіберудің жеделдігі көптеген дәстүрлі салаларда қалыптасқан бизнес жүргізу принциптеріне әсер етумен бірге кәсіпкерлік қызметтің дамуының жаңа бағыттарын ашты [21].

Бірақ компьютерлік технологиялар аймағында адам ойының соңғы табыстары дербес компьютерлер, деректерді жіберу желілері және электронды ақша пайда болуымен ғана байланысты емес, сонымен қатар қасқой, ақпараттық қару, компьютерлік вирустар және т.б. түсініктермен байланысты.

Тәжірибеде АТЖ қауіптері ұқсас жүйелердің соңғы пайдаланушылардың қызығушылықтарын тұдыратын ақпаратқа, АТЖ ақпараттық ресурстарына және телекоммуникациялық қызметтеріне тікелей әсер етумен жүзеге асырылу мүмкін. Мысалы, *Internet* арқылы кең тараған шабуыл түрі бар - *TCP* (*Transmission Control Protocol – жіберуді басқару хаттамасы*) жалған ақпараттар дауылы – жүйе қашықтағы пайдаланушыларға қызмет көрсетуді уақытша тоқтатуға әкелетін байланыс.

Ақпараттық қауіпсіздік деп АТЖ сақталатын, жіберілетін, өңделетін деректерді заңсыз танысудан, түрлендіруден және жоюдан (түрлендірудің шеткі жағдайы сияқты) қорғалғандық күйін,

сонымен қоса олардың жұмыс қабілеттігін бұзуға бағытталған әсерлерден ақпараттық ресурстардың қорғалған күйін түсінеміз.

Пайдаланушыға арналған ақпаратты қорғаудың негізгі мақсаттары келесілерді қамтамасыз ету [21]:

- ақпараттың конфиденциалдығын;
- ақпараттың тұтастығын;
- ақпараттың шынайлығын;
- ақпаратқа қатынас құрудың жеделдігін;
- электронды құжат түрінде ұсынылған ақпараттың заңдық мәндігін;
- клиент әрекеттерінің қадағаламаушылығын.

Ақпараттың конфиденциалдығы - ақпарат тарайтын АТЖ пайдаланушыларының шектелген шеңберіне қол жетімді болу қасиеті.

Ақпараттың тұтастығы деп ақпараттың немесе бағдарламалық қамтаманың жіберу және/немесе сақтау процессінде өзінің құрылымын және/немесе мазмұнын сақтау қасиетін түсінеді. Желі арқылы ақпаратты хабар түрінде жіберу сұрақтарын қарастырып, әр хабар өзінің мәндік мазмұны бойынша кейбір класты құрайтыны туралы қорытынға келуге болады. Басқа сөзбен айтқанда, егер ақпаратты электронды түрде көрсеткен кезде пішімі өзгерсе де соңғы хабар мәні алғашқыдай болады. Сонымен, әр хабарда кейбір тілде өзінің эквиваленттік класы болады, және берілген жағдайға *тұтастықты сақтау* қасиетін келесі түрде қалыптастыруға болады: *M* жіберілген хабар өзінің тұтастығын сақтады деп есептеледі, егер жіберу нәтижесінде алынған хабар *M'* хабардың *M* эквиваленттік класына кірсе.

Ақпараттың шынайлығы – оның көзі болған объектке немесе ол ақпаратты жіберген объектке қатаң тиесілі болуды білдіретін қасиет.

Жеделдік – оның уақыт талаптарына сәйкес соңғы пайдаланушыға ақпараттың немесе кейбір ақпараттық ресурстың қол жетімді болу қабілеттігі.

Заңдық мәндік құжатта заң күші бар екендігін білдіреді. Бұл мақсатпен жіберетін хабардың заңдық мәнін растауды мұқтаж ететін объекттер ақпараттың заңды мәнді болатын қабілеттігін көрсететін кейбір атрибуттарын барлық жерде қабылдау туралы келіседі. Ақпараттың бұл қасиеті ақшалық құралдарды аудару операцияларын орындайтын электронды төлемдер жүйелерінде

ерекше маңызды. Айтылғанға қарап, ақпараттың, оның заңды мәнділігін растайтын қасиетін білдіретін, атрибуттарына қойылатын кейбір талаптарын қалыптастыруға болады. Ақпаратты, формалды көз қарастан тек оған тиесілі жіберуші ғана құрғаны анық анықталатындай қалыптастыру керек.

Қадағаламаушылық – басқа объекттер байқамайтындай АТЖ кейбір әрекеттерді орындау қабілеттігі. Бұл талаптың өзектілігі *электронды ақша* және *Internet-banking* түсініктері пайда болу арқасында сірә болды. Электронды төлем жүйесіне қатынас құруды авторландыру үшін пайдаланушы оны бірмәнді идентификациялайтын кейбір мәліметтер беру керек. Осы жүйелердің тез дамуына байланысты нақты қауіп пайда болу мүмкін. Мысалы, барлық төлем операциялары бақыланады, сонда АТЖ пайдаланушылар артынан жаппас бақылау шарттары туындау мүмкін.

Қадағаламаушылық проблемасын шешудің бірнеше жолдары бар:

- АТЖ пайдаланушылар артынан кез келген жаппас бақылауды заңберу актылар көмегімен тыйым салу;
- қадағаламаушылықты қолдау үшін криптографиялық әдістерді қолдану.

Айтылып кеткендей, ақпараттық қауіпсіздік кейбір конфиденциалды мәліметтер үшін ғана емес, сонымен қатар АТЖ берілген функцияларды орындау қабілеттігі үшін де қарастырылу мүмкін.

АТЖ жұмыс қабілеттігіне қатысты ақпараттық қауіпсізідік шеңберінде шешілетін негізгі есептер келесілерден қорғауды қамтамасыз ету керек:

- ақпараттық арналарға, дабылдама арналарына, коммутациялық жабдықтарының дерекқорын жүктеу және басқаруға, жүйелік және қолданбалы бағдарламалық қамтамаға әсер етумен білдірілетін телекоммуникациялық жүйенің жұмысын бұзудан;
- деректердің ағып кетуіне, ақпараттың және желінің тұтастығын бұзуға, ақпаратты тарату ішкі жүйесінің жұмысына, дерекқордың қол жетімдігіне әкелетін желі ресурстарын қолдану талпыныстарынан және ақпараттық ресурстарға рұқсатсыз қатынас құрудан;
- ендірілген және сыртқы қорғау құралдарын бұзудан;

- желі пайдаланушылардың және қызмет көрсетуші қызметкерлерінің құқықсыз әрекеттерінен.

Ақпараттық қауіпсіздіктің көрсетілген есептер арасында басымдылықтары әр нақты АТЖ үшін жеке анықталады және ақпараттық жүйелерге тікелей қойылатын талаптарынан тәуелді болады.

Мемлекеттік құрылымдар көз қарасынан, қорғау бойынша шаралар біріншіден ақпараттың конфиденциалдығын, тұтастығын және қол жетімдігін қамтамасыз етуге шақырылғанын есепке алу керек. Режімдік мемлекеттік ұйымдар үшін мәліметтердің конфиденциалдығы бірінші орнында тұрғандығы, ал тұтастық олардың өзгермеушілігі деп түсінетіндігі түсінікті. Коммерциялық құрылымдарға деректердің тұтастығы және оларды өңдеу бойынша деректер мен қызметтердің маңызды екені ықтимал. Мемлекеттік ұйымдарға қарағанда коммерциялық ұйымдар ашық және серпінді, сондықтан мүмкінді қауіптер сан бойынша және сапа бойынша әртүрлі болады.

АТЖ ақпараттық қауіпсіздік есебін шешу үшін керек:

- ақпаратты оны сақтау, өңдеу және желі арқылы жіберу кезінде қорғау;
- деректер объектерінің және пайдаланушылардың (байланыс орнататын жақтарды аутентификациялау) шынайлығын растау;
- деректер объектерінің тұтастығын бұзудың алдын алу және табу;
- конфиденциалдық ақпаратты ағудын және ақпаратты алудың ендірілген электронды құрылғылардан құрғау;
- бағдарламалық түліктерді бағдарламалық қыстырмаларды және вирустарды ендіруден қорғау;
- желінің ақпараттық ресурстарына және техникалық құралдарына, соның ішінде ақпараттың және жалпы желінің қорғалғандық деңгейін төмендетпеудің алдын алу үшін басқару құралдарына рұқсатсыз қатынас құрудан қорғау;
- конфиденциалды деректердің сақтығын қамтамасыз етуге бағытталған талап етілетін шараларды ұйымдастыру;
- бөлмелерді және техникалық құрылғыларды қорғау.

Ақпаратты қауіпсіздікті қамтамасыз етудің жалпы принциптерін нақты жүзеге асыру ақпаратты қорғаудың техникалық немесе ұйымдастыру шараларында болу мүмкін.

Өңделетін және жіберілетін деректерді қорғау шараларының көлемі ең алдымен болжамды залал мөлшерінен тәуелді екенін айтып кету қажет. Бұл мөлшер тұра (мысалы, тұтастығы бұзылған кезде жаңа бағдарламалық қамтаманы сатып алуға шығындар) немесе жанама (мысалы, банктің ақпараттық жүйесі тоқтап тұру кезіндегі шығындар) түрде болу мүмкін. Кейбір жағдайларда залал мөлшерін есептеу қыйын (мысалы, мемлекеттік құпияның сыртқа кету кезінде).

1.2. АҚПАРАТТЫҚ ҚАУІПСІЗДІКТІ ҚАМТАМАСЫЗ ЕТУ ЖАЛПЫ МАҚСАТЫНДА КРИПТОГРАФИЯЛЫҚ ХАТТАМАЛАРДЫҢ РӨЛІ

АТЖ ақпараттық қауіпсіздікті қамтамасыз ету негізін *криптографиялық әдістер* және *ақпаратты қорғау құралдары* құрайды. Ең сенімді қорғауды тек кешенді тәсіл көмегімен қамтамасыз ету мүмкін екенін есепке алу қажет, ол деген есепті шешу ұйымдастыру-техникалық және криптографиялық шаралардың жиыны болу керек.

Криптографиялық әдістер негізінде анықталған математикалық заңдар бойынша жасалатын *ақпаратты криптографиялық түрлендіру* түсінігі жатыр. Бұл түрлендіру бөтен пайдаланушылардың берілген ақпаратқа қатынас құруын болдырмау мақсатымен, сонымен қатар солардың ақпаратты бақылаусыз өзгертуін болдырмауын қамтамасыз ету мақсатымен орындалады.

Қорғаудың криптографиялық әдістерін қолдану ақпараттық қауіпсіздіктің негізгі есептерін шешуін қамтамасыз етеді. Бұны конфиденциалдық және қызметтік ақпаратты, соған қоса жалпы ақпараттық ресурстарды қорғаудың келесі криптографиялық әдістерін жүзеге асыру жолымен жасауға болады:

- деректерді жіберудің ашық желілері арқылы жіберілетін барлық ақпараттық трафикты және бөлек хабарларды шифрлау;
- байланыс орнататын әртүрлі деңгейлі объекттердің (ашық жүйелердің өзарақатынас үлгілерінің деңгейлері) криптографиялық аутентификациясы;
- жіберілетін ақпараттың шынайлығын және тұтастығын қамтамасыз ету мақсатымен *имитоқорғау* (жалған ақпараттан

қорғау) және *электрондық-цифрлық қолтаңба (ЭЦҚ) құралдарымен* деректерді тасымалдайтын трафикты қорғау;

- дерекқорда сақталанын немесе файл түрде берілген деректерді шифрлау;

- криптографиялық берік бақылау сомалырын қолдану жолымен бағдарламалық қамтаманың тұтастығын бақылау;

- төлем құжаттарының заңды мәндігін қамтамасыз ету үшін ЭЦҚ қолдану;

- электронды ақша түсінігінде негізделген төлем жүйелерінде клиент әрекеттерін бақыланбауын қамтамасыз ету үшін *қараңғылатын цифрлық қолтаңбаны* қолдану.

Келтірілген криптографиялық қорғау әдістерінің көбін жүзеге асыру кезінде кейбір ақпаратпен алмасу керектігі туындайды. Мысалы, АТЖ объектерін аутентификациялау идентифициялайтын және аутентификациялайтын ақпаратпен алмасуды талап етеді.

Жалпы жағдайда осындай жүйелер объекттерінің (субъектерінің) өзара қатынасы үнемі *хаттама* деп аталатын кейбір келісімдерді сақтаумен өткізіледі. Формалды хаттама деп анықталған мақсатқа жету үшін объектердің (субъектердің) әрекеттер тізімін есептейміз. Бұл жағдайда ол хаттаманы қолдану құрылымы мен спецификасын анықтайды.

Өз өшіретімен, *криптографиялық хаттама* деп оның қатысушылары анықталған мақсатқа жету үшін ақпаратты криптографиялық түрлендірулерді қолданатындарды атаймыз.

Криптографиялық хаттамалар көмегімен шешілетін ақпараттық қауіпсіздікті қамтамасыз етудің негізгі есептері:

- арықарай қорғалған деректерді алмасуды орнатумен кілттік ақпарапен алмасу; бұл жерде кілтпен алмасатындарда алдын ала өзара қатынас құру туралы ешқандай мәліметтер жоқ (мысалы, криптографиялық хаттамаларды қолданусыз деректерді жіберудің таратылған желілерінде кілттік ақпаратты тарату жүйелерін құру мүмкін емес);

- байланыс құратын жақтарды аутентификациялау;

- ақпараттық және телекоммуникациялық қызметтерге қатынас құру кезінде пайдаланушыларды авторландыру.

Бүгінгі күнге *Internet* сияқты және олардың негізінде құрылған *intranet* және *extranet* деректерді жіберудің ашық желілердің барлық жерде қолдану арқасында криптографиялық хаттамалар әртүрлі есептер шеңберін шешуде және сондай желілер

пайдаланушыларына берілетін үнемі кеңейетін қызметтерді қамтамасыз етуде кең қолданауын тапты.

Жоғарыда көрсетілген хаттамаларды классикалық қолдану аймақтарынан басқа, сәйкес криптографиялық хаттамалар көмегімен шешілетін арнайы есептерінің кең шеңбері бар. Бұл, ең алдымен құпияның өзін толық көлемде ашусыз мәліметтердің бөлігін ашу, сонымен қатар құпияны бөлшектеп ашу. Мысалы, қатысушылар жалпы мақсатқа жету үшін бір біріне өз ақпараттың бөлігін ашады немесе әрқайсысына белгісіз құпияны ашу үшін ықпалдарын біріктіреді.

Криптографиялық хаттамалардың тез дамуы электронды төлемдер жүйелерінің, интеллектуалды карталарының дамуымен, электронды ақша пайда болумен және т.б. байланысты.

Бүгінгі күнге *Internet* ақпаратты қорғаудың негізгі криптографиялық құралы хаттамалар болғандықтан, осындай коммерциялық құпияларды қорғау құралдарының сапа және сан жағынан дамытынын айтуға болады.

Жергілікті және таратылған ақпараттық жүйелерде ақпараттық қауіпсіздікті қамтамасыз ету есептерін шешуде криптографиялық хаттамаларды қолдану көпқырлығы олардың негізгі типтерін, оларды тәжірибелік қорғау сұрақтарын және олардың негізінде арнайы ақпараттық жүйелерді құруды толық қарастыруды талап етеді.

1.3. КЛАССИКАЛЫҚ КРИПТОЛОГИЯ, КРИПТОГРАФИЯ ЖӘНЕ КРИПТОГРАФИЯЛЫҚ ТАЛДАУ ТУРАЛЫ ЖАЛПЫ МӘЛІМЕТТЕР

1.3.1. Криптология туралы жалпы мәліметтер

Криптология – құпия жазбаны (*криптографияны*) және оны ашу әдістерін (*криптологиялық талдауды*) зерттейтін білім аймағы, *RSA* әйгілі криптографиялық жүйенің авторларының бірі *Рональд Ривесттің* (Массачусетстің технологиялық институтының профессоры – *MIT*) айтуы бойынша криптология «барлық *computer science* кіндік шешесі» [2, 12]. Криптологияның аты грек тілінен (*криптос* – құпия, *логос* – ғылым немесе сөз) пайда болды.

Заманауи криптологияны ғылым ретінде құру математиканың, физиканың, ақпарат теориясының және табиғы қыйын күрделі

есептеулердің фундаменталды түсініктері мен фактілер жинағында негізделген. Бірақ, оған тән қыйындыққа қарамастан криптологияның көптеген теориялық жетістіктері біздің ақпараттық технологиялармен нығайтылған өмірде кең қолданысқа ие, мысалы: пластикалық *smart*-карталарда, электрондық пошталарда, банктік төлем жүйелерінде, *Internet* арқылы электронды саудада, электронды құжатайналым жүйелерінде, дерекқорларды жүргізу кезінде, электронды дауыс беру жүйелерінде және т.б. Осындай жалпы ішкі қыйындық пен тәжірибелік қолданушылық қатынасы теориялық ғылым үшін бірегей.

Криптография - ақпараттың мазмұнын жасыру, өзгертуді немесе рұқсатсыз қолдануды болдырмау мақсатымен ақпаратты түрлендірудің (шифрлаудың) бағдарламалық және аппараттық құралдарын, әдістерді, алгоритмдерді, үлгілерді зерттейтін қолданбалы математика тарауы. Басқа сөзбен айтқанда, криптограф хабарлардың құпиялығын және/немесе шынайлығын қамтамасыз ету әдістерін іздейді.

Ашық электронды байланыс арналар бойынша жіберілетін деректерді қорғау кезінде, электрондық ақпараттың тұтастығын растау кезінде немесе оның авторлығын дәлелдеу кезінде криптографиясыз болмайды.

Криптографиялық талдау кілтті білусіз ақпараттың конфиденциалдығын және шынайлығын бұзудың математикалық әдістерін біріктіреді.

Криптологияға кірмейтін, бірқатар ұқсас білім салалары бар. Ақпараттық массивтерде ақпаратты жасыруды қамтамасыз етумен *стеганография* айналысады. Кездейсоқ әсер ету кезінде ақпараттың тұтастығын қамтамасыз ету *кедергіге төземді кодтау теориясына* жатады. Сонымен қатар, криптологияға жанама аймақ болып ақпаратты қысудың математикалық әдістері есептеледі.

Бұл оқулықта криптографиялық хаттамаларды қолданатын жүйелердің беріктілігін анықтау үшін криптографиялық талдау бөлігі және криптография негіздері ғана қарастырылады.

1.3.2. Криптологияның даму тарихы

Криптография тарихында шартты түрде төрт кезеңді белгілеуге болады [6]: аңғырт, формалды, ғылыми, компьютерлік.

Аңғырт криптология (XVI ғ. басына дейін) үшін хабар мазмұнын жасыру үшін қарсыласты шатастырудың кез келген қарапайым тәсілдерін қолдану сипат. Бастапқы кезеңде ақпаратты қорғау үшін криптологияға тең емес, бірақ қатар *кодтау* және *стеганография* әдістері қолданылды.

Қолданатын көптеген шифрлар *алмастыруға* немесе *біралфавитты ауыстыруға* алып келетін. Тіркелген мысалдардың алғашқылардың бірі *Цезарь шифры,* алғашқы хабардың әр әріпі ол әріптен алфавитте анықталған позициялар санына жылжыған әріпке ауыстырылады. Басқа шифр, *полибиандық төртбұрыш,* оның авторлығы грек жазушы *Полибийге* тіркеледі, біралфавитты ауыстыру болып есептеледі. Бұл шифрда шифрлау алфавитпен кездейсоқ толтырылған төртбұрыш кесте (грек алфавиты үшін 5×5) көмегімен орындалады. Алғашқы хабардың әр әріпі кестеде астында тұратын әріпке ауыстырылады.

Формалды криптология кезеңі (XV ғ. соңы - XX ғ. басы) формалданған және қолымен жасалатын криптографиялық талдауына берік шифрлардың пайда болуымен байланысты. Бұл еуропалық мемлекеттерде ғылым және сауда дамуы ақпаратты қорғаудың сенімді тәсілдеріне сұраныс тұдырған кезде Жаңару дәуірінде болды. Бұл кезеңде маңызды рөл бірінші қатарда *біралфавитты ауыстыруды* ұсынған италиялық сәулетші *Леон Батист Альбертиге* тиесіл. XVI ғ. *Блез Вижинер* дипломаттың атына ие болған бұл шифр алғашқы хабардың әріптерін кілтпен (процедураны арнайы кесте көмегімен жеңілдетуге болады) рет бойынша *«қосудан»* тұрады. Оның *«Шифр туралы трактат»* (1466 ж.) жұмысы криптология бойынша бірінші ғылыми жұмыс деп есептеледі.

Баспадан шыққан бірінші жұмыстардың бірі неміс аббат *Иоганн Тритемиустың* «Полиграфия» (1508 ж.) жұмысы, оның ішінде сол мезгілге белгілі шифрлау алгоритмдері қалыптастырылған және жалпыланған. Ол екі үлкен емес, бірақ маңызды жаңалық ашты: полибиандық төртбұрышты толтыру тәсілі (бірінші позициялар жеңіл есте сақталатын кілттік сөзбен толықтырылады, ал қалғандары – алфавиттың қалған әріптерімен) және әріптер жұптарын (биграммаларды) шифрлау.

Қарапайым, бірақ берік көпалфавитты ауыстыру (биграммаларды ауыстыру) тәсілі болып XIX ғ. басында *Чарльз Уитстонмен* ашылған *Плейфер шифры* есептеледі. Уитстон

маңызды жақсарту енгізді – «қос төртбұрышпен» шифрлау. *Плейфер* және *Уитстон* шифрлары қолмен жасалған криптографиялық талдауға берік болғандықтан, бірінші әлемдік соғысқа дейін қолданыста болды.

XIX ғ. голландтық *Огюст Керкгоффс* кәзіргі уақытқа дейін өзекті болып қалған криптографиялық жүйелерге басты талапты қалыптастырды: *шифрлардың құпиялығы алгоритмнің емес, кілттің құпиялығында негізделену керек.*

Ақырында, криптографиялық беріктіліктің оданда жоғары деңгейін қамтамасыз еткен, сонымен қатар шифрлау процессін автоматтандыруға (механикаландыру мағынасында) мүмкіндік берген ғылымға дейінгі криптографияның соңғы сөзі болған *роторлық криптожүйелер.*

Осындай жүйелердің бірінші қатарына жататынның бірі 1790 жылы болашақ президент *Томас Джефферсон* ойлап тапқан механикалық машина. Роторлық машина көмегімен көпалфавитты ауыстыру айналатын роторлардың өзара күйлердің вариациясымен жүзеге асырылады. Роторлардың әрқайсысы оның ішінде «тігілген» ауыстыруды жүзеге асырады.

Роторлық машыналар тәжірибелік тарауын тек XX ғасырдың басында алды. Тәжірибелік түрде қолданатын машиналардың бірі 1917 жылы *Эдвард Хебернмен* құрылған және *Артур Кирх* жетілдірген немістік *Enigma* машинасы. Роторлық машиналар Екінші әлемдік соғыс уақытында белсенді қолданылды. Немістік *Enigma* машинасынан басқа *Sigaba* (АҚШ), *Typex* (Ұлыбритания), *Red, Orange* және *Purple* (Жапония) құрылғылары қолданылды. Роторлық жүйелер – өте берікті шифрларды қарапайым жүзеге асырғандықтан, формалды криптографияның шыңы болып табылады. Роторлық жүйелерге жемісті криптографиялық шабуылдар XX ғасырдың 40 жылдардың басында *электрондық есептеу машиналардың (ЭЕМ)* пайда болуымен мүмкін болды.

Ғылыми криптологияның (1930 - 1960 жж.) басты айырмашылығы – криптографиялық беріктілікті қатал математикалық негіздеумен криптографиялық жүйелердің пайда болуы. 30 жылдардың басына криптологияның ғылыми негізі болып табылатын математика тараулары біржолата қалыптасты: ықтималдық теориясы және математикалық статистика, жалпы алгебра, сандар теориясы, ал алгоритмдер теориясы, ақпарат теориясы, кибернетика белсенді дамуды бастады. Криптография

және криптографиялық талдауға ғылыми қор жасаған *Клод Шеннонның «Құпия жүйелерде ақпарат теориясы»* (1949 ж.) жұмысы өзіндік суайрығы болды. Сол уақыттан бастап *криптологияны* ақпараттың құпиялығын қамтамасыз ету үшін түрлендіру ғылымы деп атап кетті. 1949 жылға дейінгі криптография және криптографиялық талдаудың даму кезеңін *ғылымның алдындағы криптология* деп атап кетті. *Шеннон «шашырату»* және *«араластыру»* түсініктерін енгізді, керекті беріктілігі бар криптографиялық жүйелерді құру мүмкіндігін негіздеді.

1960 жылдары жетекші криптографиялық мектептер *блокты шифрларды* құруға келді, олар роторлық криптографиялық жүйелерге қарағанда беріктірек, бірақ тәжірибелік жүзеге асыруы тек цифрлық электронды құрылғылар түрінде мүмкін

Компьютерлік критография (1970 жылдардан бастап) өзінің пайда болумен *қолдан жасалған* және *механикалық шифрларға* қарағанда, шифрлаудың жоғары жылдамдығы бар кезде бірнеше деңгейге жоғары криптографиялық беріктілікті қамтамасыз ететін криптографиялық жүйелерді жүзеге асыру үшін жеткілікті өнімділікпен есептеу құралдарының пайда болуына міндетті.

Қуатты және шағын есептеу құралдары пайда болуымен тәжірибелік пайдалану мүмкіндігіне ие болған криптографиялық жүйелердің бірінші класы - блокты шифрлар. XX ғасырдың 70-жылдары *DES – Data Encryption Standard (Деректерді шифрлау стандарты) американдық шифрлау стандарты* құрылды. Ол 1978 жылы қабылданды. Оның авторларының бірі *Хорст Фейстель* (*IBM* қызметкері) блокты шифрлар үлгісін сипаттады, оның негізінде басқа беріктілігі жоғарырақ симметриялық криптожүйелері құрылды, соның ішінде 1989 жылы КСРО құрылған *ГОСТ 28147-89 шифрлау стандарты.*

DES пайда болуымен криптографиялық талдауда байыды, американдық алгоритмге шабуыл жасау үшін бірнеше жаңа криптографиялық талдау (сызықты, дифференциалды және т.б.) түрлері құрылды, олардың тәжірибелік жүзеге асуы тек қуатты есептеу жүйелері пайда болуымен мүмкін болды.

XX ғасырдың 70 жылдардың ортасында заманауи криптографияда кәдімгі жарылу болды - екі жақ ортасында құпия кілтті жіберуді талап етпейтін *асимметриялық криптографиялық жүйелердің* пайда болуы. Бұл жерде 1976 жылы *Уитфилд Диффи*

және *Мартин Хеллман* жариялымға шығарған «Заманауи криптографияда жаңа бағыттар» жұмысы шыққан нүкте деп есептеледі. Бұл жұмыста алғашқы рет құпия кілтпен алмасусыз шифрланған ақпаратпен алмасу принциптері қалыптастырылды. Асимметриялық криптографиялық жүйелер идеясына тәуелсіз *Ральф Меркли* жетті. Бірнеше жылдан кейін *Рон Ривест, Ади Шамир* және *Леонард Адлеман RSA* жүйесін ашты - беріктілігі үлкен сандарды факторизациялау проблемасында негізделген бірінші тәжірибелік асимметриялық криптографиялық жүйе. *Асимметриялық криптография* бірден бірнеше қолданбалы бағытты ашты, соның ішінде *электрондық цифрлық қолтаңба* және *электронды ақша* жүйелері.

1980-1990 жылдары криптографияның жаңа бағыттары пайда болды: *ықтималдық шифрлау, кванттық криптография* және басқа. Олардың тәжірибелік құндылығын сезіну әлі алда. Симметриялық криптографиялық жүйелерді жетілдіру есебі маңызды болып қалып тұр. Осы периодта фейстелдік емес шифрлар *(SAFER, RC6* және басқа) құрылды, ал 2000 жылы ашық халықаралық байқауынаң кейін АҚШ жаңа ұлттық шифрлау стандарты қабылданды – *AES (Advanced Encryption Standard* – Жетілдірілген шифрлау стандарты).

Криптография конфиденциалдықты қамтамасыз етудің және ақпараттың тұтастығын бақылаудың қуатты құралдарының бірі болып есептеледі. Көптеген жағдайларда ол қауіпсіздіктің бағдарламалық-техникалық реттеуіштер ішінде орталық орын алады. Мысалы, физикалық түрде қорғау өте қыйын портативті компьютерлер үшін ұрлау оқиғасы болса да тек криптография ақпараттың конфиденциалдығын кепілдеуге мүмкіндік береді.

1.3.3. Криптологияда пайдаланатын негізгі анықтамалар

Заңсыз пайдаланушылардан (рұқсатсыз қатынас құрудан) қорғау мақсатымен ақпаратты түрлендіру (шифрлау) әдістері мен тәсілдері *шифр* деп аталады.

Шифрлау – қорғалатын ақпаратқа шифрды қордану процессі, яғни шифрда анықталған ережелер көмегімен қорғалатын ақпаратты (ашық ақпаратты, деректерді) шифрланған хабарға (шифрланған деректер) түрлендіру.

Керішифрлау – шифрлауға қарсы процесс, яғни шифрда анықталған ережелер көмегімен шифрланған хабарды қорғалатын ақпаратқа түрлендіру.

Кілт – нақты хабарды шифрлау үшін қолданатын түрлендіруге жауап беретін шифрдың маңызды компоненті. Әдетте кілт болып әріптік немесе сендық тізбегі табылады. Бұл тізбек шифрлау алгоритміне «баптайтын» сияқты.

Хаттама – есепті бірігіп шешу үшін екі немесе одан көп жақ орындайтын қадамдар реті. Барлық қадамдар қатаң ретпен орындалады, олардың біреуіде алдынғысы бітпей орындалмау керек. Сонымен қоса, әр хаттамаға аз дегенде екі жақ қатысу керек. Ең соңғысы, хаттама міндетті түрде кейбір мақсатқа жету үшін тағайындалған.

Негізінде криптографиялық алгоритм жатқан хаттама *криптографиялық хаттама* деп аталады. Алайда криптографиялық хаттаманың мақсаты ақпаратты басқалардан құпия сақтау ғана болмайды. Криптографиялық хаттаманың қатысушылары бір бірінен құпиялары жоқ жақын дос болу мүмкін, бірақ бір біріне ештеңе айтпайтын бітіспес душпан да болу мүмкін. Соған қарамастан, оларға бірлестік келісімге қол қою немесе өзін куәландыру қажет болу мүмкін. Бұл жағдайда криптография хаттаманың қатысушылары емес басқа адамдармен тыңдауды табу немесе болдырмау үшін, аялақтықты болдырмау үшін қажет. Сондықтан, криптографиялық хаттама келесіні қамтамасыз етуі жиі талап етіледі: оның қатысушылары хаттамамен анықталғаннан басқа ештеңе білмеу және жасамау керек.

Кез келген криптографиялық хаттаманың негізі *криптографиялық алгоримдер.*

Әр түрлендіру *кілтпен* бірмәнді анықталады және кейбір *криптографиялық алгоритммен* сипатталады. Бір криптографиялық алгоритм әртүрлі режимдерде шифрлау үшін қолдану мүмкін. Солай шифрлаудың әртүрлі тәсілдері жүзеге асырылады. Шифрлаудың әр режимінің артықшылықтары және кемшіліктері бар. Сондықтан режимді таңдау нақты жағдайдан тәуелді. Керішифрлау кезінде хабарды шифрлауға қолданған алгоритмнен жалпы жағдайда айырмашылы бар болу мүмкін алгоритмі қолданылады. Осыған сәйкес *шифрлау* және *керішифрлау кілттерінің* айырмашылығы болу мүмкін. Шифрлау және керішифрлау алгоримдердің жұбын *криптографиялық жүйе* деп

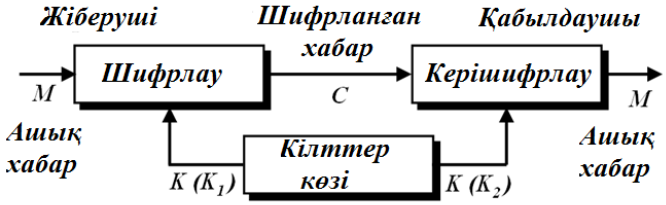
атайды, ал оларды жүзеге асыратын құрылғыларды – шифрлаушы құрылғылар деп атайды.

1.3.4. Криптографиялық жүйенің жалпылама сұлбасы

Криптографиялық жүйе – ақпаратты қорғау мақсатымен криптографиялық түрлендіруді орындайтын бағдарламалық, аппаратты немесе бағдарламалық-аппаратты түрде жүзеге асырылған жүйе [8, 22].

Жіберуші қабылдаушыға хабар жібергісі келді дейік. Осыған қоса, жіберуші өз хабарын қауіпсіз жібергісі келеді: ол тасымалдаушыны жолай ұстаған қасқой хабардың мазмұнын біле алмайтынына сенімді болғысы келеді. Жіберілетін хабар *ашық хабар (мәтін, деректер)* деп аталады. Хабардың түрін, оның мәнін жасыру мақсатымен өзгерту *шифрлау* деп аталады. Шифрланған хабар *шифрмәтін (деректер, криптограмма)* деп аталады. Шифрланған мәтінді ашық хабарға түрлендіру процессі *керішифрлау* (криптографиялық талдау кезінде керішифрлау) деп аталады.

Жіберілетін ақпаратты (хабарды, мәтінді, деректерді) шифрлауды және керішифрлауды қамтамасыз ететін криптографиялық түрлендірудің жалпылама сұлбасы 1.1 суретінде көрсетілген.



1.1 сурет - Криптографиялық түрлендірудің жалпылама сұлбасы

Ашық мәтінді *M* (*message* – хабар) немесе *P* (*plaintext* – ашық мәтін) деп белгілейміз. Бұл биттер ағыны, мәтіндік файл, биттік бейне, цифрланған дыбыс, цифрлық бейнекөрініс және басқа болу мүмкін.

Шифрмәтінді *C* (*ciphertext* – *шифрланған мәтін*) деп белгілейміз. Бұл да екілік деректер, кей кезде *M* бірдей көлемді, кей

кезде үлкенрек. Егер шифрлау қысумен болса, онда *C* кішірек болу мүмкін. Бірақ шифрлаудың өзі ақпаратты қысуды қамтамасыз етпейді. *E шифрлау функциясы* ашық мәтінге шифрланған мәтінді құрып әсер етеді:

$$C = E(M). \tag{1.1}$$

Ашық мәтінді шифрланған мәтін бойынша қалпына келтіру *керішифрлау* деп аталады және *D* керішифрлау функциясы көмегімен орындалады:

$$M = D(C). \tag{1.2}$$

Хабарды шифрлау және ары қарай керішифрлаудың мағынасы ашық мәтінді қалпына келтіру болғандықтан, келесі теңдік орындалады:

$$M = D(E(M)).$$

Шифр деп аталатын криптографиялық алгоритм шифрлауға және керішифрлауға қолданатын математикалық функция (мысалы, (1.1) және (1.2) өрнектерінде *E* және *D* функциялары) болады.

Егер алгоритмнің қауіпсіздігі алгоритмнің өзін құпия сақтауда негізделсе, онда бұл *шектелген алгоритм.* Шектелген алгоритмдер тек тарихи қызығушылықты тұдырады, бірақ олар бүгінгі стандарттарға мүлдем сәйкес келмейді. Үлкен немесе өзгеретін пайдаланушылар тобы бұндай алгоритмдерді қолдана алмайды, себебі пайдаланушы топтан шыққан сайын оның мүшелері басқа алгоритмді қолдану керек. Егер сырттан біреу құпияны кездейсоқ біліп қойса, алгоритмді ауыстыру керек.

Заманауи криптография бұл проблемаларды *K* кілт көмегімен шешеді. *Кілт* - берілген алгоритм үшін барлық мүмкінді нұсқалардан біреуін таңдауды қамтамасыз ететін, деректерді криптографиялық түрлендіру алгоритмнің кейбір параметрлерінің нақты құпия күйі. Мүмкінді кілттер көптігін *кілттер кеңістігі* деп атайды.

Кілтті қолдануды есептеумен шифрлау (1.1) және керішифрлау (1.2) функциялары келесідей жазылады:

$$C = E_K(M) \tag{1.3}$$

және

$$M = D_K(C), \tag{1.4}$$

келесі орындалу керек:

$$M = D_K(E_K(M)). \tag{1.5}$$

Шифрлау және керішифрлау кезінде кейбір алгоритмдер үшін әртүрлі кілттер қолданылады. Бұл жағдайда (1.3) және (1.4) келесі түрде жазылады:

$$C = E_{K_1}(M), \quad M = D_{K_2}(C), \tag{1.6}$$

ал теңдік (1.5):

$$M = D_{K_2}(E_{K_1}(M)).$$

Шифрлауға және керішифрлауға арналған ақпарат ретінде кейбір *алфавитте* құрылған мәтіндер (хабарлар) қарастырылады. Бұл терминдер астында келесіні түсінеді. *Алфавит* – ақпаратты кодтау үшін қолданатын білгілердің шегі бар көптігі. *Мәтін (хабар, деректер)* – алфавит элементтерінен реттелген жинағы.

Заманауи ақпараттық жүйелерде қолданатын алфавиттер мысалы ретінде келесілерді келтіруге болады:

- алфавит Z_{26} – ағылшын алфавитының *26* әріпі (бос орынсыз);
- алфавит Z_{32} – орыс тілінің *32* әріпі ("ё" әріпісіз) және бос орын;
- алфавит Z_{256} – *ASCII* стандартты кодтарына кіретін *256* символ;
- алфавит Z_{16} – он алтылық алфавиттың *{0,1,...,f} 16* символы;
- алфавит Z_2 – екілік алфавиттың *2* символы *{0,1}*.

(1.3), (1.4) және (1.6) өрнектермен сипатталатын алгоритмдердің қауіпсіздігі алгоритмнің ерекшеліктерінде емес, кілттерде толық негізделген. Бұл, алгоритм жариялану және талдану мүмкін екенін білдіреді. Бұл алгоритмді қолданатын түліктер кең тираждану мүмкін. Бірінші рет голландтық *О. Керкгоффспен* XIX ғасырда қалыптастырылған криптографиялық талдаудың фундаменталды ережесі келесі: шифрдың беріктілігі (криптографиялық жүйенің) тек кілттің құпиялық дәрежесімен ғана анықталу керек. Басқа сөзбен, *Керкгоффс ережесі* бойынша құпия кілттен басқа шифрлау және керішифрлау алгоритмі қарсыластың криптографиялық талдаушысына толық белгілі. Криптографиялық түрлендіру жинағын жүзеге асыратын криптографиялық жүйе әдетте ашық жүйе ретінде қарастырылады. Бұл ақпаратты қорғау

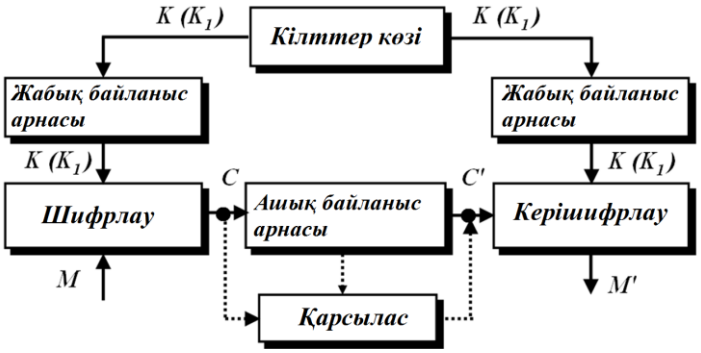
технологиясының маңызды принципын бейнелейді: құпия (конфиденциалды) ақпараттың сыртқа кету жағдайда жүйенің қорғалғандығы тез өзгертілмейтін нәрседен тәуелді болмау керек. Әдетте криптографиялық жүйе аппараттық және бағдарламалық құралдар жинағы, оны тек уақыт пен құралдардың көптеген шығындарымен өзгертуге болады, ал кілт жеңіл өзгертілетін объект болып табылады. Сондықтан криптографиялық жүйенің беріктілігі тек кілттің құпиялық деңгейімен және шифрдың әртүрлі шабуылдарға қарсы тұру берітілігімен анықталу керек.

1.4. К. ШЕННОННЫҢ ҚҰПИЯЛАНҒАН БАЙЛАНЫС ТЕОРИЯСЫНЫҢ НЕГІЗДЕРІ

Криптографиялық алгоритмдерді қарастырудың алдында, криптография шеңберінде классикалық болып танылған сұрақтарға көңіл бөлу қажет, ол деген – *құпияланған байланыс жүйелерін* құру негіздеріне.

Құпияланған байланыс жүйелері деп криптографиялық түрлендіру көмегімен ақпарат мәні жасырылатын ақпаратты жіберу жүйесін түсінеміз [34]. Бұл жерде ақпаратты жіберу фактісі жасылырмайды. Әр құпияланған байланыс жүйесінің негізінде конфиденциалдықты сақтаудың негізгі құрылғысы ретінде шифрлау алгоритмдерін қолдану.

К. Шеннон хабар көзі *M* ашық мәтінді тұындыратын үлгіні (1.2 сурет) қарастырды. Кілт көзі *K* кілтін генерациялайды.



1.2 сурет - Шифрланған хабарларды жіберу жүйесінің үлгісі

Шифрлайтын құрылғы *M* ашық мәтінін *K* кілт көмегімен *C: C = E_K (M)* шифрланған мәтінге түрлендіреді. Керішифрлайтын құрылғы *C* шифрланған мәтінін алып *M = D_K(C)* кері операциясын орындайды.

Қарсыластың криптографиялық талдаушысының мақсаты ашық мәтін және/немесе кілтті шифрланған мәтінді талдау негізінде алу.

Шеннон теориялық және тәжірибелік құпиялық сұрақтарын қарастырды. Теориялық құпиялықты анықтау үшін Шеннон келесі сұрақтарды қалыптастырды:

1. Егер қарсыластың талдаушысы уақытпен шектелмеген болса және криптограммаларды талдау үшін барлық қажетті құралдары болса, жүйе қаншалықты берік болады?

2. Криптограммада бір ғана шешім бар ма?

3. Шешім бір ғана болу үшін криптографиялық талдаушыға криптограммалардың қандай көлемін жолай ұстау керек?

Бұл сұрақтарға жауап беру үшін *Шеннон аса құпиялық* түсінігін келесі шарт көмегімен енгізді: барлық *C* үшін апостериорды ықтималдықтар априорды ықтималдықтарына тең, яғни шифрланған мәтінді жолай ұстау қарсыластың криптографиялық талдаушысына ешқандай ақпаратты бермейді. *Байес теоремасы* бойынша:

$$p(M,C) = p(M)\cdot p(C/M) = p(C)\cdot p(M/C), \qquad (1.7)$$

бұл жерде *p(M) – M* хабарының априорды ықтималдығы*; p(C/M) – M* хабары таңдалды деген шартымен *C* криптограмманың шартты ықтималдығы, яғни *M* хабарын *C* криптограмасына аударатын барлық кілттер ықтималдықтарының қосындысы; *p(C) – C* криптограммасын алудың ықтималдығы; *p(M/C) – C* криптограммасы жолай ұсталды шартымен *M* хабарының апостериорды ықтималдығы.

(1.7) өрнегінен келесі шығады:

$$p(M/C) = \frac{p(M)\cdot p(C/M)}{p(C)}. \qquad (1.8)$$

Криптографиялық жүйенің аса құпиялығы үшін барлық *M* және *C* үшін *p(M/C)* және *p(M)* мәндері тең болу қажет, яғни *p(M/C) = p(M)*. Сондықтан, (1.8) талдаудан теңдіктің біреуі орындалу қажет:

- *p(M) = 0* (*p(M)* кез келген мәндері үшін теңдік орындалу талап етілгендіктен, бұл шешім шығарылып тасталу керек);

- кез келген *M* және *C* үшін *p(C/M) = p(C)*.

Керісінше, егер $p(C/M) = p(C)$, онда $p(M/C) = p(M)$, және жүйе аса құпия. Сонымен, келесіні қалыптатыруға болады: барлық M және C үшін $p(C/M) = p(C)$, яғни $p(C/M)$ M тәуелді болмау керектігінде аса құпиялық үшін қажетті және жеткілікті шарт.

Басқа сөзбен, M_i хабарын берілген C шифрланған хабарға аударатын барлық кілттер ықтималдығы M_j хабарын барлық M_i, M_j және C үшін тұра сол C шифрланған хабарға аударатын барлық кілттердің толық ықтималдығына тең.

Ары қарай, қанша M хабары бар сонша C шифрланған хабар болу қажет, себебі тіркелген i үшін E_i бейнесі барлық M және кейбір C арасында өзара-бірмәнді сәйкестікті береді. Аса құпия жүйе үшін әр C және кез келген M үшін

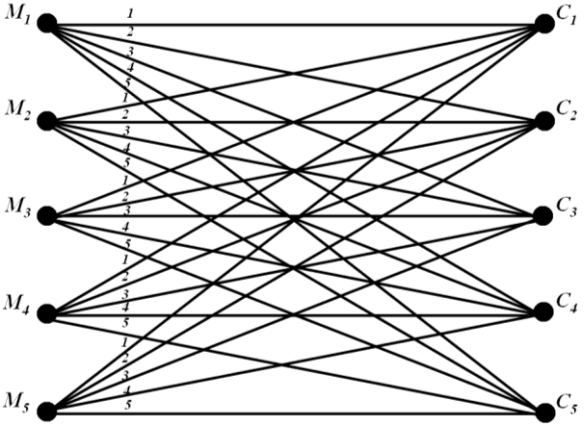
$$p(C/M) = p(C) \neq 0.$$

Сондықтан, берілген M кез келген C бейнелейтін кемінде бір кілт табылады. Бірақ тіркелген M әртүрлі C бейнелейтін барлық кілттер әртүрлі болу керек. Сондықтан, *әртүрлі кілттер саны M хабарлар санынан аз болмау керек.*

Келесі мысал көрсеткендей, хабар саны кілт санына нақты тең болғанда аса құпиялықты алуға болады. M_i C_i сияқты *1-ден n-ға* дейінгі сандармен нөмірленген болсын және n кілт қолданылсын. Сонда:

$$E_i(M_j) = C_s,$$

бұл жерде $s = (i + j) \bmod n$.



1.3 сурет – Жетілдірілген криптографиялық жүйе

Бұл жағдайда $p(M/C) = 1/n = p(C)$ теңдік әділ және жүйе аса құпия болыда. Осындай жүйенің бір мысалы 1.3 суретінде көрсетілген, бұл жерде $s = (i + j - 1) \bmod 5$.

Шифрланған хабар саны алғашқы хабар санына және кілт санына тең аса құпия жүйелер келесі екі қасиетпен сипатталады:

1) әр *M* әр *C* тек бір сызықпен байланысады;

2) барлық кілттер теңықтималды.

Сонымен, осындай жүйенің матрицалық бейнесі *«латын төртбұрышы»* болады.

«Математикалық ақпарат теориясында» ақпаратты сапалы өлшеу энтропия көмегімен ыңғайлы екені көрсетілген. Егер $p_1, p_2,...,p_n$ ықтималдықтары бар мүмкіндіктер жинағы бар болса, онда энтропия келесі өрнекпен көрсетіледі

$$H = -\sum p_i \cdot \log\, p_i \,.$$

Құпия жүйеге екі статистикалық таңдау кіреді: хабарды таңдау және кілтті таңдау. Хабарды таңдау кезінде құрылатын ақпарат санын *H(M)* арқылы өлшеуге болады

$$H(M) = -\sum P(M) \cdot \log P(M),$$

бұл жерде, қосу барлық мүмкін хабарлар бойынша орындалады.

Тұра солай, кілт таңдаумен байланысты белгісіздік келесі өрнекпен анықталады

$$H(K) = -\sum P(K) \cdot \log P(K).$$

Жоғарыда көрсетілген аса құпия жүйелерде хабардағы ақпарат саны ең көп дегенде $\log n$ (бұл шама теңықтималды хабар үшін) тең. Бұл ақпарат кілт белгісіздігі $\log n$ кіші болмаса ғана толық жасырылады. Бұл төменде жиі кездесетін жалпы принциптің бірінші мысалы: берілгін кілт белгісіздігі кезінде оны асуға болмайтын шек болады – шешімге енгізілу мүмкін белгісіздік саны кілт белгісіздігінен аса алмайды.

Егер хабар саны шексіз болса, онда жағдай қыйындайды. Мысалы, хабарлар сәйкес *марк процессімен* шексіз әріптер жинағы түрінде туындалсын. Ешқандай шекті кілт аса құпиялықты бермейтіні анық. Онда, кілт көзі кілтті тұра солай туындатсын дейік, яғни шексіз символдар жинағы ретінде.

Әрі қарай, L_M ұзындығы бар хабарды шифрлау және керішифрлау үшін тек анықталған L_K ұзындығы бар кілт керек дейік. Хабар алфавитының әріптер санының логарифмы R_M, ал кілттікі R_K болсын. Онда соңғы жағдайда аса құпиялық үшін келесі теңсіздік орындалу керек

$$R_M \cdot L_M \leq R_K \cdot L_K.$$

Бұл қорытындылар хабар ықтималдығы белгісіз немесе кез келген болжамымен жасалады. Бұл жағдайда аса құпиялық болу үшін кілт мүмкін хабарлардың толық санынан тәуелді болады.

Егер хабар кеңістігінде *«Математикалық ақпарат теориясында»* қабылданған мәнінде R хабарларды құрудың анықталған жылдамдығы бар болған жағдайда тіркелген белгілі статистикалық байланыстар бар болса, онда керекті кілт көлемін орташа R/R_M ретке төмендетуге болатын еді. Шынында, хабарды артықтықты алып тастайтын және хабардың орташа ұзындығын сол керек ретке қысқартатын түрлендіргіш арқылы өткізуге болады. Содан кейін, нәтижеге *Вернам шифрын* қолдануға болады. Хабар әріпіне қолданылатын кілт көлемі статистикалық түрде R/R_M кішірейеді және бұл жағдайда кілт көзі және хабар көзі нақты келісілген – кілттің бір биті хабар ақпаратының бір битін жасырады. *«Математикалық ақпарат теориясында»* қолданылатын әдістер көмегімен ең жаксы жағдайда осыған жетуге болады.

1.5. ХАБАРЛАРДЫ ШИФРЛАУ ӘДІСТЕРІН ЖІКТЕУ

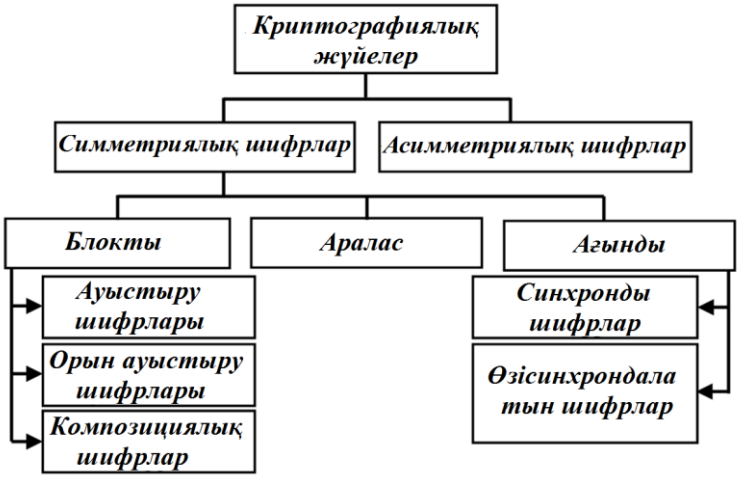
Классикалық немесе біркілтті криптография симметриялық шифрлау алгоритмдерін қолдануға сүйенеді, бұл жерде шифрлаудың және керішифрлаудың тек орындау ретінде және кейбір қадамдар бағытында айырмашылығы болады. Бұл алгоритмдер сол құпия элементті (кілтті) қолданады, және екінші әрекет (керішифрлау) біріншінің (шифрлаудың) қарапайым айналысы болады. Сондықтан, алмасудың әр қатысушысы хабарды шифрлай және керішифрлай алады. Сондай жүйенің сұлбалық құрылымы 1.1 суретінде көрсетілген.

Жіберуші жағында M хабар көзі және K кілт көзі бар. Кілт көзі берілген жүйенің барлық мүмкін кілттерінен нақты K кілтін таңдайды. Ол кілт K кейбір тәсілмен қабылдаушы жаққа жіберіледі,

және де оны жолай ұстауға болмайды деп есептеледі, мысалы, кілт арнайы курьер арқылы жіберіледі (сондықтан симметриялық шифрлау *жабық кілтті шифрлау* деп аталады). Хабар көзі кейбір *M* хабарын қалыптастырады, ол содан кейін таңдалған кілт *K* арқылы шифрланады. Шифрлау процедурасы нәижесінде *C* шифрланған мәтін алынады (*криптограмма* деп аталады). Ары қарай криптограмма *C* байланыс арнасымен жіберіледі. Байланыс арнасы ашық (қорғалмаған) болғандықтан, мысалы, радиоарна немесе компьютерлік желі, жіберілген хабарды қарсылас жолай ұстау мүмкін. Қабылдайтын жақта *C* криптограммасы *K* кілт көмегімен керішифрланады және *M* кіріс хабары алынады.

Табиғи тілдердің үлкен артықтығы арқылы тікелей шифрланған мәтінге мәні бар өзгерісті енгізу өте қыйын, сондықтан классикалық криптография осымен қоса жалған ақпаратты тықпалаудан қорғауды қамтамасыз етеді. Егер хабарды түрлендіруден сенімді қорғау үшін табиғи артықтық жеткілікті болмаса, онда хабарға *имитоендірме* деп аталатын арнайы бақылау комбинацияны қосу арқылы жасанды жоғарлату мүмкін.

Әртүрлі шифрлау әдістері белгілі (1.4 сурет).



1.4 сурет – Хабарды шифрлау әдістерінің жіктеліу

Тәжірибеде ауыстыру, орын ауыстыру алгоритмдері және құрамдастыру әдістері жиі қолданылады.

Криптографиялық әдістер шифрлау алгоритмі типі бойынша екі үлкен топқа бөлінеді (1.4 сурет): *симметриялық* және *асимметриялық*. Осыған қоса, алгоритмдер түрлендіру типі, ақпаратты өңдеу тәсілі бойынша бөлінеді.

Симметриялық криптографиялық жүйелерде шифрлау және керішифрлау бір кілт арқылы орындалады. Және осыған сәйкес кілтті құпия сақтау керек (бұл жерден симметриялық криптографиялық жүйелердің басқа аты – *құпия кілті бар криптографиялық жүйелер*).

Асимметриялық криптографиялық жүйелерде екі әртүрлі кілт бар: біреуі шифрлау үшін қолданылады және *ашық* деп аталады, басқасы – керішифрлау үшін, *жабық* деп аталады. Асимметриялық криптожүйелердің басты айырмашылығы келесіде: ашық кілтпен шифрлаған адам да хабарды құпия кілтсіз керішифрлай алмайды. Сондықтан бұл жүйелер *асимметриялық* немесе *ашық кілтті жүйелер* деп аталады.

Криптографиялық жүйелердің екі типін де біріктіретін криптографиялық жүйелерді гибридті деп атайды. Оларда, ереже сияқты, хабар мәтіні симметриялық криптографиялық жүйені қолданумен шифрланады, ал симметриялық криптографиялық жүйемен қолданылған құпия кілт асимметриялық криптографиялық жүйені қолданумен шифрланады.

Кіріс ақпарат тізбегін өңдеу типі бойынша жүйелер *ағындыға* және *блоктыға* бөлінеді. Ағынды жүйелерде ашық мәтіннің әр символы немесе биті шифрланған мәтін символына ағынмен шифрланады. Блокты жүйелерде хабарлар анықталған ұзындығы бар блоктармен түрлендіріледі.

Орын ауыстыру әдістерінде кіріс мәтіннің символдары анықталған ереже бойынша бір бірімен орын ауыстырады. *Ауыстыру әдістерінде* ашық мәтін символдары шифрланған мәтіннің эквиваленттерімен ауыстырылады. Шын мәнде, орын ауыстыру және ауыстыру шифрлары кейбір кірпиш болады, олардан басқа әртүрлі берік шифрлар құрылады.

Шифрлау сенімділігін жоғарлату мақсатымен бір әдіспен шифрланған мәтін басқа шифрмен тағы шифрлану мүмкін. Бұл жағдайда *құрамдас* немесе *композициялық шифр* болады. Кәзіргі уақытта тәжірибеде қолданылатын блокты немесе симметриялық ағынды шифрлар құрамдас шифрларға жатады, олар хабарды шифрлау үшін бірнеше операцияны қолданады.

Заманауи криптографияның компьютерліктің алдындағы криптографиядан негізгі айырмашылығы келесіде: бұрын криптографиялық алгоритмдер табиғи тілдерінің символадарымен, мысалы, ағылшын немесе орыс тілінің әріптерімен жұмыс істеді. Бұл әріптер анықталған ереже бойынша орын ауыстырады немесе басқалармен ауыстырылды. Заманауи криптографиялық алгоритмдерде екілік белгілерге операциялар қолданылады, яғни нөл мен бірлерге. Кәзіргі уақытта шифрлау кезіндегі негізгі операциялар орын ауыстыру немесе ауыстыру, және де шифрлаудың сенімділігін жоғарлату үшін бұл операциялар бірге қолданылады (құрамдастырылады) және көп рет циклды түрде қайталанады.

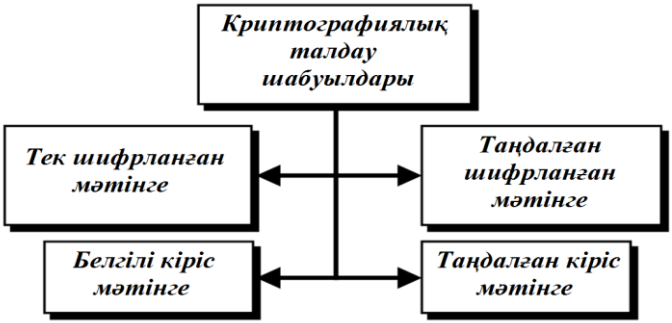
Құрамдастырылған немесе *композициялық шифрлар* негізінде жатқан идея криптографиялық берік жүйені қарапайым криптографиялық түрлендірулерді көп рет қайталау жолымен пайдалануда болды. *К. Шеннон* түрлендіру ретінде *ауыстыруларды* (*substitution*) және *транспозицияларды* (*permutation*) қолдануды ұсынды. Бұл түрлендірулерді көп рет пайдалану берік шифрларға тән екі қасиетті қамтамасыз етуге мүмкіндік берді: *шашырату* (*diffusion*) және *араластыру* (*confusion*).

Шашырату ашық мәтіннің бір белгісінің, сонымен қоса кілттің бір әріпінің шифрланған хабардың бірталай белгілеріне әсерін таратуды ескереді. Шифрда бұл қасиеттің бар болуы, бір жағынан, ашық мәтін белгілері арасында статистикалық тәуелдікті жасыруға мүмкіндік береді, басқаша айтқанда, кіріс тілдің артықтығын бүкіл мәтінге тарату арқылы қайта бөлуді, ал басқа жағынан – белгісіз кілтті бөліктермен қалпына келтіруге мүмкіндік бермейді. Мысалы, символдарды қарапайым орын ауыстыру биграммалардың, үшграммалардың және ары қарай кездесулердің жиілігін жасыруға мүмкіндік береді.

Араластыру мақсаты – кілт және шифрланған мәтін арасындағы тәуелдікті күрделендіру. Криптографиялық талдаушы араластырылған мәтінді статистикалық талдау негізінде қолданылған кілт туралы кез келген ақпарат санын алмау керек. Әдетте араластыру ауыстыру көмегімен жүзеге асырылады. Шашырату мен араластыруды бөлек қолдану қажетті беріктілікті қамтамасыз етпейді, берік криптографиялық жүйені тек оларды бірге қолдану нәтижесінде алуға болады.

1.6. КРИПТОГРАФИЯЛЫҚ ТАЛДАУ

Алдында байқалғандай, криптография – құпия кодтарды құру өнері және ғылымы, криптографиялық талдау – сол кодтарды бұзу өнері және ғылымы. Криптография әдістерін зерттеуге қосымша криптографиялық талдау әдістерін зерттеу қажет. Бұл басқа адамдардың кодтарын бұзу үшін емес, өз криптографиялық жүйелердің осал жерлерін бағалау үшін қажет. Криптографиялық талдауды зерттеу жақсы құпия кодтарды құруға көмектеседі. Криптографиялық талдаудың төрт жалпы шабуылдар типы бар, олар 1.5 суретінде көрсетілген [8].



1.5 сурет - Криптографиялық талдау шабуылдарын жіктеу

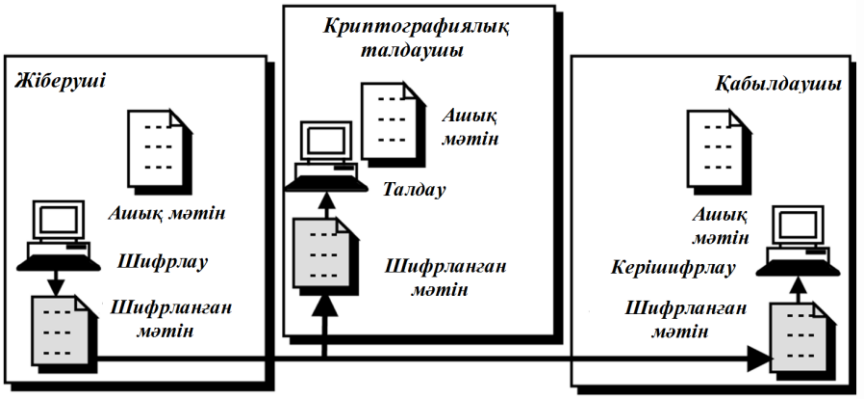
1.6.1. Шифрланған мәтінге шабуыл

Тек шифрланған мәтінге шабуылда криптографиялық талдаушы тек кейбір шифрланған мәтінге қол жеткізе алады. Ол лайықты кілтті және кіріс мәтінді іздеуге тырысады. Сонымен бірге, болжамға сәйкес, криптографиялық талдаушы алгоритмді біледі және шифрланған мәтінді жолай ұстай алады. Тек шифрланған мәтінге шабуыл – ең ықтималды, өйткені криптографиялық талдаушыға ол үшін тек шифрланған мәтіннің өзі керек. Шифр бұл шабуыл типіне шындап қарсы тұру қажет және қарсыласқа хабарды керішифрлауға мүмкіндік бермеу керек. Шабуыл процессін 1.6 сурет бейнелейді.

Шифрланған мәтінге шабуылда әртүрлі әдістер қолдану мүмкін. Олардың кейбіреуін қарастырайық.

«Қатқыл күш» шабуылы

«Қатқыл күш» шабуылы көмегімен немесе *толық кілттік іздеу әдісімен* криптографиялық талдаушы барлық мүмкін кілттерді қолдануды талпынады. Ол алгоритмді және көптеген кілттерді (мүмкін кілттер тізімін) біледі деп болжаймыз. Бұл әдісті қолдану жолында шифрланған мәтін жолай ұсталады және барлық мүмкін кілттер кіріс мәтін шыққанша қолданылады. «Қатқыл күш» шабуылды құру бұрын қыйын есеп болатын; бүгін компьютер көмегімен бұл жеңілірек болды. Бұл шабуыл типін болдырмау үшін, мүмкін кілттер саны өте үлкен болу қажет.



1.6 сурет – Тек шифрланған мәтінге шабуыл процессін түсіндіру

Статистикалық шабуыл

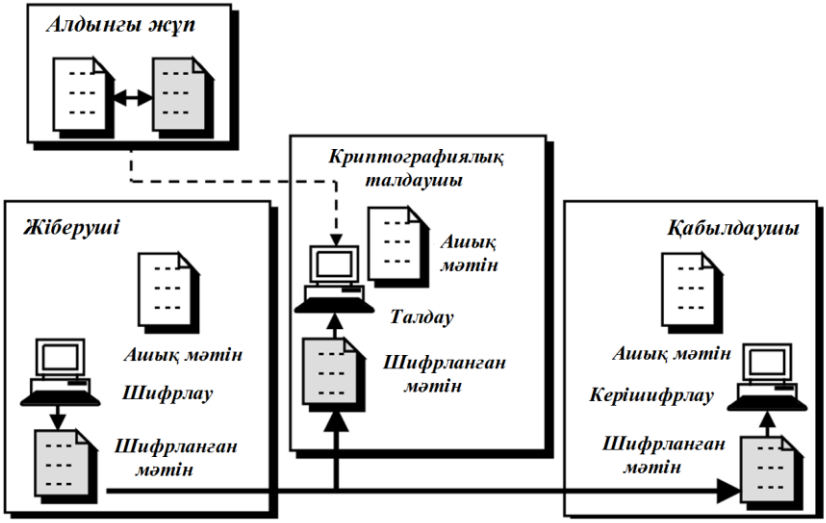
Статистикалық шабуылды бастау үшін криптографиялық талдаушы ашық мәтіннің тіліне тиісті кейбір сипаттамалардан пайданы таба алады. Мысалы, ағылшын мәтінінде ең жиі пайдаланатын әріп *E* екені белгілі. Криптографиялық талдаушы шифрланған мәтінде ең жиі пайдаланатын символды табады және ол кіріс мәтіннің қажетті символы *E* деп есептейді. Бірнеше жұптарды анықтағаннаң кейін талдаушы кілтті тауып хабарды керішифрлай алады. Бұл типті шабуылдарды болдырмау үшін, шифр тілдің сипаттамаларын жасыру керек.

Үлгі бойынша шабуыл

Кейбір шифрлар тіл сипаттамаларын жасырады, бірақ шифрланған мәтінде кейбір үлгілерді құрады. Криптографиялық талдаушы шифрды бұзу үшін үлгі бойынша шабыулды қолдану мүмкін. Сондықтан, қарастырылатын шифрланған мәтінді мүмкін бойынша белгісіз (абстаркты) болдыратын шифрларды қолдану маңызды.

1.6.2. Белгілі кіріс мәтінге шабуыл

Белгілі кіріс мәтінге шабуыл кезінде криптографиялық талдаушы жолай ұсталған шифрланған мәтінге қоса кейбір *«кіріс/шифрланған мәтін»* жұптарына 1.7 суретте көрсетілгендей қол жеткізе алады.



1.7 сурет – Белгілі кіріс мәтінге шабуыл процессін түсіндіру

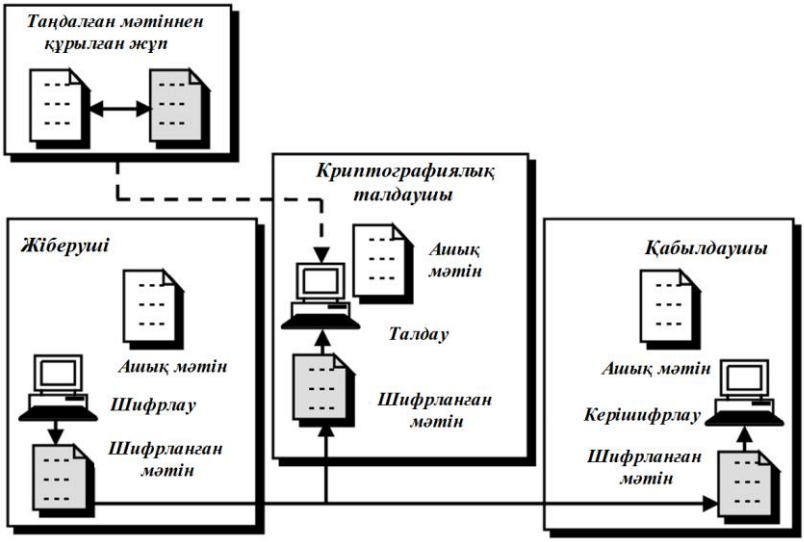
Кіріс/шифрланған мәтін жұптары алдын ала жиналған. Мысалы, жіберуші қабылдаушыға құпия хабарды жіберді, бірақ бірнеше уақыттан кейін хабар мазмұнын басқа адамдарға ашты. Криптографиялық талдаушы жіберуші өз кілтін өзгертпейді деп есептеп, кіріс мәтінді және шифрланған мәтінді сақтап қалады.

Оларды келесі жіберушіден қабылдаушыға жіберілетін құпия мәтінді бұзу үшін қолданады. Криптографиялық талдаушы ағындағы шифрланған мәтінді талдау үшін алдынғы жұптардың қатынастарын қолданады.

Тек шифрланған мәтінге шабуылда қолданылатын әдістер бұл жерде де қолданылу мүмкін. Бірақ бұл шабуылды жүзеге асыру оңайрақ, себебі криптографиялық талдаушыда талдау үшін көбірек ақпараты бар. Бірақ жіберуші өз кілтін өзгертіп немесе алдынғы хабарлардың мазмұнын ашпау мүмкін, - онда бұндай шабуыл мүмкін емес болады.

1.6.3. Таңдалған кіріс мәтінге шабуыл

Кіріс мәтінді таңдаумен шабуыл кіріс мәтінге шабуылға ұқсас, бірақ *«кіріс/шифрланған мәтін»* жұптары шабуыл жасаушымен таңдалған және жасалған. Бұл процессті 1.8 сурет көрсетеді.



1.8 сурет - Таңдалған кіріс мәтінге шабуылдар процессін түсіндіру

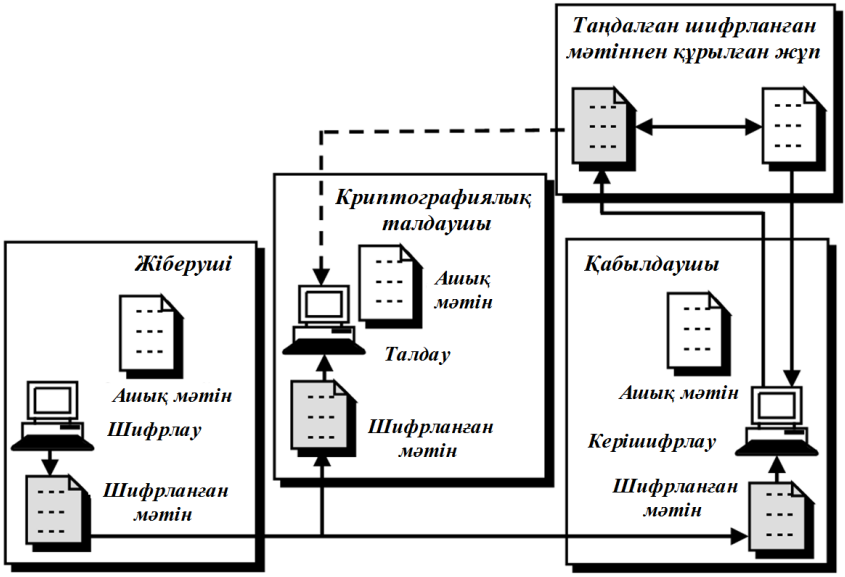
Бұл болу мүмкін, мысалы, егер криптографиялық талдаушы жіберушінің компьютеріне қатынас құра алса. Ол анықталған кіріс мәтінді таңдайды және компьютер көмегімен шифрланған мәтінді

құрады. Кілт жіберуші қолданатын бағдарламалық қамтамада болғандықтан, талдаушыда кілт болмайды.

Бұл шабуыл типі оңай, бірақ ол аз ықтималды, себебі бұл жерде *«егер»* өте көп.

1.6.4. Таңдалған шифрланған мәтінге шабуыл

Таңдалған шифрланған мәтінге шабуыл таңдалған кіріс мәтін шабуылына ұқсас, бұл жерде криптографиялық талдаушы анықталған шифрланған мәтінді таңдайды, оны *«шифрланған/алғашқы мәтін»* жұбын қалыптастыру үшін керішифрлайды (бұл жағдай болады, егер талдаушы жіберуші компьютеріне қол жеткізе алса). Бұл процессті 1.9 суреті көрсетеді.



1.9 сурет – Таңдалған шифрланған мәтін шабуыл процессін түсіндіру

Заманауи криптографиялық жүйелерді қолданумен криптографиялық шабуылдар келесі тарауларда қарастырылады.

Бақылау сұрақтары және тапсырмалары

1. Криптографиялық жүйелер қолдану көмегімен шешілетін проблемаларды айтыңыз.
2. Пайдаланушыға арналған ақпаратты қорғаудың негізгі есептері.
3. Ақпарат қасиеттерінің анықтамасын беріңіз: конфиденциалдық, тұтастық, шынайлық.
4. Келесілерге анықтама беріңіз: шифр; шифрлау кілті; криптографиялық алгоритм; криптографиялық жүйе.
5. Криптографиялық хаттамаға анықтама беріңіз.
6. Деректерді шифрлау кезінде (1.5) шартын орындалу қажеттілігін түсіндіріңіз.
7. Криптографиялық жүйенің аса құпиялығы үшін қажетті және жеткілікті шарттарын қалыптастырыңыз.
8. Деректерді шифрлау процессінде оларды шашыратуға түсініктеме беріңіз.
9. Деректерді шифрлау процессінде араластыру мақсаты неде?
10. Криптографиялық шабуыл деген не?
11. Криптографиялық шабуылдың қандай типтері бар?
12. Тек шифрланған мәтінге шабуылға сипаттама беріңіз. «Қатқыл күш» шабуылды түсіндіріңіз.
13. Тек шифрланған мәтінге шабуылға сипаттама беріңіз. Статистикалық шабуылды түсіндіріңіз.
14. Тек шифрланған мәтінге шабуылға сипаттама беріңіз. Үлгі бойынша шабуылды түсіндіріңіз.
15. Белгілі кіріс мәтінге шабуылға сипаттама беріңіз.
16. Таңдалған кіріс мәтінге шабуылға сипаттама беріңіз.
17. Таңдалған шифрланған мәтінге шабуылға сипаттама беріңіз.

2 тарау
ДӘСТҮРЛІ ТАРИХИ ШИФРЛАР

Симметриялық кілті бар дәстүрлі шифрларды екі үлкен категорияға бөлуге болады: *ауыстыру шифрлары* және *орын ауыстыру шифрлары*. Ауыстыру шифрында шифрланған мәтіннің бір символы басқа символға ауыстырылады; орын ауыстыру шифрында - алғашқы мәтіндегі символдар позициялары орындарын ауыстырады.

2.1. АУЫСТЫРУ ШИФРЛАРЫ

Ауыстыру шифры бір символды басқасымен ауыстырады. Егер алғашқы мәтіндегі символдар - алфавиттің символдары болса, онда бір әріп басқасынмен ауыстырылады. Мысалы, *A* әріпін *D* әріпімен ағылшын алфавитін қолдана отырып ауыстыруға болады, ал *T*-ны – *Z*-пен ауыстырады. Егер символдар - (0-дан 9-ға дейін) сандар болса, мысалы, 3-ті 7-ге және 2-ні 6-ға ауыстыруға болады. Ауыстыру шифрлары екі категорияға бөліну мүмкін: *моноалфавитті (біралфавитті)* және *көпалфавитті* шифрлар.

Ауыстыру шифрлары үшін ашық мәтіннің (мәліметтер) алфавитінің ұзындығы (n_M) және шифрланған мәтіннің (деректердің) алфавитінің ұзындығы (n_C) бірдей, яғни $n_M = n_C$.

Бұл шифрларды қолдану кезінде алфавит әріптерін олардың реттік номерлерімен теңестіру ыңғайлы болады.

2.1.1. Ауыстырудың біралфавитті шифрлары

Ауыстырудың *моноалфавитті шифрларында* алғашқы мәтіндегі әріп (немесе символ) шифрланған мәтіндегі сол тек бір ғана әріпке (немесе символға) әрқашан да оның мәтіндегі

позициясына тәуелсіз өзгереді. Мысалы, егер алгоритм алғашқы мәтінде *A* әріпін *D* әріпіне өзгеретіндігін анықтаса, онда мұндай жағдайда әрбір *A* әріпі *D* әріпіне өзгереді. Басқаша айтқанда, алғашқы және шифрланған мәтіндегі әріптер бірден бірге қатынасында болады.

2.1 мысал. hello алғашқы мәтіні шифрлау нәтижесінде *KHOOR*-ға айналды. Шифрлау қандай шифрмен орындалғандығын анықтау керек. Алғашқы мәтінді көрсету үшін кіші символдарды, және шифрланған мәтінді алу үшін бас әріптерді (жоғарғы әріптер регистрін) қолданамыз.

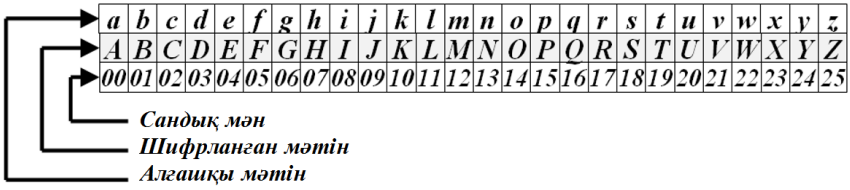
Шешім. l-дің екеуі де *O* ретінде шифрланғандықтан, шифр моноалфавитті болады.

2.2 мысал. hello алғашқы мәтіні шифрлау нәтижесінде *ABNZF*-қа айналды. Шифрлау қандай шифрмен орындалғандығын анықтау керек.

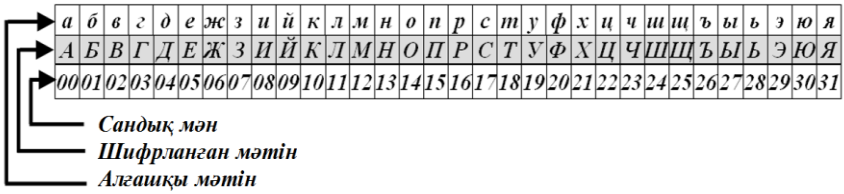
Шешім. Шифр моноалфавитті емес, себебі *l*-дің (эль) әрбір әріпі әртүрлі символдармен шифрланған. *l*-дің бірінші әріпі - *N,* ал екіншісі - *Z* ретінде шифрланған.

Ауыстырудың аддитивті шифрлары

Ең қарапайым моноалфавитті шифр - *аддитивті шифр,* оны кейде *жылжыту шифры*, ал кейде - *Цезарь шифры* деп те атайды, бірақ *аддитивті шифр* термині оның математикалық мағынасын көрсетеді. Алғашқы мәтін кіші әріптерден (*a*-дан *z*-ке дейін) және шифрланған мәтін бас әріптерден құралған (*A*-дан *Z*-ке дейін) деп болжайық. Математикалық операцияларды алғашқы және шифрланған мәтіндерге қолдануды қамтамасыз ету үшін әр әріпке сандық мәндер (төменгі және жоғарғы регистрі үшін) меншіктейміз, 2.1 суретінде көрсетілгендей ағылшын тілі (26 әріп) үшін және 2.2 суретінде көрсетілгендей орыс тілі үшін (32 әріп).



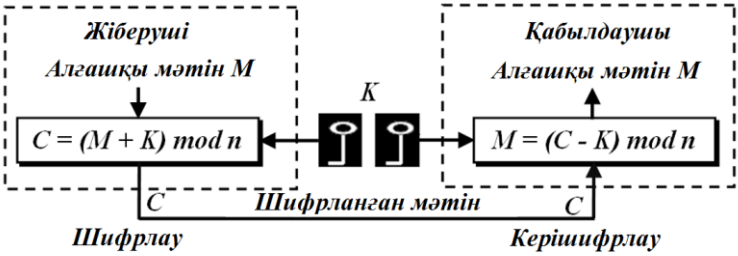
2.1 сурет - Ағылшын тілі үшін Z_{26}-да алғашқы және шифрланған мәтіндердің әріптерінің көрсетілуі



2.2 сурет - Орыс тілі үшін Z_{32}-де алғашқы және шифрланған мәтіндердің әріптерінің көрсетілуі

2.1 суретінде әрбір символға (төменгі регистр немесе жоғарғы регистр) Z_{26}-ден бүтін сан сәйкестендірілген. Жіберуші мен қабылдаушы арасындағы құпиялау кілті - Z_n-дегі бүтін сан. Шифрлау алгоритмі алғашқы мәтін символына кілтті қосады; керішифрлау алгоритмі шифрланған мәтін символынан кілтті алады. Барлық операциялар Z_n соңғы өрісінде жүргізіледі.

2.3 суреті шифрлау және керішифрлау үрдісін көрсетеді.



2.3 сурет - Ауыстырудың аддитивті шифры көмегімен хабарларды шифрлау және керішифрлау үрдістерін түсіндіру

Ауыстырудың аддитивті шифры көмегімен мәтіндік хабарларды шифрлау төмендегі өрнек көмегімен жүзеге асырылады:

$$C = (M + K) \bmod n, \qquad (2.1)$$

бұл жерде n – алфавит ұзындығы (ағылшын алфавиті үшін $n = 26$); M және C – сәйкесінше алғашқы және шифрланған мәтін; K – шифрлау (керішифрлау) кілті, ал керішифрлау

$$M = (C - K) \bmod n. \qquad (2.2)$$

Шифрлау және керішифрлау үрдістері бір-біріне инверсті екендігін оңай көрсетуге болады, себебі қабылдаушы құрған алғашқы мәтін (M_1) мен жіберуші жіберген мәтіні (M) бірдей:

$$M' = (C - K) \bmod n = (M + K - K) \bmod n = M.$$

2.3 мысал. hello хабарын $K = 15$ кілті бар ауыстырудың аддитивті шифрын қолданумен шифрлау.

Шешім. Алғашқы мәтінге әріптен әріпке шифрлау алгоритмін (2.1) қолданамыз:

$m1 = h \rightarrow 07;$ $c1 = (07 + 15) \bmod 26 = 22 \rightarrow W;$
$m2 = e \rightarrow 04;$ $c2 = (04 + 15) \bmod 26 = 19 \rightarrow T;$
$m3 = l \rightarrow 11;$ $c3 = (11 + 15) \bmod 26 = 0 \rightarrow A;$
$m4 = l \rightarrow 11;$ $c4 = (11 + 15) \bmod 26 = 0 \rightarrow A;$
$m5 = o \rightarrow 14;$ $c4 = (14 + 15) \bmod 26 = 3 \rightarrow D,$

және нәтижесінде шифрланған хабарды аламыз – WTAAD.

Назар аударыңыз, шифр моноалфавитті, себебі алғашқы мәтіннің бірдей әріпінің (*l*) екі бейнесі бірдей (*A*) символы ретінде шифрланған.

2.4 мысал. WTAAD хабарын $K = 15$ кілті бар аддитивті шифрды қолданумен керішифрлау.

Шешім. Шифрланған мәтінге әріптен әріпке керішифрлау алгоритмін (2.2) қолданамыз:

$c_1 = W \rightarrow 22;$ $m_1 = (22 - 15) \bmod 26 = 07 \rightarrow h;$
$c_2 = T \rightarrow 19;$ $m_2 = (19 - 15) \bmod 26 = 04 \rightarrow e;$
$c_3 = A \rightarrow 00;$ $m_3 = (00 - 15) \bmod 26 = 11 \rightarrow l;$
$c_4 = A \rightarrow 00;$ $m_4 = (00 - 15) \bmod 26 = 11 \rightarrow l;$
$c_5 = D \rightarrow 03;$ $m_5 = (03 - 15) \bmod 26 = 14 \rightarrow o,$

және нәтижесінде алғашқы хабарды аламыз – *hello*.

Назар аударыңыз, операциялар *26* модулі бойынша жүргізіледі, теріс нәтиже Z_{26} бейнелену қажет (мысалы, *-15* мәні 11 болады).

Тарихи аддитивті шифрлар *жылжыту шифрлары* деп аталды, себебі шифрлау алгоритмі "*әріпті төменге жылжыту пернесі*" ретінде түсіндірілді, ал керішифрлау алгоритмі "*әріпті жоғарыға жылжыту пернесі*" ретінде түсіндірілді. Мысалы, егер кілт $K = 15$

болса, шифрлау алгоритмі әріпті 15 әріпке төмен жылжытады (алфавит соңына). Керішифрлау алгоритмі әріпті 15 әріпке жоғары жылжытады (алфавит басына). Әрине, алфавиттің басы мен соңына жеткенде, сақина бойынша басына жылжу қажет (*n* модулі бойынша операцияның жарияланған қасиеттері).

Юлий Цезарь аддитивті шифрды өзінің шенеуніктерімен байланысу үшін қолданды. Сол себепті аддитивті шифрлар кейде *Цезарь шифры* деп аталады. Цезарь өз байланысы үшін кілт ретінде *3* (*K = 3*) санын қолданды.

Ауыстырудың аддитивті шифрлары тек шифрланған мәтінге кілттерді толық талдау кезде ғана шабуылға әлсіз болады (“*қатқыл күш*” шабуылы). Ауыстырудың аддитивті шифрының кілттер көптігі өте аз – олар тек *26* (ағылшын алфавиті үшін) және *32* (орыс алфавиті үшін). Кілттердің бірі, нөлдік, жарамсыз болып табылады (шифрланған мәтін алашқы мәтінге сәйкес болады). Демек, ағылшын алфавиті үшін тек *25* және орыс алфавиті үшін *31* мүмкін кілттер қалады. Сондықтан, зиян келтіруші шифрланған мәтінге “*қатқыл күш*” шабуылын оңай бастай алады.

2.5 мысал. Зиян келтіруші *UVACLYFZLJBYL* шифрленген мәтінін ұстап алды дейік. Ол шифрды “*қатқыл күш*” шабуылын қолданып қалай бұза алатынын көрсетейік.

Шешім. Зиян келтіруші мәтінді ашып көреді және кілттерді біріншісінен бастап іріктейді. 7 нөмерлі (*K = 7*) кілттің көмегімен ол "*not very secure*" (*өте қауіпсіз емес*) мәтінін алады.

Шифрланған мәтін: *UVACLYFZLJBYL*

K = 1	Алғашқы мәтін: *tubkxeykiaxk*
K = 2	Алғашқы мәтін: *styajwdxjhzwj*
K = 3	Алғашқы мәтін: *rsxzivewigyvi*
K = 4	Алғашқы мәтін: *qrwyhubvhfhuh*
K = 5	Алғашқы мәтін: *pqvxgtaugewtg*
K = 6	Алғашқы мәтін: *opuwfsztfdvst*
K = 7	Алғашқы мәтін: *notverysecure*

Ауыстырудың аддитивті шифрлары сонымен бірге статистикалық шабуылдың объектілері бола алады. Бұл әсіресе, егер қарсылас үлкен ұзындығы бар шифрланған мәтінді ұстап алған кезде нақты болады. Ол символдарды нақты тілде қолдану жиілігі туралы білімді пайдалана алады. 2.1 кесте ұзындығы *100* символ

ағылшын мәтіні үшін белгілі әріптердің пайда болу жиілігін көрсетеді [29, 32].

2.1 кесте

Ағылшын мәтінінде әріптердің пайда болу жиілігі

Әріп	*Жиілік*	*Әріп*	*Жиілік*	*Әріп*	*Жиілік*
E	*12,7*	*D*	*4,3*	*B*	*1,0*
T	*9,1*	*L*	*4,0*	*V*	*0,09*
A	*8,2*	*C*	*2,8*	*K*	*0,08*
O	*7,5*	*U*	*2,8*	*J*	*0,02*
I	*7,0*	*W*	*2,3*	*Q*	*0,01*
N	*6,7*	*F*	*2,2*	*X*	*0,01*
S	*6,3*	*G*	*2,0*	*Z*	*0,01*
H	*6,1*	*Y*	*1,9*		
R	*6,0*	*P*	*1,5*		

Дегенмен жалғыз символ жиілігі туралы ақпарат жеткіліксіз, және бұл әріптердің пайда болу талдауына негізделген шифрланған мәтінді талдауды қиындатады. Символдар қиыстыруларының жиілігін білу дұрыс болады, сонымен бірге екі немесе үш символдардың қиыстыруларының шифрланған мәтінде пайда болу жиілігін білу қажет және алғашқы құжат жазылған тілде оның жиілігімен салыстыру қажет.

Ағылшын мәтіні үшін екі символы бар (диаграмма (*diagrams*)) және үш символы бар (үшграмма (*trigrams*)) ең жиі қолданылатын топтар 2.2 кестеде көрсетілген.

2.2 кесте

Ағылшын тілінде пайда болу жиілігіне негізделген диаграммалар мен триграммалар топтары

Диграмма	*TH,HE,IN,ER,AN,RE,ED,ON,ES,ST,EN,AT,TO,NT,HA, ND,OU,EA,NG,AS,OR, TI,IS,ET,IT,AR,TE,SE,HI,OF*
Триграмма	*THE,ING,AND,HER,ERE,ENT,THA,NTH,WAS, ETH, FOR,DTH*

2.6 мысал. Зиян келтіруші төмендегі шифрланған мәтінді ұстап алды

XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRH
SPPEVWMXMWASVX-LQSVILY-
VVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW.

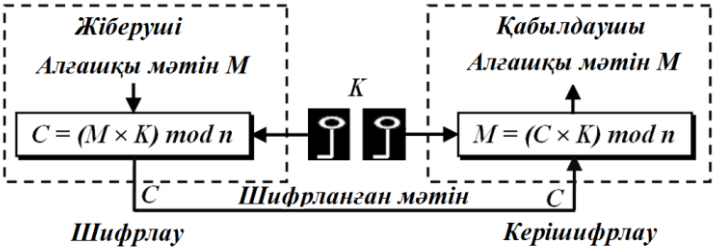
Статистикалық шабуылды қолданып алғашқы мәтінді табу қажет.

Шешім. Зиян келтіруші бұл шифрланған мәтіндегі әріптер жиілігінің кестесін құрған кезде, ол келесіні алады: *I = 14, V = 13, S = 12* және тағы басқалары. Ең жиі кездесетін символ - *I* , ол *14* рет кездеседі. Бұл шифрланған мәтіндегі *I* символы алғашқы мәтіндегі *e* символына сәйкес келуі ықтимал екендігін көрсетеді. Осылайша, кілт *K = 4* болады. Сондықтан, зиян келтіруші *K = 4* кілтін қолдана отырып, мәтінді керішифрлап төмендегіні алады:

the house is now for sale for four million dollars it is worth
more hurry before the seller receives more offers
(*енді үй төрт миллион долларға сатылады, сатушы көбірек*
ұсыныс алмай тұрғанда асығу қажет).

Ауыстырудың мультипликативті шифрлары

Ауыстырудың *мультипликативті шифрында* шифрлау алгоритмі алғашқы мәтінді кілтке көбейтуді қолданады, ал керішифрлау алгоритмі шифрланған мәтінді кілтке бөлуді қолданады, ол 2.4 суретінде көрсетілген.



2.4 сурет - Ауыстырудың мультипликативті шифры көмегімен хабарларды шифрлау және керішифрлау

Ауыстырудың мультипликативті шифрында операциялар Z_{26} соңғы өрісінде жүргізілетіндіктен, керішифрлау мұнда кілттің

мультипликативті инверсиясына көбейтуін білдіреді. Кілт Z^*_n жиынына жату керек екеніне назар аударыңыз, бұл шифрлау және керішифрлау алгоритмдері бір біріне инверсті болатынын кепілдейді.

Ауыстырудың мультипликативті шифры көмегімен мәтіндік хабарларды шифрлау келесі өрнек көмегімен жүзеге асырылады:

$$C = (M \times K) \bmod n, \qquad (2.3)$$

ал керішифрлау:

$$M = (C \times K^{-1}) \bmod n, \qquad (2.4)$$

мұндағы K^{-1} – K кілтінің мультипликативті инверсиясы.

2.7 мысал. Ағылшын алфавитін қолдану кезінде ауыстырудың мультипликативті шифры үшін кілттер жиыны қандай екенін анықтау керек.

Шешім. Кілт Z^*_{26} соңғы өрісінде болу керек. Бұл жиында тек *12* элемент қана бар: *1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23* және *25*. Бұл *2, 4, 6, 8, 10, 12, 13, 14, 16, 18, 20, 22* және *24* мәндері үшін Z^*_{26} жиынының мультипликативті инверсиясы жоқ екендігімен түсіндіріледі.

2.8 мысал. Мультипликативті шифрды қолданып $K = 7$ кілтімен *hello* хабарын шифрлау.

Шешім. Шифрланған мәтінге әріптен әріпке шифрлау алгоритмін (2.3) қолдып:

$m_1 = h \rightarrow 07;$ $\quad c_1 = (07 \times 07) \bmod 26 = 23 \rightarrow X;$
$m_2 = e \rightarrow 04;$ $\quad c_2 = (04 \times 07) \bmod 26 = 02 \rightarrow C;$
$m_3 = l \rightarrow 11;$ $\quad c_3 = (11 \times 07) \bmod 26 = 25 \rightarrow Z;$
$m_4 = l \rightarrow 11;$ $\quad c_4 = (11 \times 07) \bmod 26 = 25 \rightarrow Z;$
$m_5 = o \rightarrow 14;$ $\quad c_5 = (14 \times 07) \bmod 26 = 20 \rightarrow U,$

XCZZU – шифрланған хабарды аламыз.

Аффинды шифр деп аталатынды алу үшін ауыстырудың аддитивті және мультипликативті шифрларын араластыруға болады. *Аффинды шифр* - кілттер жұбымен екі шифрдың қиыстыруы. Бірінші кілт мультипликативті шифрмен қолданылады, ал екінші – аддитивті шифрмен. 2.5 сурет аффинды

шифр – шын мәнінде бірінен кейін бірі қолданылатын екі шифр екенін көрсетеді.

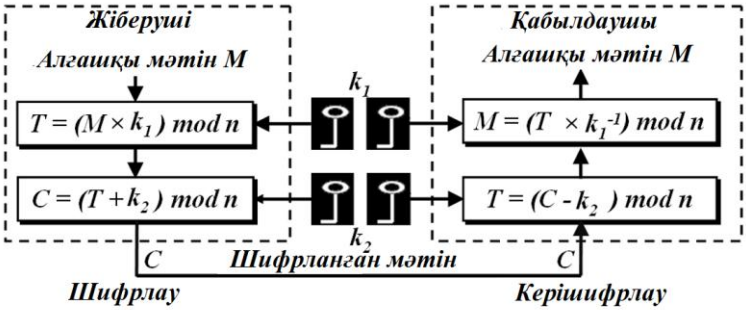
Ауыстырудың аффинды шифрын қолдану кезінде алғашқы M және шифрланған C мәтіні арасындағы, сонымен бірге k_1 және k_2 кілттермен қатынас төмендегідей анықталады:

$$C = (M \times k_1 + k_2) \bmod n, \qquad (2.5)$$

және

$$M = ((C + (-k_2)) \times k_1^{-1}) \bmod 26, \qquad (2.6)$$

бұл жерде k_1^{-1} – k_1 мультипликативті инверсиясы, ал $-k_2$ – k_2 аддитивті инверсиясы.



2.5 сурет - Ауыстырудың аффинды шифры көмегімен хабарларды шифрлау және керішифрлау

(2.5)-дегі сияқты 2.5 суреттегі шифрлау немесе керішифрлау үшін тек бір ғана кешенді операцияны көрсетуге болады. Дегенмен 2.6 суретінде аралық нәтиже (T) қолданылады және шифлар қиыстыруын қолданылатын барлық кезде, олардың әрқайсысының сызықтың басқа жағына инверсиясы бар екендігіне және олар шифрлау мен керішифрлау кезінде кері тәртіпте қолданылатындығына көз жеткізу керек екендігін көрсететін екі жеке операция көрсетілген. Егер қосу шифрлау кезіндегі соңғы әрекет болса, онда азайту керішифрлау кезіндегі бірінші әрекет болады.

2.9 мысал. Ауыстырудың аффинды шифры бірінші кілт Z^*_{26} – дан, ал екіншісі - Z_{26}-дан тұратын кілттер жұбын қолданады.

Ағылшын алфавитін қолданған кезде ауыстырудың аффинды шифры үшін кілттер жиыны қандай екенін анықтау керек.

Шешім. 2.7 мысалында ағылшын алфавиті үшін k_1^{-1} кілттер көптігі *12*-ге тең деп анықталған. Сондықтан аффинды шифр үшін кілттер аймағы келесіге тең болады

$$K = Z_{26} \times Z^*_{26} = 25 \times 12 = 300.$$

2.10 мысал. Ауыстырудың аффинды шифрын қолданып $k_1 = 7$ және $k_2 = 2$ кілттер жұбымен *hello* хабарын шифрлау.

Шешім. Мультипликативті кілт үшін $k_1 = 7$ және аддитивті кілт үшін $k_2 = 2$ қолданып, сонымен бірге хабарға әріптен әріпке ауыстырудың шифрлау алгоритмін (2.5) қолдана отырып:

$m_1 = h \rightarrow 7;$ $\quad c_1 = (7 \times 7 + 2) \bmod 26 = 25 \rightarrow Z;$
$m_2 = e \rightarrow 4;$ $\quad c_2 = (4 \times 7 + 2) \bmod 26 = 4 \rightarrow E;$
$m_3 = l \rightarrow 11;$ $\quad c_3 = (11 \times 7 + 2) \bmod 26 = 1 \rightarrow B;$
$m_4 = l \rightarrow 11;$ $\quad c_4 = (11 \times 7 + 2) \bmod 26 = 1 \rightarrow B;$
$m_5 = o \rightarrow 14;$ $\quad c_5 = (14 \times 7 + 2) \bmod 26 = 22 \rightarrow W,$

ZEBBW– шифрланған хабарын аламыз.

2.11 мысал. Ауыстырудың аффинды шифрын қолданып $k_1 = 7$ және $k_2 = 2$ кілттер жұбымен *ZEBBW* хабарын керішифрлау.

Шешім. Алғашқы мәтін символын табу үшін алынған шифрланған мәтінге $(-k_2) \bmod 26 = (-2) \bmod 26 = 24$–дан аддитивті инверсияны қосамыз. Сосын нәтижені $k_1^{-1} \bmod 26 = 7^{-1} \bmod 26 = 15$ –дан мультипликативті инверсияға көбейтеміз. k_2-нің Z_{26} –да аддитивті инверсиясы және k_1-дің Z^*_{26} мультипликативті инверсиясы болғандықтан, алғашқы мәтін 2.10 мысалда қолданылғандай:

$c_1 = Z \rightarrow 25;$ $\quad m_1 = ((25 + 24) \times 15) \bmod 26 = 7 \rightarrow h;$
$c_2 = E \rightarrow 4;$ $\quad m_2 = ((4 + 24) \times 15) \bmod 26 = 4 \rightarrow e;$
$c_3 = B \rightarrow 1;$ $\quad m_3 = ((1 + 24) \times 15) \bmod 26 = 11 \rightarrow l;$
$c_4 = B \rightarrow 1;$ $\quad m_4 = ((1 + 24) \times 15) \bmod 26 = 11 \rightarrow l;$
$c_5 = W \rightarrow 22;$ $\quad m_5 = ((22 + 24) \times 15) \bmod 26 = 14 \rightarrow o.$

Демек, *hello* алғашқы хабарын аламыз.

Ауыстырудың аддитивті шифры – $k_1 = 1$ болған кездегі ауыстырудың аффинды шифрының жеке жағдайы. Ауыстырудың

мультипликативті шифры – $k_1 = 0$ болған кездегі ауыстырудың аффинды шифрының жеке жағдайы.

"*Қатқыл күш*" және статистикалық шабуыл әдістері тек шифрланған мәтінге қолданылатындықтан, таңдалған алғашқы мәтінге шабуыл жасап көреміз. Зиян келтіруші ағылшын тіліндегі төмендегі шифрланған мәтінді ұстап алды дейік:

PWUFFOGWCHFDWIWEJOUUNJORSMDWRHVCMWJUPVCCG.

Зиян келтіруші сонымен бірге хабар жіберуші компьютерге қатынас құру мүмкіндігін алады және *et* екі символынан тұратын алғашқы мәтінді басуға уақыты болады. Сол кезде ол екі түрлі алгоритмді қолданып қысқа алғашқы мәтінді шифрлап көреді, себебі олардың қайсысы ауыстырудың аффинды шифры екендігі екенін нақты білмейді. Нәтижесінде ол деректердің бірінші жиынын "*алғашқы мәтін/шифрланған мәтін*" – $et \rightarrow WC$ және деректердің екінші жиынын $et \rightarrow WF$ алады.

Кілтті табу үшін, зиян келтіруші төмендегі стратегияны қолданады:

1. Зиян келтіруші егер бірінші алгоритм ауыстырудың аффинды шифры болса, онда ол деректердің бірінші жиынына негізделген төмендегі теңдеулерді құра алатындығын біледі:

$m_1 = e \rightarrow 4;$ $\quad c_1 = (4 \times k_1 + k_2) \bmod 26 = 22 \rightarrow W;$

$m_2 = t \rightarrow 19;$ $\quad c_2 = (19 \times k_1 + k_2) \bmod 26 = 2 \rightarrow C.$

Салыстырудың бұл екі теңдеулерді шешуге болады (k_1 және k_2 мәндері табылады)

$$\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ 19 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 22 \\ 2 \end{pmatrix} \bmod 26 = \begin{pmatrix} 19 & 7 \\ 3 & 24 \end{pmatrix} \cdot \begin{pmatrix} 22 \\ 2 \end{pmatrix} \bmod 26 = \begin{pmatrix} 16 \\ 10 \end{pmatrix}.$$

Бұдан $k_1 = 16$, ал $k_2 = 10$ аламыз. Дегенмен бұл жауап тиімсіз, себебі $k_1 = 16$ кілттің бірінші бөлігі бола алмайды. Оның мәні 16, Z^*_{26}-да мультипликативті инверсиясы болмайды.

2. Зиян келтіруші енді деректердің екінші жиынының нәтижесін қолданып көреді:

$m_1 = e \rightarrow 4;$ $\quad c_1 = (4 \times k_1 + k_2) \bmod 26 = 22 \rightarrow W;$
$m_2 = t \rightarrow 19;$ $\quad c_2 = (19 \times k_1 + k_2) \bmod 26 = 5 \rightarrow F.$

Төртбұрышты матрица және оның инверсиясы – алдыңғы мысалдағыдай:

$$\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ 19 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 22 \\ 5 \end{pmatrix} mod\ 26 = \begin{pmatrix} 19 & 7 \\ 3 & 24 \end{pmatrix} \cdot \begin{pmatrix} 22 \\ 5 \end{pmatrix} mod\ 26 = \begin{pmatrix} 11 \\ 4 \end{pmatrix}.$$

Енді зиян келтіруші $k_1 = 11$ және $k_2 = 4$ алады, бұл жұп тиімді, себебі k_1-дің Z^*_{26} –да мультипликативті инверсиясы бар. Ол (*19*, *22*) кілттер жұбын пайдаланып көреді, олар (*11*, *4*) жұбының инверсиясы болып табылады:

$c_1 = W \rightarrow 22;$ $\quad m_1 = ((22 + 22) \times 19) \bmod 26 = 04 \rightarrow e,$
$c_2 = F \rightarrow 5;$ $\quad m_2 = ((5 + 22) \times 19) \bmod 26 = 19 \rightarrow t$

және хабарды керішифрлайды:

PWUFFOGWCHFDWIWEJOUUNJORSMDWRHVCMWJUPVCCG.

Бұл кезде алғашқы мәтін болады:

Best time of the year is spring when flower bloom
(*Ең жақсы жыл уақыты - көктем, гүлдер гүлдеген кезде*).

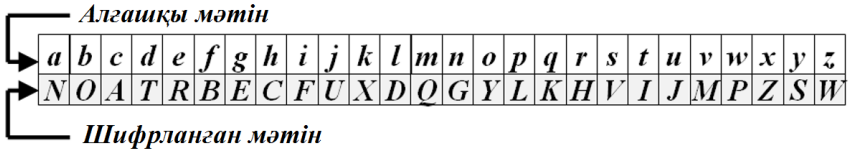
Ауыстырудың бір алфавитті шифры

Ауыстырудың аддитивті, мультипликативті және аффинды шифрларында кілттер көптігі аз болғандықтан, олар "*қатқыл күш*" шабуылына өте әлсіз. Жіберуші мен қабылдаушы алғашқы мәтіндегі әр әріпті шифрлау немесе шифрланған мәтіндегі әр әріпті керішифрлау үшін қолданатын жалғыз кілтті тағайындайды. Басқаша айтқанда, кілт жіберілетін әріптерден тәуелсіз.

Ең жақсы шешім шифрланған мәтіннің сәйкес символына алғашқы мәтіннің әрбір әрпінің бейнесін құру болып табылады. Жіберуші мен қабылдаушы әр әріптің бейнесі туралы келісіп және

оларды кесте түрінде жазып ала алады. 2.6 суреті осындай бейнеленудің мысалын көрсетеді.

2.13 мысал. 2.6 суретте көрсетілген кілтті пайдаланып хабарды шифрлау керек



2.6 сурет - Ағылшын алфавитының ауыстыру бір алфавитті шифры үшін кілт мысалы

This message is easy to encrypt but hard to find the key
(*Бұл хабарды шифрлау қарапайым, бірақ шифрланған мәтінге қолданған кілтті табу қыйын*).

Шешім. Шифрланған хабар түрі:

ICFVQRVVNEFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS.

Ауыстырудың бір алфавитті шифры үшін кілттік кеңістік өлшемі - 26-ның орын ауыстыру саны, яғни *26!* (шамамен $4 \cdot 10^{26}$). Бұл "*қатқал күш*" шабуылын зиян келтіруші үшін, тіпті ол қуатты компьютер қолданса да, өте күрделі етеді. Дегенмен ол символдардың пайда болу жиілігіне негізделген статистикалық шабуыл әдісін қолдана алады, себебі бұл шифр символдардың пайда болу жиілігін өзгертпейді.

2.1.2. Ауыстырудың көпалфавитті шифрлары

Ауыстырудың көпалфавитті шифрын қолдану символдың әр пайда болуының әртүрлі ауыстыруы болуына алып келеді. Алғашқы мәтіндегі символ мен шифрланған мәтіндегі символ арасындағы қатынас – *«бірден көпке».* Мысалы, *a* әріпі мәтін басында *D* ретінде шифрланса, бірақ мәтін ортасында – *N* ретінде шифрлану мүмкін. Ауыстырудың көпалфавитті шифрларының артықшылығы бар: олар негізгі тіл символдарының пайда болу жиілігін жасырады. Зиян

келтіруші шифрланған мәтінді бұзу үшін жеке символдардың статистикалық жиілігін қолдана алмайды.

Ауыстырудың көпалфавитті шифрын құру үшін шифрланған мәтіннің әрбір символын алғашқы мәтіннің сәйкес символына және хабарда алғашқы мәтін символының позициясына тәуелді болуын жасау қажет. Бұл кілт ішкі кілттер ағыны болу керектігін жобалады, әр ішкі кілт шифрлаудың ішкі кілттерін таңдау үшін қолданылатын алғашқы мәтін символының позициясына тәуелді болуы керек. Басқаша айтқанда, k_i алғашқы мәтінде *i*-ші символды шифрлауға және шифрланған мәтінде *i*-ші символды құруға қолданылатын $k = (k_1, k_2, k_{3,}...)$ кілттік ағыны болуы қажет.

Ауыстырудың автокілттік шифры

Кілттің позициясынан тәуелділігін түсіну үшін автокілттік деп атылатын қарапайым көпалфавитті шифрды қарастырайық. Бұл шифрда кілт – ішкі кілттер ағыны, оның ішінде әр кілт алғашқы мәтінде сәйкес символды шифрлау үшін қолданылады. Бірінші ішкі кілт – жіберуші мен қабылдаушының жасырын келісілген алдын ала анықталған мән. Екінші ішкі кілт – алғашқы мәтіннің бірінші символының мәні (0 мен 25 арасындағы). Үшіншісі – екінші алғашқы мәтіннің *i*-ші мәні. Ары қарай осылайша. Алғашқы мәтін: $M = m_1, m_2, m_3, ...$ болсын дейік. Шифрланған мәтін: $C = c_1, c_2, c_3, ...$ Кілт: $K = (k_1, k_2 = m_1, k_3 = m_2, k_4 = m_3, ...)$.

Ауыстырудың автокілттік шифры көмегімен мәтіндік хабарларды шифрлау келесі өрнек көмегімен жүзеге асырылады:

$$c_i = (m_i + k_i) \bmod n, \qquad (2.7)$$

ал керішифрлау:

$$m_i = (c_i - k_i) \bmod n. \qquad (2.8)$$

Ауыстырудың шифрының *автокілттік* деп аталуы ішкі кілттер шифрлау үрдісі кезінде алғашқы мәтіннің шифр символына тәуелді автоматты түрде құрылатындығымен түсіндіріледі.

2.14 мысал. Жіберуші мен қабылдаушы $k_1 = 12$ алғашқы кілттік мәні бар автокілттік шифрды қолдануға келісті дейік. Енді жіберуші

қабылдаушыға *Attack is today* (*шабуыл бүгін*) хабарламасын жібергісі келеді.

Шешім. Шифрлау символдан символға жүргізіледі. Алғашқы мәтіндегі әр символ алдымен 2.1 суретінде көрсетілгендей, бүтін санның мәнімен ауыстырылады, шифрланған мәтіннің бірінші символын құру үшін бірінші ішкі кілт қосылады. Кілттің қалған бөлігі алғашқы мәтін символдарының оқу шамасы бойынша құрылады. Шифрлау нәтижесі 2.3 кестесінде келтірілген.

Мынаған назар аударыңыз, шифр көпалфавитті болып табылады, өйткені алғашқы мәтінде *a*-ның үш рет пайда болуы әртүрлі шифрланған. *t*-ның да үш рет пайда болуы әртүрлі шифрланған.

Ауыстырудың автокілттік шифры шынымен жеке символдың пайда болу жиілігі статистикасын жасырады. Дегенмен ол да ауыстырудың аддитивті шифры сияқты «қатқыл күш» шабуылына әлсіз. Бірінші ішкі кілт *25* мәннің тек біреуі бола алады (*1 – 25*).

2.3 кесте

2.14 мысалы үшін шифрлау нәтижесі

Алғашқы мәтін	*a*	*t*	*t*	*a*	*c*	*k*	*i*	*s*	*t*	*o*	*d*	*a*	*y*
М мәні	*00*	*19*	*19*	*00*	*02*	*10*	*08*	*18*	*19*	*14*	*03*	*00*	*24*
К кілттер ағыны	*12*	*00*	*19*	*19*	*00*	*02*	*10*	*08*	*18*	*19*	*14*	*03*	*00*
С мәні	*12*	*19*	*12*	*19*	*02*	*12*	*18*	*00*	*11*	*07*	*17*	*03*	*24*
Шифрланған мәтін	*M*	*T*	*M*	*T*	*C*	*M*	*S*	*A*	*L*	*H*	*R*	*D*	*Y*

Ауыстырудың көпалфавитті шифрларында қажеттілік тілдің сипаттамаларын жасырып қана қоймайды, оларда үлкен кілттер көптігі болады.

Плейфердің ауыстыру шифры

Ауыстырудың көпалфавитті шифрының басқа мысалы – бірінші дүние жүзілік соғыс кезінде британ әскері қолданған *Плейфер шифры*. Бұл шифрда құпиялау кілті *5 × 5* матрицасына орналасқан алфавиттің *25* әрпінен жасалған (латын алфавиті үшін, кирилица үшін – *6 × 6* матрицасында).

Матрицаны құру мен шифрды қолдану үшін кілттік сөзбен төрт қарапайым ережені есте сақтау қажет. Кілттік матрицаны құру үшін

ең алдымен, матрицаның бос ұяшықтарын кілттік сөздің әріптерімен толтыру керек (қайталанған символдарды жазбай). Сосын кілттік сөзде кездеспейтін қалған алфавит символдармен тәртіп бойынша матрицаның бос ұяшықтарын толтыру керек (ағылшын мәтінінде әдетте *q* символы алынып тасталады, басқа нұсқаларында *i* мен *j* бір ұяшыққа бірігеді). Кілттік сөз матрицаның жоғарғы жолына сол жақтан оң жақа қарай немесе сол жақтағы жоғары бұрыштан ортаға қарай спираль ретінде жазылу мүмкін. Алфавиттің символдарымен толықтырылған кілттік сөз 5×5 матрицасын құрады және шифрдың кілті болып саналады. Матрицада әріптерді орналастыру туралы түрлі келісімдер көмегімен құпиялаудың көптеген түрлі кілттерін құруға болады.

Хабарды шифрлау үшін оны биграммаға (екі символдар тұратын топ) бөлу қажет, мысалы *hello world* хабары *he ll ow or ld* болады, және бұл биграммаларды кестеден табу қажет. Биграмманың екі символы кілттік матрицада тікбұрыштың бұрыштарына сәйкес келеді. Бұл тікбұрыштың бұрыштарының бір біріне қатысты орнын анықтайық. Содан кейін, төмендегі ережелерге сүйене отырып, кіріс мәтінінің символдар жұбын шифрлаймыз:

1. Егер биграмманың екі символы сәйкес келсе, бірінші символдан кейін *x* жалған символын қосамыз, символдардың жаңа жұбын шифрлаймыз және жалғастырамыз. *Плейфер* шифрының кейбір нұсқаларында *x* символының орнына *q* немесе _ символы қолданылады.

2. Егер кіріс мәтіннің биграммасының символдары бір жолда кездессе, онда бұл символдар сәйкес символдың сол жағында жақын бағанда орналасқан символдармен ауыстырылады. Егер символ жолдың соңында болса, онда ол осы жолдың бірінші символымен ауыстырылады.

3. Егер кіріс мәтіннің биграммасының символдары бір бағанда кездессе, онда олар тікелей солардың астында орналасқан сол бағанның символына айналады. Егер символ бағанның төменгі жағында орналасса, онда ол осы бағанның бірінші символына алмастырылады.

4. Егер кіріс мәтіннің биграммасының символдары әртүрлі бағандар мен жолдарда орналасқан болса, онда олар сол жолдарда орналасады, бірақ кілттік матрицада тікбұрыштың басқа бұрыштарына сәйкес келеді.

Шифрланған хабарды керішифрлау үшін *x* (немесе *q*) символын алып тастап, егер олардың кіріс хабарда мағынасы болмаса, осы төрт ереженің инверсиясын қолдану қажет.

Плейфер шифры көпалфавитті шифрларға арналған критерийлерге сәйкес келеді. *Кілт* - ішкі кілттер ағыны, оның ішінде олар бір уақытта екеуден құрылады. *Плейфер* шифрында кілттер ағыны мен шифрлар ағыны бірдей. Бұл дегеніміз жоғарыда аталған ережелерді кілттер ағынын құру ережелері ретінде қабылдауға болатындығын білдіреді. Шифрлау алгоритмі алғашқы мәтіннен символдар жұбын алады және жоғарыда көрсетілген ережелерге сүйене отырып, ішкі кілттер жұбын құрады. Кілттер ағыны алғашқы мәтіндегі символдардың позициясынан тәуелді деп айтуға болады. Позицияға тәуелділікті бұл жерде әртүрлі түсінуге болады: алғашқы мәтіннің әр символына арналған ішкі кілт келесі немесе алдыңғыға тәуелді болады. *Плейфер* шифрын осылай қарастыра отырып, шифрланған мәтін – бұл кілттердің нақты ағыны деп айтуымызға болады.

Алғашқы мәтін: $M = m_1, m_2, m_3, \ldots$болсын дейік. Шифрланған мәтін: $C = c_1, c_2, c_3, \ldots$. Кілт: $K = [(k_1,k_2), (k_3,k_4), \ldots]$.

Плейфер шифры көмегімен мәтіндік хабарды шифрлау келесі өрнек көмегімен жүзеге асырылады:

$$c_i = k_i, \tag{2.9}$$

ал керішифрлау:

$$m_i = k_i. \tag{2.10}$$

2.15 мысал. Плейфер шифры көмегімен *hello world* (*Салем әлем*) алғашқы мәтінін *playfair example* кілттік сөзімен шифрлау. Бұл жағдайда құпиялау матрицасы (немесе құпия кілт) 2.7 суретте көрсетілгендей болады.

Шешім. Әріптерді жұп-жұбымен топтастырған кезде, аламыз: *he ly lo wo rl d*. Соңғы символды жұппен қамтамасыз ету үшін *x* символын *y* символынан кейін қосу керек, одан кейін аламыз: *he ly lo wo rl dy*. Кіріс мәтінін кесте (2.7 сурет) көмегімен түрлендіріп аламыз:

1. *he* биграммасы тікбұрышты құрады, оны *DM*-ға ауыстырамыз.

2. *ly* биграммасы тікбұрышты құрады, оны *AF*-қа ауыстырамыз.

3. *lo* биграммасы бір бағанда орналасқан, оны *YK*-ке ауыстырамыз.

4. *wo* биграммасы бір бағанда орналасқан, оны *WY*-қа ауыстырамыз.

5. *rl* биграммасы тікбұрышты құрады, оны *CR*-ге ауыстырамыз.

6. *dy* биграммасы бір бағанда орналасқан, оны *GA*-ға ауыстырамыз.

Құпия кілт =

P	*L*	*A*	*Y*	*F*
I	*R*	*E*	*X*	*M*
B	*C*	*D*	*G*	*H*
I/J	*K*	*N*	*O*	*S*
T	*U*	*V*	*W*	*Z*

2.7 сурет - *Плейфер* шифрының құпия кілтіне мысал

Сонымен, шифрланған мәтін келесі түрде көрсетіледі:

DM AF YK WY CR GA.

Бұл мысал *Плейфер* шифры – шын мәнінде көпалфавитті екенін көрсетеді: алғашқы мәтіннің *l* (эль) символының үш рет кездесуі сәйкесінше *A, Y* және *R* ретінде шифрланған; алғашқы мәтіннің *o* символының екі рет кездесуі сәйкесінше *K* және *Y* ретінде шифрланған.

Кирилицаны қолдану үшін матрица өлшемін *6×6*-ға дейін көбейту керек. Алфавиттің 32 әріпі мен қосымша төрт символ қолданылады: . (нүкте); , (үтір); – (тире); _ (астыңғы сызық). Ресей алфавиті (кілттік сөзді қолданбай) үшін құпиялау матрицасына мысал 2.8 суретінде көрсетілген.

А	*Х*	*Б*	*М*	*Ц*	*В*
Ч	*Г*	*Н*	*Ш*	*Д*	*О*
Е	*Щ*	,	*Ж*	*У*	*П*
.	*З*	*Ъ*	*Р*	*И*	*Й*
С	*Ь*	*К*	*Э*	*Т*	*Л*
Ю	*Я*	_	*Ы*	*Ф*	–

2.8 сурет - Орыс алфавиті үшін құпиялау матрицасының мысалы

(астыңғы сызық) символы биграммада бірдей символдар пайда болған кезде немесе соңғы символды жұппен қамтамасыз ету үшін қолданылады.

Байқағанымыздай, *Плейфер* шифрының «қатқыл күш» шабуылы өте қиын. Домен өлшемі – *n!* (*n* факториал). Бұдан басқа, шифрланған мәтін жеке әріптердің пайда болу жиілігін жасырады.

Дегенмен екі әріптік киыстыру (биграмма) жиілігі толықтырғышты енгізу салдарынан кейбір дәрежеде сақталады, сол себепті криптографиялық талдаушы кілтті табу үшін шабуылды тек қана биграмма жиілігінің сынағына негізделген шифрланған мәтінге ғана қолдана алады.

Виженердың ауыстыру шифры

Ауыстырудың көпалфавитті шифрының тағы бір қызықты түрі он алтыншы жүз жылдықтың француз математигі *Блез де Виженер* құрған *Виженер шифры* болып табылады [5, 32]. *Виженер* шифры кілттер ағынын құрудың түрлі стратегиясын қолданады. *Кілттер ағыны* – *m* ұзындықты құпиялау кілтінің бастапқы ағынының қайталануы. Шифр келесі түрде сипатталуы мүмкін: $(k_1, k_2, \ldots, k_n)$ – жіберуші және қабылдаушымен келісілген құпиялаудың бастапқы кілті.

Алғашқы мәтін: $M = m_1, m_2, m_3, \ldots$болсын дейік. Шифрланған мәтін: $C = c_1, c_2, c_3, \ldots$. Кілттер ағыны: $K = k_1, k_2, k_3, \ldots$.

Виженер шифры көмегімен мәтіндік хабарды шифрлау төмендегі өрнек көмегімен жүзеге асырылады:

$$c_i = (m_i + k_i) \bmod n, \qquad (2.11)$$

ал керішифрлау:

$$m_i = (c_i - k_i) \bmod n. \qquad (2.12)$$

Виженер ауыстыру шифры мен қарастырылған ауыстырудың басқа екі көпалфавитті шифрлары арасындағы тағы бір маңызды айырмашылық: Виженер ауыстыру шифрының кілттер ағыны алғашқы мәтін символдарына тәуелсіз; ол тек алғашқы мәтіндегі символдың позициясына ғана тәуелді. Басқаша айтқанда, кілттер ағыны алғашқы мәтін мағынасын білмей ақ құрылады.

2.16 мысал. *PASCAL* кілттік сөзінің 6 символын қолдана отырып *She is listening* (*Ол тыңдап тұр*) хабарды шифрлау.

Шешім. Кілттердің бастапқы ағыны – бұл *(15, 0, 18, 2, 0, 11)*. Кілттер ағыны – кілттердің осы бастапқы ағынының қайталануы (қанша қажет болса, сонша рет). Шифрлау үрдісі 2.4 кестесінде көрсетілген.

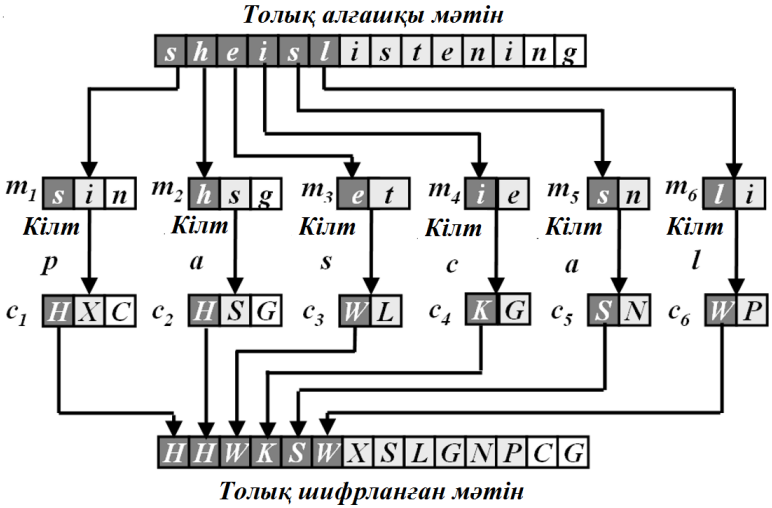
2.4-кесте

2.16 мысал үшін шифрлау үрдісі

Алғашқы мәтін	*s*	*h*	*e*	*i*	*s*	*l*	*i*	*s*	*t*	*e*	*N*	*i*	*n*	*g*
М мәні	*18*	*07*	*04*	*08*	*18*	*11*	*08*	*18*	*19*	*04*	*13*	*08*	*13*	*06*
Кілттер ағыны	*15*	*00*	*18*	*02*	*00*	*11*	*15*	*00*	*18*	*02*	*00*	*11*	*15*	*00*
С мәні	*07*	*07*	*22*	*10*	*18*	*22*	*23*	*18*	*11*	*6*	*13*	*19*	*02*	*06*
Шифрланған мәтін	*H*	*H*	*W*	*K*	*S*	*W*	*X*	*S*	*L*	*G*	*N*	*T*	*C*	*G*

Осылайша, шифрланған мәтін келесі түрде көрсетіледі: *HHWKSWXSLGNTCG*.

Виженер ауыстыру шифры аддитивті шифрлар қиыстыруы ретінде қарастырылады. 2.9 суретінде алдыңғы мысалдың алғашқы мәтінің әрқайсысында алты элементі бар (біреуінде алғашқы мәтіннің бір әріпі жетпесе де), ал әр элемент жеке шифрланған бірнеше бөліктерден тұратындай қарастырылғандығын көрсетеді.



2.9 сурет - *Виженер* ауыстыру шифры аддитивті шифрлар қиыстыруы ретінде

Сурет кейінірек *Виженер* ауыстыру шифрының криптографиялық талдауын түсінуге көмектеседі. Алғашқы мәтіннің *n* бөліктері бар, олардың әрқайсысы шифрланған мәтінді *m* бөліктерге бөлу үшін түрлі кілттермен шифрланған.

Аддитивті шифр – *m = 1* болғандағы Виженер ауыстыру шифрының жеке жағдайы.

Виженер ауыстыруын қарастырудың басқа тәсілі – 2.5 кестеде көрсетілген *Виженер* (*Vigenere tableau*) тізімі немесе кестесі көмегімен.

2.5 кесте

Виженер тізімі

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
A	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
B	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*
C	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*
D	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*
E	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*
F	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*
G	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*
H	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*
I	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
J	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
K	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
L	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*
M	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
N	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
O	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*
P	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*
Q	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*
R	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*
S	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*
T	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*
U	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*
V	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
W	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*
X	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*
Y	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*
Z	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*

Бірінші жол шифрланатын алғашқы мәтіннің символдарын көрсетеді. Бірінші баған кілтпен қолданылатын символдар бағанын құрады. Кестенің басқа бөлігі шифрланған мәтін символдарын көрсетеді. *PASCAL* сөзін кілт ретінде қолдана отырып, *she listening* алғашқы мәтіні үшін шифрланған мәтінді табу үшін алғашқы мәтіннің *s* символын бірінші жолда табуға болады, кілттік сөздің *P* символын бірінші бағанда және жолдар мен бағандар қиылысында – шифрланған мәтіннің *H* символы болады. Бірінші жолда *h* және бірінші бағанда *A* табамыз, жолдар мен бағандар қиылысында – шифрланған мәтіннен *H* символын табамыз. Шифрланған мәтіннің барлық символдары табылмайынша, осы әрекеттерді қайталаймыз.

PASCAL сөзін кілт ретінде қолданып, *HHWKSWXSLGNTCG* шифрланған мәтініне алғашқы мәтінді табу үшін кілттік сөзден бірінші бағанда *P* символын табу керек; содан кейін, жол бойынша оңға жылжып, шифрланған мәтіннің *H* символын табамыз; *H* символы орналасқан баған мен бірінші жолдың қиылысында – алғашқы мәтіннің *s* символын табамыз. Бірінші бағанда кілттік сөздің *A* символын; содан кейін, жол бойынша оңға жылжып шифрланған мәтіннің *H* символын табамыз; *H* символы орналасқан баған мен бірінші жол қиылысында – алғашқы мәтіннің *h* символын табамыз. Алғашқы мәтіннің барлық символдары табылмайынша, осы әрекеттерді қайталаймыз.

Виженер ауыстыру шифры барлық көпалфавитті шифрлар сияқты символдардың пайда болу жиілігін сақтамайды. Дегенмен зиян келтіруші ұсталған шифрланған мәтінді керішифрлау үшін кейбір әдістерді қолдануы мүмкін. Осы жағдайда криптографиялық талдау екі бөлімнен тұрады: кілттің ұзындығын табады және сосын тікелей кілтті табады.

1. Кілттің ұзындығын табу үшін бірнеше әдістер табылды. Бір әдісті төменде қарастырамыз. *Касиский (Kasiski) тесті* деп аталатын әдісте криптографиялық талдаушы шифрланған мәтінде үш символдан тұратын қайталанатын сегменттерді іздейді [20, 32]. Екі сегмент табылды деп ойлап, олардың арасындағы ара-қашықтық – *d* болсын дейік. Криптографиялық талдаушы $d|m$ болатындығын болжайды, мұндағы *m* – кілттің ұзындығы. Егер d_1, d_2, ..., d_n ара-қашықтығымен бірнеше қайталанатын сегменттерді табатын болсақ, онда $gcd\ (d_1, d_2, ..., d_m....)/m$ болады (қазіргі заманғы әдебиеттерде ең үлкен ортақ бөлгіш үшін *gcd* белгіленуі қолданылады – *greatest common divisor* ағылшын сөзінен – *ең үлкен*

ортақ бөлгіш дегенді білдіреді). Бұл болжам логикалық дұрыс, себебі егер екі символ бірдей болса және алғашқы мәтінде ерекшеленген екі символ $k \times m$ ($k = 1, 2, ...$) болса, онда шифрланған мәтінде ерекшеленген $k \times m$ символдары да бірдей болады. Криптографиялық талдаушы символдардың бірдей кілті болатын жағдайлардан қашу үшін кемінде үш символдан тұратын сегменттерді қолданады.

2. Кілттің ұзындығы табылғаннан кейін криптографиялық талдаушы аддитивті шифрды қолданады – $m = 1$ болғанда *Виженер* ауыстыру шифрының жеке жағдайы. Шифрланған мәтін m әртүрлі бөліктерге бөлінеді және символдардың пайда болу жиілігінің статистикалық шабуылын қоса отырып аддитивті шифрдың криптографиялық талдауында қолданылатын әдісті пайдаланылады. Бүтін алғашқы мәтінді алу үшін шифрланған мәтіннің әр m бөлігі керішифрланады және басқа m бөліктермен байланысады. Барлық шифрланған мәтін алғашқы мәтіннің жеке әріптерінің пайда болу жиілігін сақтамайды, бірақ әр бөлік оны орындайды.

2.17 мысал бұл пікірлерді түсінуге көмектеседі.

2.17 мысал. Зиян келтіруші төмендегі шифрланған мәтінді ұстап алды дейік:

LIOMWGFEGGDVWGHHCQUCRHRWAGWIOWQLKGZETK – KMEVLWPCZVG'TH VTSGXQOVGCSVETQLTJSUMVWVEU – VLXEWSLGFZMVVWLGYHCUSWXQHKVGSHEEVFLCFDGV – SUMPHKIRZDMPHHBVWVWJWIXGFWLTSHGJOUEEHHVUC – FVGOWICQLIJSUXGLW.

Зиян келтірушіге осы шифрланған мәтінді керішифрлау керек.

Шешім. Үш символды сегменттің қайталануына жүргізілетін *Касиски* тесті 2.6 кестеде көрсетілген нәтижелерге әкеледі.

2.6-кесте

2.17 мысал үшін Касиски тесті

Комбинация	*Бірінші ара-қашықтық*	*екінші ара-қашықтық*	*Айырма*
JSU	*68*	*168*	*100*
SUM	*69*	*117*	*48*
VWV	*72*	*132*	*60*
MPH	*119*	*127*	*8*

Ең үлкен ортақ бөлгіш – *4*, бұл төртке тепе-тең кілттің ұзындығын білдіреді. Алдымен зиян келтіруші $m = 4$ байқап көреді. Шифрланған мәтінді төрт бөлікке бөледі. c_1 бөлігі *1, 5, 9, ...* символдарынан тұрады; c_2 бөлігі *2, 6, 10, ...* символдарынан және ары қарай. Ол 2.7 кестеде көрсетілгендей бүтін алғашқы мәтінді алу үшін керішифрланатын бөліктерді бір символы бойынша бір мезгілде талдайды.

2.7 кесте

2.17 мысалы үшін Касиски тесті көмегімен
шифрланған мәтінді керішифрлау үрдісі

c_1	*LWGW*	*CRAO*	*KTEP*	*GTQC*	*TJVU*	*EGVG*
m_1	*jueu*	*apym*	*ircn*	*eroa*	*rhts*	*thin*
c_1	*UQGE*	*CVPR*	*PVJG*	*TJEU*	*GCJG*	
m_1	*ytra*	*hcie*	*ixst*	*hcar*	*rehe*	
c_2	*IGGG*	*QHGW*	*GKVC*	*TSOS*	*QSWV*	*WFVY*
m_2	*usss*	*ctsl*	*swho*	*feae*	*ceih*	*cete*
c_2	*SHSV*	*FSHZ*	*HWWF*	*SOHC*	*OQSL*	
m_2	*soec*	*atnp*	*nkhe*	*rhck*	*esex*	
c_3	*OFDN*	*URWQ*	*ZKLZ*	*HGVV*	*LUVL*	*SZWH*
m_3	*lcae*	*rotn*	*whiw*	*edss*	*irsi*	*irh*
c_3	*WKHF*	*DUKD*	*HVIW*	*HUHF*	*WLUW*	
m_3	*eteh*	*retl*	*tiid*	*eatr*	*airt*	
c_4	*MEVN*	*CWIL*	*EMWV*	*VXGE*	*TMEX*	*LMLC*
m_4	*iard*	*yseh*	*aisr*	*rtca*	*piaf*	*pwte*
c_4	*XVEL*	*GMIM*	*BWHL*	*GEVV*	*ITX*	
m_4	*thec*	*arha*	*esft*	*erec*	*tpt*	

Егер алғашқы мәтіннің мағынасы болмаса, зиян келтіруші басқа *m*-ді байқап көреді. Қарастырылып отырған жағдайда алғашқы мәтінде мағына бар (бағандар бойынша оқимыз):

Julius Caesar used a cryptosystem in his wars, which is now referred to as Caesar chipper. It is an additive chipper with the key set to three. Each character in the plaintext is shift three characters to create ciphertext.

Бұл мәтін аудармасы:

Юлий Цезарь өз соғыстарында криптографиялық жүйені қолданған, енді ол Цезарь шифры деп аталады. Бұл кілті үшке белгіленген аддитивті шифр. Шифрланған мәтінді алу үшін алғашқы мәтінде әр символ үш символға жылжытылған.

Орыс алфавитін қолданып хабарды шифрлау мен керішифрлауды қолдану үшін 2.8 кестені қолдануға болады.

Хилл шифры

Көпалфавитті шифрдың тағы бір қызықты мысалы *Лестер С. Хилл* ойлап тапқан *Хилл шифры*. Осыған дейін қарастырылған басқа көпалфавитті шифрлардан айырмашылығы мұнда алғашқы мәтін тең өлшемді бөліктерге бөлінген. Блоктағы әр символ блоктағы басқа символдарды шифрлауға үлесін қосатындай тәсілмен блоктар бір бірден шифрланған. Осы себеппен *Хилл* шифры *блокты шифрлар* деп аталатын шифрлар категориясына жатады.

Қарастырылған басқа шифрлар *ағынды шифрлар* деп аталатын категорияға жатады. Блок және ағын шифрлары арасындағы айырмашылық 3 және 4 тарауларда қарастырылады.

$$\begin{pmatrix} k_{11} & k_{12} & \dots & k_{1g} \\ k_{21} & k_{22} & \dots & k_{2g} \\ \dots & \dots & \dots & \dots \\ k_{g1} & k_{g2} & \dots & k_{gg} \end{pmatrix}$$

2.10 сурет - Хилл шифрында кілт

Шифрланған мәтіннің бір блогы қалай алынатынын көрсетеміз. Егер m_1, m_2, ..., m_g алғашқы мәтіннің n символдар блоктарын белгілесек, шифрланған мәтіннің блоктарындағы сәйкес символдар c_1, c_2, ..., c_g болады. Сол кезде келесі болады:

$$c_1 = m_1 \cdot k_{11} + m_2 \cdot k_{21} + \dots + m_g \cdot k_{g1};$$

$$c_2 = m_1 \cdot k_{12} + m_2 \cdot k_{22} + \dots + m_g \cdot k_{g2};$$

$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$

$$c_g = m_1 \cdot k_{1g} + m_2 \cdot k_{2g} + \dots + m_g \cdot k_{gg}.$$

Теңдеулер шифрланған мәтіннің c_1 сияқты әр символы (m_1, m_2, ..., m_g) блоктағы барлық алғашқы мәтіннің символдарына тәуелді екенін көрсетеді. Дегенмен барлық төртбұрышты матрицалардың Z_g-дағы мультипликативті инверсиясы болмайтынын білу қажет, сондықтан жіберуші мен қабылдаушы кілтті таңдауда байқау қажет.

.8-кесте

Орыс алфавиті үшін Виженер тізімі																															
	а	*б*	*в*	*г*	*д*	*е*	*ж*	*з*	*и*	*й*	*к*	*л*	*м*	*н*	*о*	*п*	*р*	*с*	*т*	*у*	*ф*	*х*	*ц*	*ч*	*ш*	*щ*	*ъ*	*ы*	*ь*	*э*	*ю*	*я*
А	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*
Б	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*
В	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*
Г	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*
Д	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*
Е	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*
Ж	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*
З	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*
И	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*
Й	*Й*	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*
К	*К*	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*
Л	*Л*	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*
М	*М*	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*
Н	*Н*	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*
О	*О*	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*
П	*П*	*Р*	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*Щ*	*Ъ*	*Ы*	*Ь*	*Э*	*Ю*	*Я*	*А*	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*Й*	*К*	*Л*	*М*	*Н*	*О*

2.8-кестесінің жалғасы:

Орыс алфавиті үшін Виженер тізімі																																
	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
Р	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
С	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
У	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ъ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ы	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
Э	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
Ю	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Егер матрицада мультипликативті инверсия болмаса, қабылдаушы алған шифрланған мәтінді керішифрлай алмайды. *Хилл шифрында кілттік матрицаның мультипликативті инверсиясы болу қажет.*

Матрицаларды қолдану жіберушіге барлық алғашқы мәтінді шифрлауға мүмкіндік береді. Бұл жағдайда алғашқы мәтін – l блок номері болып табылатын $l \times g$ матрицасы. *Хилл* шифры көмегімен мәтіндік хабарларды шифрлау келесі ереже бойынша жүзеге асырылады:

$$C(l \times g) = M(l \times g) \times K(g \times g) \bmod n, \qquad (2.13)$$

ал керішифрлау:

$$M(l \times g) = C(l \times g) \times K^{-1}(g \times g) \bmod n. \qquad (2.14)$$

2.18 мысал. code is ready (*код дайын*) алғашқы мәтінін шифрлау қажет, ал содан кейін шифрланған мәтінді *Хилл* шифры көмегімен *K* (*GYBNQKURP* әріптік түрде) кілттік матрицасын қолданып керішифрлау қажет болсын:

$$K = \begin{pmatrix} 06 & 24 & 01 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}.$$

Шешім. Алғашқы мәтін соңғы блокқа *z* қосымша жалған символын қосу және бос орындарды өшіру кезінде *4×3* өлшемді матрица ретінде көрсетілуі мүмкін. Бұндай жайғдайда шифрлау үшін алғашқы мәтін келесі болады: *codeisreadyz*. Бұл хабардың сандық эквиваленті келесі болады: *02, 14, 03, 04, 08, 18, 17, 00, 03, 24, 25*.

Хилл шифры көмегімен хабарды шифрлау 2.11 суретінде көрсетілген.

$$\underset{C}{\begin{pmatrix} 20 & 11 & 05 \\ 20 & 10 & 16 \\ 24 & 04 & 05 \\ 24 & 23 & 20 \end{pmatrix}} = \underset{M}{\begin{pmatrix} 02 & 14 & 03 \\ 04 & 08 & 18 \\ 17 & 04 & 00 \\ 03 & 24 & 25 \end{pmatrix}} \times \underset{K}{\begin{pmatrix} 06 & 24 & 01 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}}.$$

2.11 сурет - 2.18 мысалы үшін *Хилл* шифры көмегімен хабарларды шифрлау

Көріп отырғанымыздай, шифрлау нәтижесінде келесі шифрланған хабарды аламыз: *20, 11, 05, 20, 10, 16, 24, 04, 05, 24, 23, 20*. Бұл шифрланған хабар ағылшын алфавитін қолданумен *ULFUKQYEFYXU* сәйкес келеді.

Қабылдаушы $(K \times K^{-1})$ *mod n = I,* бұл жерде *I* – бірлік матрица болатын, K^{-1} мультипликативті инверстік (кері) матрица-кілтін қолданып хабарды керішифрлай алады. Біздің мысалымыз үшін *K* матрица-кілтін K^{-1} мультипликативті инверстік (кері) матрица-кілтіне көбейту келесіге тең болады:

$$K \times K^{-1} = \begin{pmatrix} 06 & 24 & 01 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \times \begin{pmatrix} 08 & 05 & 10 \\ 21 & 08 & 21 \\ 21 & 12 & 08 \end{pmatrix} = \begin{pmatrix} 01 & 00 & 00 \\ 00 & 01 & 00 \\ 00 & 00 & 01 \end{pmatrix} = I.$$

2.18 мысалында шифрланған хабарды *Хилл* шифры көмегімен керішифрлау 2.12 суретінде көрсетілген.

$$\underset{M}{\begin{pmatrix} 02 & 14 & 03 \\ 04 & 08 & 18 \\ 17 & 04 & 00 \\ 03 & 24 & 25 \end{pmatrix}} = \underset{C}{\begin{pmatrix} 20 & 11 & 05 \\ 20 & 10 & 16 \\ 24 & 04 & 05 \\ 24 & 23 & 20 \end{pmatrix}} \times \underset{K^{-1}}{\begin{pmatrix} 08 & 05 & 10 \\ 21 & 08 & 21 \\ 21 & 12 & 08 \end{pmatrix}}$$

2.12 сурет - 2.18 мысалы үшін шифрланған хабарды *Хилл* шифры көмегімен керішифрлау

2.12 суреттен керішифрлау нәтижесінде ағылшын алфавитін қолданып *codeisreadyz* сәйкес келетін келесі алғашқы мәтін алынды: *2, 14, 03, 04, 08, 18, 17, 04, 00, 03, 24, 25*. Соңғы символды алып тастап, нәтиже ретінде кіріс мәтініне сәйкес келетін co*deisready* мәтінін аламыз.

Хилл шифрымен шифрланған мәтін үшін криптографиялық талдау қиын. Біріншіден, Хилл шифрын қолдану кезінде «қатқыл күш» шабуылы өте күрделі, себебі матрица-кілт – $g \times g$. Әр кірісте *n* мәннің біреуі болуы мүмкін. Бұл кілт өлшемі $n^{g \times g}$ екенін білдіреді. Дегенмен, барлық матрицаларда мультипликативті кері матрица-кілт болмайды. Сондықтан кілттің болу аймағы өте үлкен емес.

Екіншіден, *Хилл* шифры қарапайым мәтін статистикасын сақтамайды. Зиян келтіруші екі немесе үш әріптен тұратын жеке

әріптің пайда болу жиілігіне статистикалық шабуыл жүргізе алмайды. g өлшемді сөз жиілігін талдау жүруі мүмкін, бірақ алғашқы мәтінде g өлшемді көп жеке жолдардың болуы өте сирек. Зиян келтіруші алғашқы мәтінді білу әдісін қолданып, шифрға шабуыл жасауы мүмкін, егер де ол g мәнін білсе және жоқ дегенде m блоктар бойынша *алғашқы мәтін/шифрланған мәтін* жұбын білсе. Блоктар сол хабарға немесе әртүрлі хабарларға тиісті болуы мүмкін, бірақ әртүрлі болу керек. Зиян келтіруші екі $g \times g$ матрица құру мүмкін, олар сәйкес жолдары *алғашқы мәтін/шифрланған мәтін* белгілі жұбын көрсететін M (*алғашқы мәтін*) және C (*шифрланған мәтін*). $C = M \times K$ болғандықтан, зиян келтіруші егер M қайтымды болып табылса, кілтті табу үшін, $K = M^{-1} \times C$ қатынасын қолдану мүмкін. Егер M қайтымды болмаса, онда зиян келтіруші *алғашқы мәтін/шифрланған мәтін* жұбының g әртүрлі жиындарын іске қосу керек.

Егер зиян келтіруші g мәнін білмесе, онда m өте үлкен болмайтын шартта ол әртүрлі мәндерді байқай алады.

2.19 мысал. Зиян келтіруші $g = 3$ екенін біледі дейік. Ол 2.14 суретте көрсетілгендей *алғашқы мәтін/шифрланған мәтін* блогының үш жұбын ұстап алды (сол хабардың өзінен болуы міндетті емес).

$$\begin{array}{ccc} (05\ \ 07\ \ 10) & \longleftrightarrow & (03\ \ 06\ \ 00) \\ (13\ \ 17\ \ 07) & \longleftrightarrow & (14\ \ 16\ \ 19) \\ (00\ \ 05\ \ 04) & \longleftrightarrow & (03\ \ 17\ \ 11) \\ M & & C \end{array}$$

2.13 сурет - 2.19 мысал үшін *Хилл* шифры көмегімен шифрланған мәтінді қалыптастыру

Шешім. Зиян келтіруші 2.11 суретте көрсетілген жұптардан M және C матрицаларын құрады. Бұл жағдайда M матрицасы қайтымды болғандықтан, ол бұл матрицаны төңкереді және оны C-ға көбейтеді, бұл 2.14 суретінде көрсетілгендей K кілттің матрицасын береді.

Енді оның кілті бар және бұл кілт қолданылатын кез келген шифрланған мәтінді бұза алады.

$$\underset{K}{\begin{pmatrix} 02 & 03 & 07 \\ 05 & 07 & 09 \\ 01 & 02 & 11 \end{pmatrix}} = \underset{M^{-1}}{\begin{pmatrix} 21 & 14 & 01 \\ 00 & 08 & 25 \\ 13 & 03 & 08 \end{pmatrix}} \times \underset{C}{\begin{pmatrix} 03 & 06 & 00 \\ 14 & 16 & 09 \\ 03 & 17 & 11 \end{pmatrix}}$$

2.14 сурет - 2.19 мысалы үшін *Хилл* шифрының матрица-кілтін *K* алу

Шифрлаудың бір реттік жүйесі

Криптографияның мақсаттарының бірі – мінсіз құпиялылық. Тәжірибеде қолданылатын шифрлардың көбісі *шартты сенімді* болып сипатталады, себебі олар шектелмеген есептеу мүмкіндіктері бар кезінде ашылу мүмкін. *Өте сенімді шифрларды* тіпті шектелмеген есептеу мүмкіндіктерді қолдану кезінде де бұзуға болмайды.

1917 жылы американдық *Г. Вернам* мен *Д. Моборн* [3, 9] ойлап тапқан және тәжірибеде қолданатын осындай жалғыз шифр бар – *бір реттік блокнот*. Мұндай шифрдың ерекшілігі кілттік тізбекті бір рет қолданылуы болып табылады. Әдебиеттерде шифрлаудың бұл жүйесін *Вернам шифры* деп атайды. Бұл шифрдың кілтінің ұзындығы кіріс мәтіндікіндей, және мүлде кездейсоқ сайланады.

1949 жылы *Вернам шифрының* абсолютті беріктілігін дәлелдейтін *К. Шеннонның* жұмысы жарық көрді. *Шеннонның* жұмысы осындай қасиеттері бар басқа шифрдың жоқ екендігін көрсетеді. Бұл мынадай тұжырымның шығуына әкелді: *Вернам* шифры – барлық криптографиялық жүйелер ішіндегі ең қауіпсіз.

К. Шеннонның зерттеулері мінсіз құпиялылыққа төмендегі ережелердің орындалуы кезінде жетуге болатындығын көрсетті:

- шифрлау үшін кілт кездейсоқ таңдалады;
- кілттің ұзындығы шифрланатын мәтін ұзындығына тең болу керек;
- кілт тек бір рет қана қолданылу керек.

Егер бір кілтті қолданса, аддитивті шифрды оңай бұзуға болатындығы белгілі. Бірақ бұл шифрдың өзі де идеалды болуы мүмкін, егер де әр символды шифрлау үшін (*00, 01, 02, ..., n-1*) кілттер жиынынан кездейсоқ таңдалатын кілт қолданылса: егер бірінші символ *04* кілті көмегімен шифрланса, екінші – *02* кілт көмегімен, үшінші – *21* кілт және әрі қарай. Сол кезде тек шифрланған мәтінге шабуыл мүмкін емес болады. Егер жіберуші әр

кезде бүтін санның басқа кез келген реттілігін қолданып кілтті өзгерткен жағдайда, онда шабуылдың басқа да түрлері мүмкін емес болады.

Ауыстырудың осындай жүйесін жүзеге асыру үшін кейде *бірреттік блокнотты* қолданады. Бұл блокнот әрқайсысына k_i кездейсоқ сандары (кілттері) бар кесте басылған үзілмелі парақтардан тұрады. Блокнот екі экземплярда орындалады: біреуі жіберушімен қолданылады, ал басқасы – қабылдаушымен. Хабардың m_i әр символы үшін кестеден k_i өз кілті тек бір рет қолданады. Кесте қолданылғаннаң кейін, ол блокноттан өшіріліп жойылуы керек. Жаңа хабарды шифрлау жаңа парақтан басталады.

Бұл шифр егер K кілттер жиыны шын мәнінде кездейсоқ және болжап болмайтын болса, абсолютті сенімді болады. Егер криптографиялық талдаушы шифрланған деректерге арналған кілттердің барлық мүмкін жиынын қолдануға тырысса және кіріс деректердің нұсқаларын қалпына келтірсе, онда олар тең ықтималды болады. Яғни жіберілген кіріс деректерді таңдау мүмкіндігі жоқ.

Осындай артықшылықтың арқасында бір реттік жүйелер абсолютті ақпараттық қауіпсіздікті талап ететін барлық жағдайларда қолдану қажет. Бірақ бір реттік жүйелерді қолдану мүмкіндіктері тәжірибелік аспектілерімен шектелген. Кездейсоқ кілттік тізбекті бір рет қолдану талабының ерекшелігі болып табылады. Хабардың ұзындығынан қысқа емес ұзындықты кілттік тізбек қабылдаушыға алдын ала немесе кейбір жабық арна бойынша жеке жіберілуі қажет. Бұл талап шын мәнінде маңызды бірретті хабарларды жіберу үшін өте қиын емес. Бірақ мұндай талап көп миллион символдарды шифрлауды талап ететін заманауи ақпаратты өңдеу жүйелері үшін орындармайтын талап.

Бір реттік блокноттардың кейбір нұсқаларында кілттік тізбекті қарапайым басқаруға сүйенеді, бірақ бұл шифрдың сенімділігінің төмендеуіне әкеледі. Мысалы, кілт хабарды жіберуші мен қабылдаушыға белгілі кітапта орынды белгілеумен анықталады. Кілттік тізбек бұл кітаптың белгіленген орнымен басталады және *Виженер* жүйесіндегідей қолданылады. Кейде мұндай шифр *жүгіртпелі кілті бар шифр* деп те аталады. Шифрдың мұндай нұсқасында кілттік тізбекті басқару әлдеқайда жеңіл, себебі ұзақ кілттік тізбекті шағын пішінде беріле алады. Бірақ басқа жағынан алғанда, бұл кілттер кездейсоқ емес. Сондықтан криптографиялық

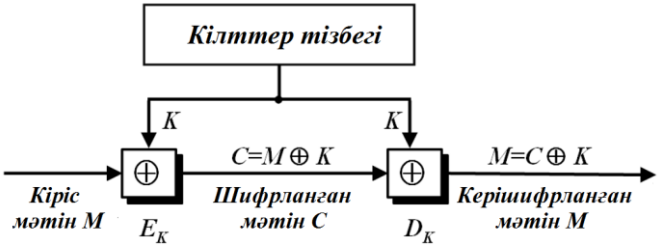
талдаушыда кіріс табиғи тілі символдарының жиілігі жөнінде ақпаратты қолданумен сәйкес шабуылды жүзеге асыру мүмкіндігі пайда болады.

Вернам шифрлау жүйесі шын мәнінде, модуль мәні $n = 2$ болатын *Виженер* шифрлау жүйесінің жеке жағдайы болып табылады. 1926 жылы AT&T (АҚШ) фирмасының қызметкері *Г. Вернам* ұсынған осы шифрдың нақты нұсқасы кіріс деректер символдарының екілік бейнеленуін қолданады.

Алты қосымша символдармен (бос орын, қайтару пернесі және тағы басқа) кеңейтілген ағылшын алфавитінен {*A, B, C, D, ... , Z*} кіріс ашық мәтінінің әр символы алдымен *Бод телеграфтық кодының* бес биттік блогында (b_0, b_1, b_2, b_3, b_4) кодталды.

{ k_0, k_1, k_2, ... } екілік бес биттік кілттердің кездейсоқ тізбегі алдын ала қағаз лентада жазылды.

Вернам әдісімен шифрлауды қолданып хабарды жіберу сұлбасы 2.15 суретінде көрсетілген.



2.15 сурет - *Вернам* әдісімен хабарларды шифрлау және керішифрлау сұлбасы

Алдын ала n екілік символдар тізбегіне түрлендірілген кіріс деректерді шифрлау k кілтінің тізбегін n символдарымен 2 модулі бойынша қосу жолымен жүзеге асырылады.

Шифрланған деректер символдары келесі түрде түрленеді:

$$c_i = m_i \oplus k_i,\ i = 1, 2, 3, \ldots, n. \qquad (2.15)$$

Керішифрлау c_i шифрланған деректер символдарын сол k_i кілт тізбегімен 2 модулі бойынша қосумен алынады:

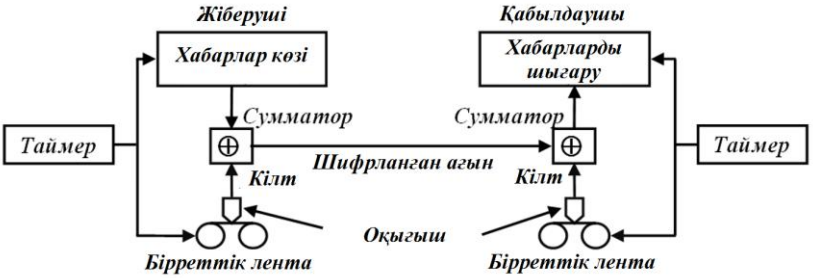
$$m_i = c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i,\ i = 1, 2, 3, \ldots, n. \qquad (2.16)$$

Бұл жерде шифрлау және керішифрлау кезінде қолданылатын кілттер тізбегі бір бірін толықтырады (*2* модулі бойынша қосу кезінде) және осының нәтижесінде m_i кіріс деректер символдары қалпына келтіріледі.

Вернам өз жүйесін құру кезінде оны кілттердің сол тізбегін қолдану үшін жіберілетін және қабылданатын жақтарда орнатылған шығыршықталған ленталар көмегімен тексерді (2.16-сурет).

Нақты жүйелерде басынан бастап кілттің кездейсоқ сандарымен екі бірдей ленталарды дайындайды. Біреуі жіберушіде, ал басқасы - *ұстап ала алмайтын* әдіспен, мысалы, күзеті бар курьермен, заңды қабылдаушыға жіберіледі. Жіберуші хабарды жібергісі келген кезде, ол алдымен оны екілік пішінге айналдырады және құрылғыға орналастырады, кілттік лентада анықталған сандарды хабардың әр санына *2* модулі бойынша қосады.

Қабылданатын жақта шифрланған хабар шифрлауға қолданылатын құрылғыға ұқсайтын машинадан өтеді және жазылады. Бұл құрылғы хабардың әр екілік санына кілттік лентада анықталған санды *2* модулі бойынша қосады (алады, себебі *2* модулі бойынша алу мен қосу эквивалентті), осылайша ашық мәтінді алады. Бұл жерде кілттік лента шифрлауға қолданылатын өзінің көшірмесімен абсолютті синхронды жылжу керек.

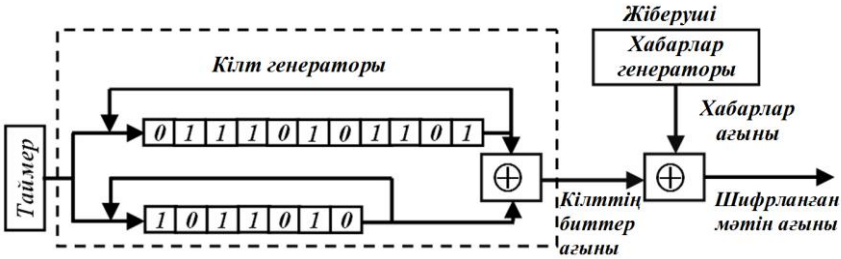


2.16 сурет - Шығыршықталған ленталарды қолданатын *Вернам* шифрлау жүйесі

Бұл жүйенің маңызды кемшілігі жіберілген ақпараттың әр биті үшін кілттік ақпараттың биті алдын ала дайындалу қажет, және де

ол биттер кездейсоқ болуы қажет. Деректердің үлкен көлемін шифрлау маңызды шектеу болып табылады. Сондықтан бұл жүйе тек қана ең жоғарғы құпиясы бар хабарларды жіберуге қолданылады.

Үлкен көлемді құпия кілтінің алдыңғы жіберілуінің проблемасын айналып өту үшін инженерлер мен өнертапқыштар кейбір алгоритмдерге сәйкес келетін бірнеше қысқа ағындардан тұратын псевдокездейсоқ сандардың өте ұзақ ағындарды генерациялаудың көп сұлбаларын ойлап тапты. Шифрланған хабарды алушыны тура жіберушідегідей генератормен қамтамасыз ету керек. Бірақ мұндай алгоритмдер шифрланған мәтінге жүйелілік қосады, оның анықталуы криптографиялық талдаушыға хабарды керішифрлауға көмектеседі. Осындай генераторларды құрудың негізгі әдістерінің бірі анықталған деректер *аралас* ағындарды алу үшін екі не одан көп биттік ленталарды қолдануда болады (2.17-суретте).



2.17 сурет - Аралас ағынды қолданатын *Вернам* шифрлау жүйесі

Вернам әдісі кілттің тізбегінің ұзындығына тәуелді емес, кілттердің кездейсоқ тізбегін қолдануға мүмкіндік беретіндігін атап өткен дұрыс. Дегенмен *Вернам* әдісін жүзеге асырған кезде қабылдаушыға тура жіберушідегідей кілттер тізбегін жеткізу қажеттілігімен байланысты, немесе жіберушіде де, қабылдаушыда да кілттердің сәйкес тізбегін қаупсіз сақтау қажеттілігімен ауыр проблемалар туындайды. *Вернам* шифрлау жүйесінің бұл кемшіліктері гаммалау әдісімен шифрлауды қолдану кезінде жойылады.

Гаммалау әдісімен шифрлау

Гаммалау әдісімен шифрлаудың мәні шифрланатын деректердің символдары гамма деп аталатын кейбір арнайы тізбектің символдарымен қосуда. Кейде мұндай әдіс кіріс деректерге гамманы салу ретінде көрсетіледі, осыдан *«гаммалау»* атауы берілген.

Кіріс деректерге гамманы салу процедурасын екі тәсілмен жүзеге асыруға болады.

Бірінші тәсілде кіріс деректердің және гамма символдары сандық эквиваленттермен ауыстырылады, содан кейін n модулі бойынша қосылады, мұндағы n – алфавиттегі символдар саны, яғни:

$$c_i = (m_i + g_i) \bmod n, \quad i = 1, 2, 3, \ldots, n, \tag{2.17}$$

бұл жерде c_i, m_i, g_i – шифрланатын мәтін, кіріс мәтіні және гамманың символдары.

2.20 мысал. *Гамбит* ашық мәтіні болсын, сандық эквивалент 2.8 кесте бойынша келесі: *03 00 12 01 08 18*. Гамма: *Модель - 12 14 04 05 11 28*. Орыс алфавиті үшін $n = 32$.

Шешім: Ашық мәтінді шифрлау үшін (2.17) өзара қатынасын қолданып төмендегіге қол жеткіземіз:

$$c_1 = (03 + 12) \bmod 32 = 15; \quad c_2 = (00 + 14) \bmod 32 = 14;$$

$$c_3 = (12 + 04) \bmod 32 = 16; \quad c_4 = (01 + 05) \bmod 32 = 06;$$

$$c_5 = (08 + 11) \bmod 32 = 19; \quad c_6 = (18 + 28) \bmod 32 = 14.$$

Сонымен, шифрланған мәтін шықты: *15 14 16 06 19 14*, ол 2.2 суретін есепке алғанда келесі түрде көрсетіледі: *Поржуо*.

Екінші тәсілді қолдану кезінде кіріс мәтінінің және гамма символдары екілік код түрінде көрсетіледі, ал содан кейін сәйкес разрядтар *2* модулі бойынша қосылады:

$$c_{ij} = m_{ij} \oplus g_{ij}, \quad i = 1, 2, 3, \ldots, n; \quad j = 0, 1, \ldots, l, \tag{2.18}$$

бұл жерде $\oplus$ – *2* модулі бойынша қосу операциясы; c_{ij}, m_{ij} және g_{ij} – шифрланатын мәтін, кіріс мәтін және гамманың сәйкес екілік

символдары; n – шифрланатын деректердің символдар саны (шифрланған деректердің символдар саны; гамманың символ саны); l – кіріс символдарының, гамманың және шифрланатын символдардың екілік биттерінің саны.

2.21 мысал. Ашық мәтін мен гамма жоғарыда көрсетілген мысалдағыдай болсын.

Шешім: Шифрлау үшін 2.18 өзара қатынасын қолданамыз:

	03	*00*	*12*	*01*	*08*	*18*
	00011	*00000*	*01100*	*00001*	*01000*	*10010*
⊕						
	01100	*01110*	*00100*	*00101*	*01011*	*11100*
	01111	*01110*	*01000*	*00100*	*00011*	*01110*
	15	*14*	*08*	*04*	*03*	*14*

Поидго-ға сәйкес келетін *15 14 08 03 04 14* шифрланған хабарын аламыз.

2 модулі бойынша қосудың орнына гаммалау кезінде басқа логикалық операцияларды қолдануға болады, мысалы, логикалық эквиваленттілік немесе логикалық эквивалентсіздік. Мұндай ауыстыру кіріс мәтіні мен гамма символдарынан шифрланған хабар символдарын қалыптастыру ережесін таңдау болып табылатын тағы бір кілттің енгізілуімен шамалас.

Гаммалау әдісімен шифрлау беріктілігі, ең басты, гамманың қасиеттерімен анықталады – период ұзақтығы және статистикалық қасиеттерінің теңөлшемділігімен. Соңғы қасиеті период ішінде әртүрлі символдардың пайда болу кезіндегі заңдылықтардың жоқтығын қамтамасыз етеді.

Керішифлау шифрланған мәтіні мен гаммаға символдарына кері операцияның қолданумен жүзеге асырылады:

$$c_i = (m_i - g_i) \bmod n, \quad i = 1, 2, 3, \ldots, l.$$

немесе

$$c_{ij} = m_{ij} \oplus g_{ij}, \quad i = 1, 2, 3, \ldots, n; \quad j = 0, 1, \ldots, l,$$

2.22 мысал. Алдыңғы мысалдан (*2.21 мысал*) екі әдіс бойынша шифрланған мәтінді керішифрлаймыз.

Шешім: Бірінші әдіс үшін:

$m_1 = (15 - 12) \bmod 32 = 03$; $m_2 = (14 - 14) \bmod 32 = 00$;
$m_3 = (16 - 04) \bmod 32 = 12$; $m_4 = (06 - 05) \bmod 32 = 01$;

$m_5 = (19 - 11) \bmod 32 = 08$; $m_6 = (14 - 28) \bmod 32 = 18$.

Сонымен, *Гамбит*-ке сәйкес келетін *03 00 12 01 08 18* ашық мәтіні алынды.

Екінші әдіс үшін:

	15	*14*	*08*	*04*	*03*	*14*
	01111	*01110*	*01000*	*00100*	*00011*	*01110*
⊕						
	01100	*01110*	*00100*	*00101*	*01011*	*11100*
	00011	*00000*	*01100*	*00001*	*01000*	*10010*
	03	*00*	*12*	*01*	*08*	*18*

Көріп отырғанымыздай, тағы да *Гамбит* ашық мәтініне жауап беретін *03 00 12 01 08 18* ашық мәтіні алынды.

Гаммалау әдісіне негізделген шифрлау жүйесінің беріктілігі гамманың қасиеттеріне – оның ұзындығына және гамма белгілерінің пайда болу ықтималдығының таратылу біркелкілігіне тәуелді.

Гаммалау әдісімен шифрлаудың екі түрі бар – шекті және шексіз гаммамен. Гамманың ең жақсы статистикалық қасиеттері кезінде шифрлау беріктілігі тек период ұзындығымен анықталады. Бұған қоса егер гамма периодының ұзындығы шифрланған мәтін ұзындығынан асып кетсе, онда мұндай шифр теориялық тұрғыдан *абсолютті берік* болып табылады. Дегенмен, бұл керішифрлау тіпті мүмкін емес дегенді білдірмейді: кейбір қосымша ақпарат бар кезінде кіріс мәтін шексіз гамманы қолданған кездің өзінде жартылай немесе толығымен қалыпқа келтірілген бола алады.

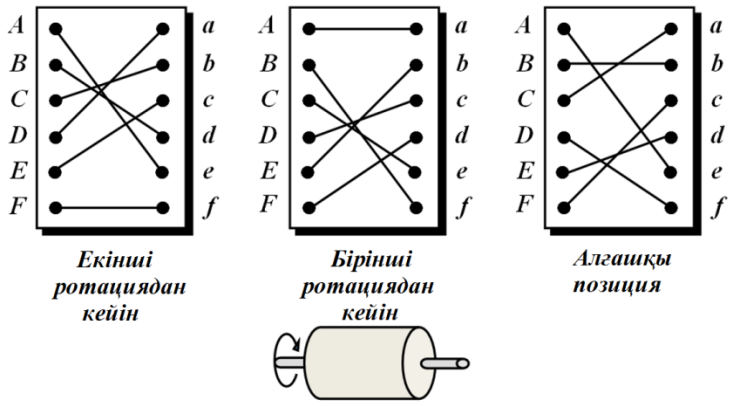
Шексіз гамма ретінде кездейсоқ символдардың кез келген тізбегін қолданыла алады, мысалы, π немесе e санының цифрлар реттілігі. Электронды-есептеу машинамен шифрлау кезінде гамма тізбегі псевдокездейсоқ сандар датчигі көмегімен қалыптасады. Қазіргі таңда қанағаттандырылатын қасиеттерін қамтамасыз ететін осындай датчиктер жұмысының алгоритмдері өңделген [32].

Ауыстырудың роторлы шифры

Бір реттік блокнот шифры тәжірибеде қолданылмаса да, одан ең қорғалған шифрға бір қадам – ол *ауыстырудың роторлы шифры*. Ол моноалфавитті ауыстыру идеясына қайта келеді, бірақ алғашқы мәтіннің әр символы үшін шифрланған мәтіннің символына бейнелену принципін өзгертеді. 2.18 сурет роторлы шифрдың жеңілдетілген мысалын көрсетеді.

2.18 суретінде көрсетілген ротор тек 6 әріпке қолданылған, бірақ нақты роторлар ағылшын алфавитінің 26 әріпін қолданады. Ротор алғашқы және шифрланған мәтіндер символдарын үнемі байланыстырады, бірақ қосылу щетка арқылы қамтамасыз етіледі. Назар аударыңыздар, алғашқы және шифрланған мәтіндер символдарының қосылуы, егер ротор мөлдір және ішкі бөлігін көре алуға болатындай көрсетілген.

Ротордың алғашқы орнатылуы (позияциясы) – жіберуші мен қабылдаушы арасында құпиялау кілті – бұл алғашқы мәтіннің шифрланған бірінші символы. Алғашқы орнатуды қолданып, екінші символ бірінші айналым жүргізілгеннен кейін шифрланады (2.19 суретте – бұл дөңгелектің *1/6*-не бұрылыс, нақты орнатуда – *1/26*-ге бұрылыс) және ары қарай.



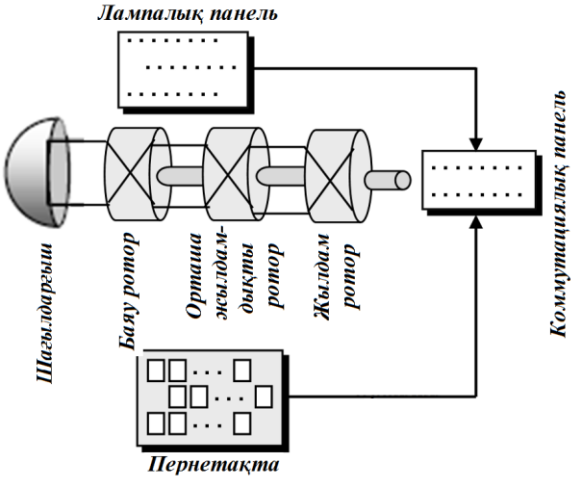
2.18 сурет - Ауыстырудың роторлы шифры

bee сияқты үш әріпті сөз *BAA* ретінде шифрланады, егер ротор қозғалмаған (ауыстырудың моноалфавитті шифры) болса, бірақ

егер ол бұрылса (ауыстырудың роторлы шифры), ол *BCA* болып шифрланады. Бұл ауыстырудың роторлы шифры – көпалфавитті шифры екенін көрсетеді, себебі алғашқы мәтіннің символының екі рет пайда болуы әртүрлі символдар ретінде шифрланған.

Ауыстырудың роторлы шифры ауыстырудың моноалфавитты шифры сияқты «қатқыл күш» шабуылына берік, себебі зиян келтіруші мүмкін *n!* (*n* факториал) арасында бейнеленуінің бірінші жиынын табу керек. Ауыстырудың роторлы шифры ауыстырудың моноалфавиттік шифрына қарағанда статистикалық шабуылға беріктірек болып табылады, себебі онда алфавиттің әрпін қолдану жиілігі сақталмайды.

"*Энигма*" ауыстырудың роторлы машинасы бастапқыда Сербияда ойлап табылды, бірақ неміс әскерінің мамандары оны өзгертті және Екінші дүние жүзілік соғысы кезінде қарқынды қолданды [2, 32]. Машина роторлы шифрлар принциптеріне негізделген. 2.19 сурет "*Энигма*" ауыстырудың машинасын құрудың жеңілдетілген сұлбасын көрсетеді.



2.19 сурет - "Энгима" ауыстырудың машинасының мысалды құрылуы

Төменде машинаның басты компоненттері атап көрсетілген:

1. 26 кілті бар пернетақта, шифрлау кезінде алғашқы мәтінді енгізу және керішифрлау кезінде шифрланған мәтінді енгізу үшін қолданылады.

2. 26 лампасы бар лампалық панель, шифрлау кезінде шифрланған мәтін символдарын және керішифрлау кезінде алғашқы мәтін символдарын көрсетеді.

3. Он үш өткізгішпен қолмен жалғанған 26 штепселі бар коммутациялық панель. Конфигурация күнде түрлі скремблирлеуді қамтамасыз ету үшін өзгеріп отырады.

4. Алдыңғы секцияларда қарастырылғандай үш құрастырылған роторлар. Бұл үш роторлар күнделікті бес қолжетімді роторлардан таңдалады. Тез ротор пернетақта арқылы енгізілген әр символ кезінде 1/26 бұрылысқа айналады. Орташа ротор тез ротордың әр толық айналысында 1/26 бұрылыс жасайды. Баяу ротор орташа ротордың әр аяқталған бұрылысы үшін 1/26 бұрылыс жасайды.

5. Үнемі және алдын ала құрастырылған шағылдырғыш.

«Энигма» ауыстыру машинасын қолдану үшін кодтық кітап шығарылды, бұл кітап әр күн ішінде төмендегілерді қоса отырып, баптаудың бірнеше параметрлерін береді:

а) бес қол жетімділерінен таңдалатын үш ротор;

б) осы роторлар орнатылатын тәртіп;

в) коммутациялық панель үшін орнату параметрлері;

г) күннің үш әріпі бар код.

Хабарды шифрлау үшін оператор төменде көрсетілген қадамдарды тізбекті орындауы қажет:

1. Күн кодына сәйкес роторлардың бастапқы позициясын орнату. Мысалы, егер код *HUA* болса, роторлар сәйкесінше *H, U* және *A*-ға инициализацияланған болуы қажет.

2. Үш әріпі бар кездейсоқ кодты таңдау, мысалы *ACF*. 1 қадамның роторлардың бастапқы орнатылуын қолдана отырып, *ACFACF* (қайталанбалы код) мәтінін шифрлау. Мысалы, шифрланған – *OPNABT* деп алайық.

3. *OPN*-ға (шифрланатын кодтың жартысы) роторлардың бастапқы позицияларын орнату.

4. 2 қадамда алынған шифрланған алты әріпті (*OPNABT*) алғашқы хабардың аяғына қосу.

5. Алты әріпі бар кодты қоса, хабарды шифрлау. Шифрланған хабарды жіберу.

Хабарды керішифрлау үшін оператор төмендегі қадамдарды орындау қажет:

1. Хабарды алу және алғашқы алты әріпті бөлек шығару.

2. Күн кодына сәйкес роторлардың бастапқы позициясын орнату.

3. 2 қадамның бастапқы орнатылуын қолданып хабардың алғашқы алты әріпін керішифрлау.

4. Керішифрланған кодтың бірінші жартысына роторлардың позицияларын орнату.

5. Хабарды керішифрлау (бастапқы алты әріпісіз).

«Энигма» ауыстыру машинасы соғыс кезінде бұзылды, дегенмен соғыстан кейін бірнеше онжылдықта неміс әскері мен дүние жүзінің басқа бөлігі бұл факті білмеді [6]. Неміс тілі өте күрделі болса да, одақтастар машинаның бірнеше көшермелерін алды. Келесі қадам әр хабарға роторларды инициализациялау үшін жіберілетін код пен әр күн ішінде орнату параметрлерін іздеу болды. Бірінші компьютерді ойлап табу одақтастарға осы қиындықтарды жеңуге көмектесті.

Рюкзакты жінау туралы есепті шешу әдісімен шифрлау

Ашық кілтті жалпылама шифрлау үшін бірінші алгоритмі *Ральф Меркел* мен *Мартин Хеллман* құрастырған рюкзак алгоритмі болды [5, 32]. Рюкзак алгоритмінің қауіпсіздігі рюкзак мәселесіне сүйенеді. Рюкзак мәселесі күрделі емес. Әртүрлі салмағы бар заттар жиыны берілген. Осы заттардың кейбіреулерін рюкзак салмағы берілген мәнге тең болатындай рюкзаққа салуға болады. Немесе формалды түрде, m_1, m_2, ..., m_n мәндер жиыны және C сомасы берілген. b_i мәнін есептеу қажет:

$$C = b_1 \cdot m_1 + b_2 \cdot m_2 + \ldots + b_n \cdot m_n,$$

бұл жерде b_i нөл немесе бір бола алады. Бір зат рюкзаққа салынғанын, ал нөл – салынбағанын көрсетеді.

Мысалы, заттар салмақтарының мәндері *1, 5, 6, 11, 14* және *20* болу мүмкін. Рюкзакты оның салмағы *22*-ге тең болатындай жинауға болады. Бұны егер *5, 6* және *11* салмақтарын қолданса ғана, орындауға болады. Рюкзакты оның салмағы *24*-ке тең болатындай жинауға мүмкін емес. Жалпы жағдайда осы мәселені шешуге қажетті уақыт жиындағы заттардың санының өсуімен бірге экспоненциалды өседі.

Фокус рюкзактың екі әртүрлі проблемасы болатындығында, біреу сызықтық уақытта шешіледі, ал басқасы, олай емес. Жеңіл

проблеманы қиындатуға болады. Ашық кілт қиын проблема, оны шифрлау үшін жеңіл қолдануға болады, бірақ хабарды керішифрлау үшін қолдану мүмкін емес. Жабық кілт жеңіл проблема, ол хабарды керішифрлауға қарапайым әдіс береді. Жабық кілтті білмейтіндерге рюкзактың қиын мәселесін шешуді тырысуға тура келеді.

Меркел-Хеллман рюкзак алгоритмінің негізінде рюкзак жинау проблемасын шешу сияқты хабарды шифрлау идеясы жатыр. Жиыннан заттар ашық мәтін блогы көмегімен жиындағы заттар санына тең ұзындығы (ашық мәтін биттері *b* мәніне сәйкес келеді) бойынша таңдалады, ал шифрланған мәтін алынған сома болып табылады.

Рюкзак (ранец) жинау туралы есептер төмендегідей қалыптасады:

Вектор берілген:

$$E = \left| e_к, e_{к-1}, \ldots, e_2, e_1 \right|,$$

ол ары қарай кілт рөлін атқарады. Ережеге сай, бұл вектордың элементтері жай сандар. Бір мағынада шифрлау және керішифрлау үшін *E* векторының элементтері ереже бойынша таңдалады [13, 15]

$$e_i > \sum_{j=1}^{i-1} e_j, \quad i = 2,3,\ldots,k .$$

Ашық хабардың жіберілетін әр символы m_i *k* биттердің тұратын тізбек ретінде беріледі:

$$m_i = \left| m_{i,k}, \quad m_{i,k-1}, \quad \ldots, \quad p_{i,1} \right|^T, \quad m_{i,j} \in \{0,1\},$$
$$j = 1,2,\ldots,k, i = 1,2,\ldots,n.$$

Нәтиже ретінде ашық хабар матрица болады

$$M = \begin{vmatrix} m_{1,k} & m_{2,k} & \ldots & m_{n,k} \\ m_{1,k-1} & m_{2,k-1} & \ldots & m_{n,k-1} \\ \ldots & \ldots & \ldots & \ldots \\ m_{1,1} & m_{2,1} & \ldots & m_{n,1} \end{vmatrix},$$

бұл жерде *n* – ашық хабар ұзындығы.

C шифрланған деректер символдары *E* матрицасы мен *M* вектор-бағанының көбейтіндісінен шығады:

$$C = E \times M. \tag{2.19}$$

2.23 мысал. Төмендегі кілтті қолданып, *Обгон* (*14, 01, 03, 14, 13*) ашық хабарын шифрлау:

$$E = \left| 29 \quad 13 \quad 7 \quad 3 \quad 2 \right|.$$

Вектор-жол элементтері жай сандар.

Шешім: ашық хабар символдарын бес разрядты екілік код түрінде жазамыз:

О	*б*	*Г*	*о*	*н*
14	*01*	*03*	*14*	*13*
01110	*00001*	*00011*	*01110*	*01101*

M хабар матрицасын құрамыз:

$$M = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{vmatrix}.$$

(2.19) сәйкес келетін операцияларды орындаймыз:

$$C = \left| 29 \quad 13 \quad 7 \quad 3 \quad 2 \right| \cdot \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{vmatrix} = \left| 23 \quad 02 \quad 05 \quad 23 \quad 22 \right|.$$

Сонымен, шифрланған хабар келесі түрге ие: *23, 02, 05, 23, 22* (*ЧВЕЧЦ*).

2.24 мысал. Осы шифрланған хабарды керішифрлаймыз.

Шешім: Ашық хабардың бесразрядты екілік сандары (2.19) матрицалық теңдеуден шығады.

Ашық хабарды алты теңдеуді шығарып алуға болады:

$$C = \left| 29 \quad 13 \quad 07 \quad 03 \quad 02 \right| \cdot \begin{vmatrix} m_{1,5} & m_{2,5} & m_{3,5} & m_{4,5} & m_{5,5} \\ m_{1,4} & m_{2,4} & m_{3,4} & m_{4,4} & m_{5,4} \\ m_{1,3} & m_{2,3} & m_{3,3} & m_{4,3} & m_{5,3} \\ m_{1,2} & m_{2,2} & m_{3,2} & m_{4,2} & m_{5,2} \\ m_{1,1} & m_{2,1} & m_{3,1} & m_{4,1} & m_{5,1} \end{vmatrix} =$$

$$= \begin{vmatrix} 29 \cdot m_{1,5} + 13 \cdot m_{1,4} + 7 \cdot m_{1,3} + 3 \cdot m_{1,2} + 2 \cdot m_{1,1} \\ 29 \cdot m_{2,5} + 13 \cdot m_{2,4} + 7 \cdot m_{2,3} + 3 \cdot m_{2,2} + 2 \cdot m_{2,1} \\ 29 \cdot m_{3,5} + 13 \cdot m_{3,4} + 7 \cdot m_{3,3} + 3 \cdot m_{3,2} + 2 \cdot m_{3,1} \\ 29 \cdot m_{4,5} + 13 \cdot m_{4,4} + 7 \cdot m_{4,3} + 3 \cdot m_{4,2} + 2 \cdot m_{4,1} \\ 29 \cdot m_{5,5} + 13 \cdot m_{5,4} + 7 \cdot m_{5,3} + 3 \cdot m_{5,2} + 2 \cdot m_{5,1} \end{vmatrix} = \begin{vmatrix} 23 \\ 02 \\ 05 \\ 23 \\ 22 \end{vmatrix}.$$

Бірінші скалярлы теңдеуді шешіп, хабарламаның бірінші элементін (бесразрядты екілік сан) аламыз.

Бірінші скалярлы теңдеу:

$$29 \cdot m_{1,5} + 13 \cdot m_{1,4} + 7 \cdot m_{1,3} + 3 \cdot m_{1,2} + 2 \cdot m_{1,1} = 23.$$

Бұл скалярлы теңдеу егер $m_{1,5} = 0$, $m_{1,4} = 1$, $m_{1,3} = 1$, $m_{1,2} = 1$ і $m_{1,1} = 0$ болған жағдайда әділ болады.

Бұл 14_{10} екілік коды = 01110_2.

Екінші скалярлы теңдеу:

$$29 \cdot m_{2,5} + 13 \cdot m_{2,4} + 7 \cdot m_{2,3} + 3 \cdot m_{2,2} + 2 \cdot m_{2,1} = 02.$$

Бұл скалярлы теңдеу егер $m_{2,5} = 0$, $m_{2,4} = 0$, $m_{2,3} = 0$, $m_{2,2} = 0$ і $m_{2,1} = 1$ болған жағдайда әділ болады.

Бұл 01_{10} екілік коды = 00001_2.

Үшінші скалярлы теңдеу:

$$29 \cdot m_{3,5} + 13 \cdot m_{3,4} + 7 \cdot m_{3,3} + 3 \cdot m_{3,2} + 2 \cdot m_{3,1} = 05.$$

Бұл скалярлы теңдеу егер $m_{3,5} = 0$, $m_{3,4} = 0$, $m_{3,3} = 0$, $m_{3,2} = 1$ і $m_{3,1} = 1$ болған жағдайда әділ болады.

Бұл 03_{10} екілік код = 00011_2.

Төртінші скалярлы теңдеу:

$$29 \cdot m_{4,5} + 13 \cdot m_{4,4} + 7 \cdot m_{4,3} + 3 \cdot m_{4,2} + 2 \cdot m_{4,1} = 23.$$

Мұндай скалярлы теңдеу егер $m_{4,5} = 0$, $m_{4,4} = 1$, $m_{4,3} = 1$, $m_{4,2} = 1$ і $m_{4,1} = 0$ болған жағдайда әділ болады.

Бұл 14_{10} екілік код = 01110_2.

Бесінші скалярлы теңдеу:

$$29 \cdot m_{5,5} + 13 \cdot m_{5,4} + 7 \cdot m_{5,3} + 3 \cdot m_{5,2} + 2 \cdot m_{5,1} = 22.$$

Мұндай скалярлы теңдеу егер $m_{5,5} = 0$, $m_{5,4} = 1$, $m_{5,3} = 1$, $m_{5,2} = 0$ і $m_{5,1} = 1$ болған жағдайда әділ болады.

Бұл 13_{10} екілік код = 01101_2.

Сонымен, мына ашық хабарды аламыз:

01110	*00001*	*00101*	*01110*	*01101*
14	*01*	*05*	*14*	*13*
О	*б*	*г*	*о*	*н*

Алынғаннан *Обгон* ашық хабары шығады.

Бұл алгоритмді жүзеге асырудың көптеген нұсқасы ұсынылды, бірақ түгелге жуығы бұзылды. Қалғандары басқа асимметриялық криптографияның алгоритмдеріне қарағанда бұзу үшін аз күш талап етеді.

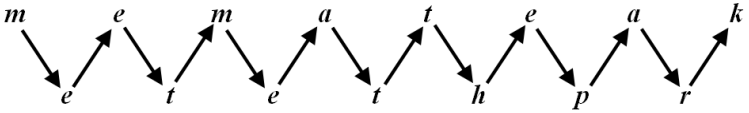
2.2. ОРЫН АУЫСТЫРУ ШИФРЛАРЫ

Орын ауыстыру шифрлары бір символды басқасымен ауыстырмайды, оның орнына ол шифрланған мәтіндегі символдардың орналасқан орнын өзгертеді. Алғашқы мәтіннің бірінші позициясындағы символ шифрланған мәтіннің оныншы позициясында пайда болу мүмкін. Алғашқы мәтіннің сегізінші позициясында орналасқан символ шифрланған мәтіннің бірінші позициясында пайда болу мүмкін. Басқаша айтқанда, орын ауыстыру шифры символдарды басқа тәртіппен (орнын ауыстырады) орналастырады.

2.2.1. Кілтті қолданусыз орын ауыстыру шифрлары

Бұрын қолданылған *қарапайым орын ауыстыру шифрлары* кілт қолданбады. Символдардың орнын ауыстырудың екі әдісі бар. Бірінші әдісте мәтін кестеде бағаннан бағанға жазылады және содан кейін жолдан жолға жіберіледі. Екінші әдісте мәтін кестеде жолдан жолға жазылады, содан кейін бағаннан бағанға жіберіледі.

2.25 мысал. Кілтті қолданбайтын шифрдың жақсы мысалы – *қоршау шифры* (*rail fence cipher*). Бұл шифрда алғашқы мәтін зигзаг сияқты шаблон ретінде екі сызыққа орналасқан (бұл екі жолы бар кестенің бағаннан бағанға сияқты қарастырылу мүмкін); шифрланған мәтін шаблонды жолдан жолға оқу кезінде құрылады. Мысалы, *Meet me at the park* (*мені саябақта күт*) хабарын жіберу үшін жіберуші алушыға жазады (2.20 сурет).



2.20 сурет - Қоршау шифры көмегімен *Meet me at the park* хабарын шифрлау мысалы

Жіберуші екінші жолмен сүйемелденетін бірінші жолды жіберіп *MEMATEAKETETHPR* шифрланған мәтінін құрады. Қабылдаушы шифрланған мәтінді алады және он екіге бөледі (бұл жағдайда екінші жартысы бір символға кем болады). Форманың бірінші жартысы – бірінші жол, екінші жартысы – екінші жол. Қабылдаушы нәтижені зигзаг бойынша оқиды. Ешқандай кілт болмағандықтан және жол нөмірі орнатылғандықтан (*2*), зиян келтіруші үшін шифрланған мәтіннің криптографиялық талдауы өте жеңіл болар еді. Оның бар білу керектігі – бұл қоршау шифры қолданылатындығы.

m	*e*	*e*	*t*
m	*e*	*a*	*t*
t	*h*	*e*	*p*
a	*r*	*k*	

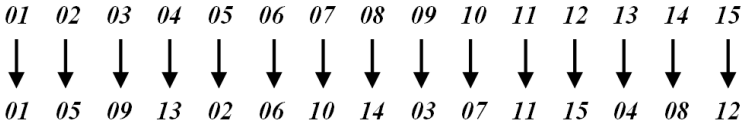
2.22 сурет - Кесте көмегімен *Meet me at the park* хабарламасын шифрлау мысалы

2.26 мысал. Жіберуші мен қабылдаушы бағандар саны туралы келісіп екінші әдісті қолдана алады. Жіберуші сол бір алғашқы

мәтінді, жолдан жолға, төрт бағаннан тұратын кестеге жазады (2.21 сурет).

Жіберуші символдарды бағаннан бағанға жіберіп, *ММТАЕЕНREAEKTTP* шифрланған мәтінін құрады. Қабылдаушы шифрланған мәтінді алады және кері үрдісті қолданады. Ол алынған шифрланған мәтінді бағаннан бағанға жазады және оны жолдан жолға алғашқы мәтін ретінде оқиды. Зиян келтіруші оның бағандар санын білсе, хабарды оңай керішифрлай алады.

2.23 мысалындағы шифр – орын ауыстырудың нақты шифры. Ары қарай алғашқы мәтіннің әр әрпінің орын ауыстыруын және олардың позициясының нөміріне негізделіп шифрланған мәтінді көрсетеміз:



Алғашқы мәтіндегі екінші символ шифрланған мәтінде бесінші позицияға ауысты; үшінші символ тығызыншы позицияға жылжыды және тағы сол сияты. Символдардың орны ауысқанымен, олар өздері шаблон болады: (*01, 05, 09, 13*), (*02, 06, 10, 14*), (*03, 07, 11, 15*) және (*04, 08, 12*). Әр секцияда екі көршілес номерлер арасындағы айырма *–4.*

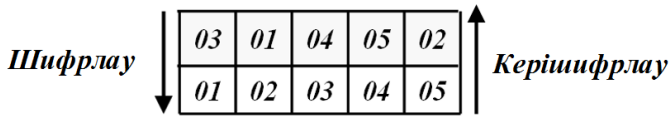
2.2.2. Кілтті қолданумен орын ауыстыру шифрлары

Кілтті қолданумен орын ауыстыру шифрлары алғашқы мәтін жазуын бір тәсілмен (мысалы, жолдан жолға) және осы мәтінді басқа ретпен (мысалы, бағаннан бағанға) жіберуді қолданып символдардың орнын ауыстырады. Барлық шифрланған мәтінді құру үшін, орын ауыстыру барлық алғашқы мәтінге жасалады. Басқа әдіс алғашқы мәтінді блок деп аталатын, алдын ала өлшемі анықталған топтарға бөледі, содан соң әрбір блокта символдардың орнын ауыстыру үшін кілтті қолданады.

2.27 мысал. Жіберуші қабылдаушыға *Enemy attacks tonight (Жаудың шабуылдары бүгін кешке)* деген хабарды жіберуі қажет. Жіберуші мен қабылдаушы мәтінді бес әріптен топқа бөліп, әрбір

топтағы символдардың орнын ауыстыруға келісті. Төменде соңына жалған символ қосылған (басқа топтармен өлшемін сәйкестендіру үшін қолданылады) топтама көрсетілген: *Enemy attacks tonightz*.

Шифрлау мен керішифрлау үшін қолданылатын кілт – орын ауыстыру кілті, ол символдарды қалай ауыстыру керектігін көрсетеді. Бұл хабар үшін, жіберуші мен қабылдаушы келесі кілтті қолданды деп қабылдайық:



Алғашқы мәтін блогының үшінші символы шифрланған мәтін блогының бірінші символы болады, ал алғашқы мәтін блогының бірінші символы шифрланған мәтін блогының екінші символы болады және ары қарай. Орын ауыстыру нәтижесі: *EEMYN TAACT TKONS HITZG*

Жіберуші қабылдаушыға *EEMYN TAACT TKONS HITZG* шифрланған мәтінді жибереді. Қабылдаушы шифрланған мәтінді бес символдан тұратын топтарға бөледі, кілтті кері ретпен қолданып, алғашқы мәтінді табады.

Кілтті қолданумен бағандардың орнын ауыстыру шифры

Заманауи орын ауыстыру шифрлары, жақсы скремблирлеуге жету үшін, екі әдісті біріктіреді. Шифрлау және керішифрлау үш кезеңде жүзеге асырылады.

Бірінші қадам: алғашқы мәтін жолдан кейін жол ретімен кестеге жазылады.

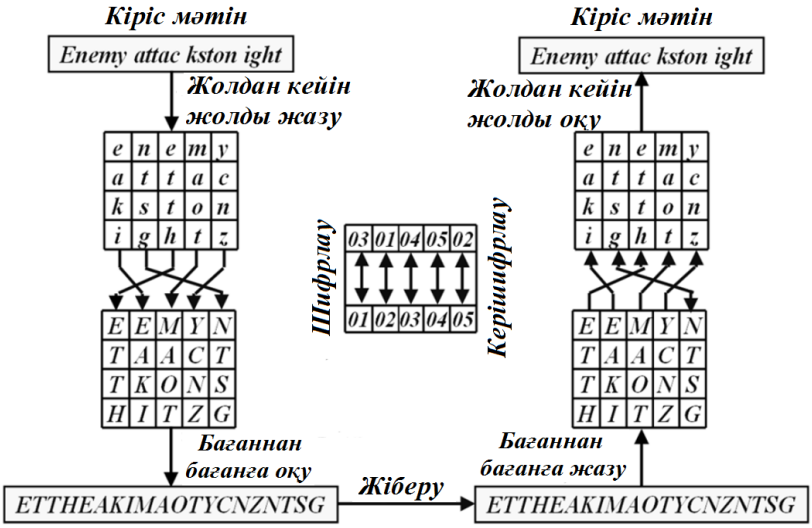
Екінші қадам: бағандардың ретін өзгерту арқылы ауыстыру жүргізіледі (берілген кілт бойынша кестенің бағандары ауыстырылады).

Үшінші қадам: шифрланған мәтін жаңа кестеден бағаннан кейін баған ретімен оқылады.

Бірінші және үшінші қадамдар кілтсіз ғаламдық ретті өзгертуді қамтамасыз етеді, ал екінші қадам блоктық кілтті орын ауыстыруды қамтамасыз етеді. Бұл шифрлар типтері жиі *кілтті қолданып бағандарды ауыстыру шифры* деп аталады.

2.28 мысал. Жіберуші *Enemy attacks tonight* хабарын біріктірілген әдісті қолданып шифрласын. Шифрлау және керішифрлау 2.22 суретінде көрсетілген.

Жіберуші құрған бірінші кесте, жолдан кейін жол жазылған алғашқы мәтіннен тұрады. Бағандар алдындағы мысалда көрсетілген кілтті қолдану арқылы орын ауыстырылған. Шифрланған мәтін бағаннан кейін бағанды оқу арқылы екінші кестедегі символдарды оқу арқылы құрылған. Қабылдаушы осы үш қадамды да жасайды, бірақ кері ретпен. Ол шифрланған мәтінді баған бойынша бірінші кестеге жазады, осы кестенің бағандарын ауыстырады, содан соң қатар бойынша екінші кестеден символдарды оқиды.

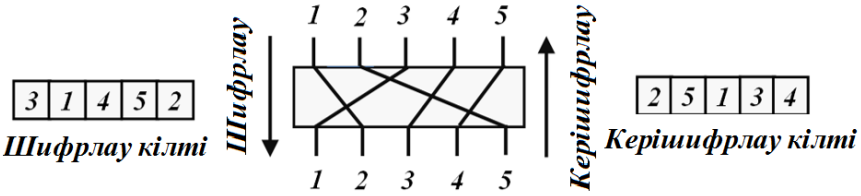


2.22 сурет - Кілтті қолданумен орын ауыстыру жолымен шифрлау және керішифрлау мысалы

2.28 мысалында жалғыз кілт бағандар ретін өзгерту үшін қолданылды - төмен қарай – шифрлау үшін, жоғары қарай – керішифрлау үшін. Әдетте бұл графикалық бейнелеу үшін екі кілт құру қабылданған: біреуі шифрлау үшін, екінші керішифрлау үшін. Кілттер әр баған үшін бір ғана мекен-жайы (кіріс) бар кестеде жиналады. Кірісте бағанның алғашқы нөмірі болады – жеткізу бағанының нөмірі, кіріс нөмірінің орнын көрсетеді. 2.23 суретте

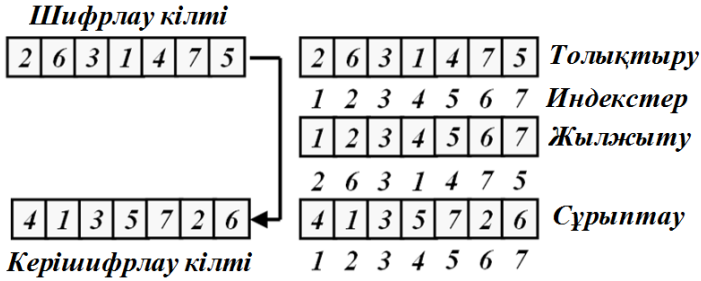
осы екі кесте кілтті графикалық бейнелеу көмегімен құрылуы көрсетілген.

Шифрлау кілті – (*3 1 4 5 2*). Бірінші кіріс, алғашқы мәтіндегі 3 баған (мазмұны) жеткізілетін орында *1* бағанға (орны немесе кіріс индексі) айналатынын көрсетеді. Керішифрлау кілті – (*2 5 1 3 4*). Бірінші кіріс, алғашқы мәтіндегі *2* баған жеткізілетін орында *1* айналатынын көрсетеді.



2.23 сурет - Екі кілт көмегімен орын ауыстыру шифрында шифрлау / керішифрлау

Егер шифрлау кілті берілсе, керішифрлау кілтін қалай табуға болады және кері қарай керішифрлау кілті берілсе, шифрлау кілтін қалай табуға болады? Үрдісті 2.24 суретте көрсетілгендей қолмен бірнеше қадам арқылы орындауға болады.



2.24 сурет - Орын ауыстыру шифрындағы кілтті инверсиялау үрдісінің түсініктемесі

Алдымен индекстерді кілттер кестесіне қосамыз, содан кейін алынған кілтке сәйкес жылжыту жасаймыз және соңында жұптарды индекстерге сәйкес сұрыптаймыз.

Орын ауыстыру шифры үшін шифрлау / керішифрлау үрдістерін көрсету үшін матрицаларды қолдануға болады. $M(l \times g)$ алғашқы мәтіні және $C(l \times g)$ шифрланған мәтіні - $l \times g$ өлшемді символдардың сандық мәнін көрсететін матрицалар; $K(g \times g)$ кілттері - $g \times g$ өлшемді орын ауыстырудың төртбұрышты матрицалары.

Бұл жағдайда деректерді шифрлау алгоритмін келесі түрде көрсетуге болады:

$$C(l \times g) = M(l \times g) \times K_{Ш}(g \times g), \qquad (2.20)$$

ал керішифрлауды

$$M(l \times g) = C(l \times g) \times K_{Р}(g \times g), \qquad (2.21)$$

бұл жерде $K_{Ш}(g \times g)$ және $K_{Р}(g \times g)$ – шифрлау және керішифрлау үшін қолданылатын кілттік матрицалар. Кілттік матрицалар кері матрица болып табылады, яғни:

$$K_{Р}(g \times g) = K_{Ш}^{-1}(g \times g) \text{ және } K_{Ш}(g \times g) = K_{Р}^{-1}(g \times g).$$

Берілген алгоритмде (2.20) және (2.21) бағандардың орнын ауыстыру үрдістерін сипаттайтын кілттік матрицалар, сондықтан оларды жиі *орын ауыстыру матрицалары* деп атайды.

Орын ауыстыру матрицаларында әрбір қатар немесе бағанда тек қана бір бірлік (*1*) болады, және қалған мәндері нөлге (*0*) тең болады. Шифрланған мәтіннің матрицасын алу үшін шифрлау алғашқы мәтін матрицасын шифрлаудың кілттік матрицасына (түзу орын ауыстырудың матрицасы) көбейту арқылы орындалады; керішифрлау шифрланған мәтінді кілттік матрицасына (орын ауыстырудың инверсті матрицасы) көбейту арқылы орындалады, соңында алғашқы мәтінді аламыз. Қызықты жағдай, бұл жағдайда және әрқашан керішифрлау матрицасы шифрлаудың инверсті матрицасы. Бірақ керішифрлау матрицасын инверстеудің еш қажеті жоқ, себебі керішифрлаудың кілттік матрицасын алу үшін, шифрлаудың кілттік матрицасында бағандар мен қатарлардың жылжыту қажет.

2.29 мысал. Жіберілетін хабар *Enemy attacks tonight*. Шифрлау кілті: *3, 1, 4, 5, 3*. Шифрлау кілтімен *3, 1, 4, 5, 3* алғашқы деректер

матрицасының бағандарын орын ауыстыру арқылы хабарды шифрлау.

Шешім: Enemy attacks tonight хабары сандық эквивалент түрінде келесі *04, 13, 04, 12, 24, 00, 19, 19, 00, 02, 10, 18, 19, 14, 13, 08, 06, 07, 19, 25*. 2.25 сурет хабарды шифрлау үрдісін көрсетеді.

Алғашқы мәтін матрицасын $M(4 \times 5)$ шифрлаудың кілттік матрицасына $K_Ш(5 \times 5)$ көбейту шифрланған мәтін матрицасын $C(4 \times 5)$ береді. 2.29 мысалындағы матрицалық жұмыс символдарды сандық мәндерге өзгеруін қажет етеді (*00* ден *25*-ке дейін). Назар аударыңыз, матрицалық көбейту тек бағандардың орын ауыстыруын қамтамасыз етеді; матрицаға жазу және оқуды алгоритмнің басқа бөліктері қамтамасыз ету қажет.

$$\begin{pmatrix} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{pmatrix} = \begin{pmatrix} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{pmatrix} \times \begin{matrix} \begin{matrix} 3 & 1 & 4 & 5 & 2 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$

Шифрланған мәтін Алғашқы мәтін Орын ауыстыру матицасы

2.25 сурет - 2.29 мысалы үшін орын ауыстыру шифрын қолданумен деректерді шифрлау үрдісін түсіндіру

Сонымен, шифрланған хабар түрі *04, 04, 12, 24, 13, 19, 00, 00, 02, 19, 19, 10, 14, 13, 18, 07, 08, 19, 25, 06*, ол: *EEMYNTAACTTKONSHITZG* сәйкес келеді.

2.30 мысал. 2.29 мысалында алынған шифрланған хабарды алғашқы деректер матрицасының бағандарын *2, 5, 4, 1, 3* кілт бойынша орын ауыстыруды қолданумен керішифрлау. Кілт шифрлаудың инверсты матрицасынан алынады (2.29 мысалын қараңыз).

Шешім: Шифрланған мәтін матрицасын $C(4 \times 5)$ керішифрлаудың кілттік матрицасына $K_Р(5 \times 5)$ көбейту $M(4 \times 5)$ алғашқы мәтін матрицасын береді.

2.26 суреті хабарды керішифрлау үрдісін көрсетеді.

$$\begin{pmatrix} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{pmatrix} = \begin{pmatrix} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{pmatrix} \times \begin{matrix} 2 & 5 & 1 & 3 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Шифрланған мәтін Алғашқы мәтін Орын ауыстыру матрицасы

2.26 сурет - 2.30 мысалы үшін орын ауыстыру шифрын қолданумен деректерді керішифрлау үрдісін түсіндіру

Сонымен, керішифрлау кезінде алғашқы мәтін келесі *04, 13, 04, 12, 24, 00, 19, 19, 00, 02, 10, 18, 19, 14, 13, 08, 06, 07, 19, 25* болады, ол: *Enemy attacks tonightz* сәйкес келеді.

Орын ауыстыру шифрларын криптографиялық талдау

Орын ауыстыру шифрлары тек қана шифрланған мәтінге шабуылдың бірнеше түрлеріне осал.

Статистикалық шабуыл

Орын ауыстыру шифры шифрланған мәтінде әріптердің жиілігін өзгертпейді; ол тек қана әріптердің орнын ауыстырады. Сондықтан, қолдануға болатын, бірінші шабуыл – шифрланған мәтінде бөлек әріптердің пайда болу жиілігін талдау болып табылады. Егер шифрланған мәтіннің ұзындығы үлкен болатын болса, онда бұл әдіс пайдалы болады. Мұндай шабуылдың мысалы алдында қарастырылған болатын. Бірақ орын ауыстыру шифрлары үшграммалар мен жұптардың пайда болу жиілігін сақтамайды. Бұл зиян келтіруші осындай құрал жабдықтарын қолдана алмайтындығын көрсетеді. Нақты алатын болсақ, егер шифр үшграммалар мен жұптардың пайда болу жиілігін сақтамайтын болса, бірақ жеке әріптердің пайда болу жиілігін сақтайтын болса, онда ол орын ауыстыру шифры болып табылады.

Зиян келтіруші хабарды оқу үшін барлық мүмкін кілттерді қолдануы мүмкін. Бірақ кілттер саны көп болуы мүмкін *1! + 2! + 3!*

+ ... + *L!*, бұл жерде *L* – шифрланған мәтін ұзындығы. Ең тиімді әдісі, бағандардың санын табу. Зиян келтіруші бағандардың саны *g* ға бөлінетінін біледі. Мысалы, егер шифр ұзындығы – 20 символ болатын болса, онда *20 = 1×2×2×5*. Бұл бағандардың нөмірі мына (*1, 2, 4, 5, 10, 20*) коэффициенттер комбинациясы болуы мүмкін екендігін білдіреді. Бірақ тек қана бір баған және бір жол – азықтималды нұсқалар [8, 32].

2.31 мысал. Зиян келтіруші 2.29 мысалын орындау кезінде алынған, шифрланған мәтін хабарын *EEMYNTAACTTKONSHITZG* ұстап алсын.

Шешім: Хабар ұзындығы *L = 20*, бағандардың саны *1, 2, 4, 5, 10* немесе *20* болуы мүмкін. Зиян келтіруші, бірінші мәнді қабылдамайды, себебі бұл тек қана бір баған және ол азықтималды.

1. Егер баған саны – *2* болса, онда екі орын ауыстыру (*1, 2*) және (*2, 1*) болады. Біріншісі, орын ауыстыру болған жоқ дегенді білдіреді. Зиян келтіруші екінші комбинацияны қарастырады. Ол шифрланған мәтінді екі символдан модульдерге бөледі: *EE MY NT AA CT TK ON SH IT ZG*. Содан соң ол әрбір модульдің орнын ауыстырып көреді және *ee ym nt aa tc kt no hs ti gz*, ешқандай мағынасы жоқ мәтінін алады.

2. Егер бағандар нөмірі – *4* болса, онда мүмкінді орын ауыстыру саны *4! = 24* тең. Бірінші орын ауыстыру (*1 2 3 4*) ешқандай орын ауыстыру болмағанын көрсетеді. Зиян келтіруші қалған мүмкін жағдайларды қарастыруы қажет. Қалған 23 мүмкіндікті тексеріп, зиян келтіруші ешқандай алғашқы мәтінің мұндай орын ауыстыруларда мағынасы болмайтындығын табады.

3. Егер бағандар нөмері – *5* болса, онда мүмкінді орын ауыстыру саны *5! = 120*. Бірінші орын ауыстыру (*1 2 3 4 5*) ешқандай орын ауыстыру болмағанын көрсетеді. Шабуылшы қалған мүмкін жағдайларды қарастыруы қажет. (*2 5 1 3 4*) орын ауыстыруы өз нәтижесін береді – алғашқы мәтін *enemyattackstonightz*, соңындағы *z* жалған әріпін алып тастап және бос орындарды орнына қойғаннаң кейін, өз мағынасын береді (*Enemy attacks tonight*).

Үлгі бойынша шабуыл

Орын ауыстыру шифрының басқа шабуылы үлгі бойынша шабуыл деп аталу мүмкін. Орын ауыстырудың кілттік шифры

көмегімен шифрланған мәтінде кейбір қайталанатын үлгілер болады. Келесі мысал 2.31 мысалындағы шифрланған мәтіннің әрбір символы алғашқы мәтіннен келесі ереже бойынша алынғандығын көрсетеді:

01 02 03 04	*05 06 07 08*	*09 10 11 12*	*13 14 15 16*	*17 18 19 20*
03 08 13 18	*01 06 11 16*	04 09 14 19	*05 10 15 20*	02 07 12 17

Шифрланған мәтіннің *1*-ші символы алғашқы мәтіннің *3*-ші символынан алынады. Шифрланған мәтіннің *2*-ші символы алғашқы мәтіннің 8-ші символынан алынады. Шифрланған мәтіннің *20*-шы символы алғашқы мәтіннің *17*-ші символынан алынады және ары қарай. Жоғарыда айтылған тізімдерді ескере отырып үлгілердің бес тобын аламыз: (*3*, *8*, *13*, *18*), (*1*, *6*, *11*, *16*), (*4*, *9*, *14*, *19*), (*5*, *10*, *15*, *20*) және (*2*, *7*, *12*, *17*). Барлық топтарда екі көршілес сандардың айырымы – *5*. Бұл жүйелілікті шифрды бұзу үшін криптографиялық талдаушы қолдануы мүмкін. Егер зиян келтіруші бағандардың санын білетін болса немесе таба алатын болса (біздің жағдайымызда ол *5*-ке тең), онда ол шифрланған мәтінді төрт символдан тұратын топтарға түрлендіре алады. Топтарды орын ауыстыру алғашқы мәтінді табу үшін кілтті қамтамасыз ету мүмкін.

Екі еселі орын ауыстырумен шифрлар

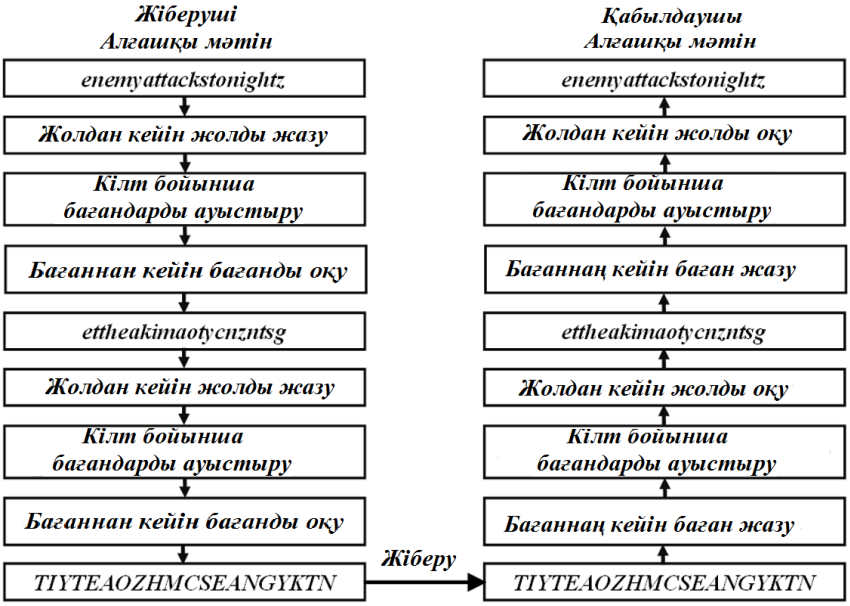
Екі еселі орын ауыстырумен шифрлар криптографиялық талдаушының жұмысын қиындату мүмкін. Мұндай шифрға мысал ретінде 2.28 мысалында шифрлау және керішифрлау үшін қолданылатын алгоритмді екі рет қайталау бола алады. Әр қадамда әртүрлі кілт қолданылуы мүмкін, бірақ әдетте бір ғана кілт қолданылады.

2.32 мысал. Екі еселі орын ауыстыру қолданылған 2.28 мысалын қайталайық. 2.27 суреті үрдісті көрсетеді.

Криптографиялық талдаушы шифрланған мәтінге статистикалық шабуыл жасау үшін бөлек символдардың пайда болу жиілігін қолдана алатын болса да, енді үлгі бойынша шабуыл қиындатылды:

13 16 05 07 03 06 10 20 18 04

10 12 01 09 15 17 08 11 19 02.



2.27 сурет - Екі еселі кілттік орын ауыстыру жолымен шифрлау және керішифрлау мысалы

Келтірілген нәтиже мен 2.29 мысалындағы нәтижені салыстыра отырып, қайталанатын үлгінің жоқ екенін көреміз. Екі еселі орын ауыстыру алдында болған жүйелілікті жойды.

Бақылау сұрақтары және тапсырмалар

1. Симметриялық шифрлаудың жалпы сұлбасын түсіндіріңіз. Барлық жабық кілтті шифрлау әдістеріне ортақ не бар?
2. Ауыстыру шифрларына анықтама беріңіз. Ауыстырумен шифрлау әдістері үшін ортақ принциптерді келтіріңіз.
3. Моноалфавитті (біралфавитті) ауыстыру шифрының мәнін түсіндіріңіз.
4. Көпалфавитті ауыстыру шифрының мәнін түсіндіріңіз.
5. Аддитивті ауыстыру шифрының мәнін түсіндіріңіз.
6. Мультипликативті ауыстыру шифрының мәнін түсіндіріңіз.
7. Аффинды ауыстыру шифрының мәнін түсіндіріңіз.
8. Автокілт ауыстыру шифрының мәнін түсіндіріңіз.
9. *Плейфер* ауыстыру шифрының мәнін түсіндіріңіз.

10. *Виженер* ауыстыру шифрының мәнін түсіндіріңіз.

11. *Виженер* кестесін қолданумен әдіс жабық кілтті шифрлау әдістері тобының қайсысына жатады? Бұл әдісте шифрлау мен керішифрлаудың қандай алгоритмдері бар?

12. *Хилл* шифрының мәнін түсіндіріңіз.

13. Шифрлаудың бір реттік жүйесінің мәнін түсіндіріңіз.

14. Гаммалау әдісімен шифрлаудың мәнін түсіндіріңіз.

15. Рюкзак жинау туралы есептер шешу әдісімен шифрлаудың мәнін түсіндіріңіз.

16. Орын ауыстыру шифрына анықтама беріңіз.

17. Кілтті қолданумен және қолданусыз орын ауыстыру шифрының мәнін түсіндіріңіз.

18. Орыс тілінде "*комната"* деген хабарды аддитивті шифрды (*K = 3*) қолданумен шифрлаңыз (2.2 суретті қараңыз).

19. Ауыстырудың аддитивті шифрын қолданып, шифрлау кезінде орыс тілін қолданған (2.2 суретті қараңыз) "*УГФХИРЛИ"* шифрланған хабарды (*K = 5*) керішифрлаңыз.

20. Егер ашық мәтін - шифрланған мәтін жұбы "*апельсин"* – "*ЗЦМТГШ ПФ"* белгілі болса, онда орыс тілін (2.2 суретті қараңыз) қолданумен ауыстырудың аддитивті шифрының кілтін анықтаңыз.

21. *K = 5* кілтімен орыс тілінде *отечество* хабарын ауыстырудың мультипликативті шифрымен шифрлаңыз. (2.2 суретті қараңыз).

22. Шифрланған "*ЖРШЙЮХРАМШЩ*" мәтінін *K = 7* кілтті және орыс тілін қолданып, мультипликативті шифры көмегімен керішифрлаңыз. (2.2 суретті қараңыз).

23. Аффинды шифрын ($k_1 = 7$ және $k_2 = 5$) қолдана отырып орыс тіліндегі "*аргумент*" хабарын шифрлаңыз (2.2 суретті қараңыз).

24. Орыс тілінде жазылған шифрланған "*ЛЗЬЩЗАЕУГ*" мәтінін аффинды шифрды қолданумен керішифрлаңыз (2.2 суретті қараңыз).

25. Ауыстырудың автокілт шифрын ($k_1 = 13$) қолдана отырып орыс тіліндегі "*гордость*" хабарын шифрлаңыз (2.2 суретті қараңыз).

26. Шифрлау кезінде орыс тілі және ауыстырудың аффинды шифры ($k_1 = 11$) қолданылған "*ЩЯГАЮЮФУЫЯГО*" мәтінін керішифрлаңыз (2.2-суретті қараңыз).

27. Плейфер шифрын қолдана отырып, орыс тіліндегі "*криптография*" хабарын шифрлаңыз (2.9 суретті қараңыз)..

28. Шифрлау кезінде орыс тілі және Плейфер шифры қолданылған "*АПЙШФЬШЧЬЛКЯ*" мәтінін керішифрлаңыз (2.9 суретті қараңыз).

29. Виженер шифрын және "*километр*" кілттік сөзін қолдана отырып, орыс тіліндегі "*комната*" хабарын шифрлаңыз. (2.8 кестесіне қараңыз).

30. Шифрлау кезінде орыс тілі, Виженер шифры және "*роза*" кілттік сөзі қолданылған "*АОЯИДЮХВРЫПЕ*" хабарын керішифрлаңыз (2.8 кестесіне қараңыз).

31. *Хилл* шифрын және 2.18 мысалынан *K* кілттік матрицасын қолданып, ағылшын тіліндегі "*perpendicular*" хабарын шифрлаңыз. (2.3 суретін қараңыз).

32. Шифрлау кезінде *Хилл* шифры, 2.18 мысалдағы K^{-1} кілттік матрицасы және ағылшын тілі қолдылған "*CVJRKDXAHIBS*" хабарын керішифрлаңыз (2.1-суретін қараңыз).

3 тарау

ЗАМАНАУИ СИММЕТРИЯЛЫҚ КРИПТОГРАФИЯЛЫҚ ЖҮЙЕЛЕРДІ ҚҰРАСТЫРУ ПРИНЦИПТЕРІ

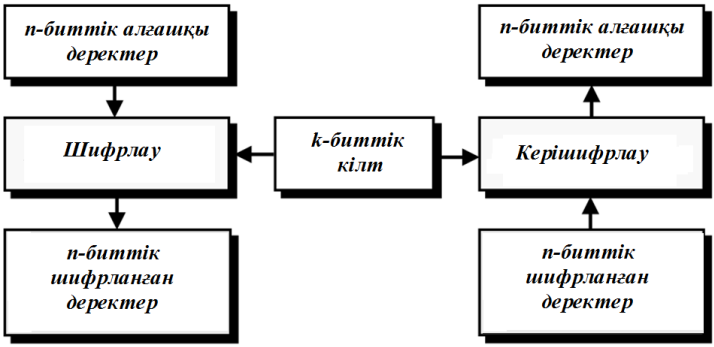
Жоғарыда қарастырылған симметриялық кілті бар дәстүрлі шифрлар символдарға бағытталады. Компьютерлік жүйелердің пайда болуымен биттерге бағытталған шифрлар қажет болды, себебі шифрланған ақпарат тек мәтіннен ғана емес, сонымен қатар сандардан, сызбалардан, аудио- және бейне- деректерден тура алады. Шифрлау үшін бұл деректер типін биттер ағынына ауыстыру қажет және содан кейін шифрланған мәтінді жіберу қажет. Осыған қоса, мәтін разряд деңгейінде өңделсе, әр символ *8 (немесе 16)* битке ауыстырылады, ал бұл деген символдар саны *8 (немесе 16)* рет көбейетінін білдіреді. Символдардың көп санын араластыру қауіпсіздікті жоғарлатады [6, 18].

3.1. ЗАМАНАУИ БЛОКТЫ ШИФРЛАР

Симметриялық кілті бар заманауи блокты шифрлар *n*-биттік ашық мәтін блогын шифрлайды немесе шифрланған мәтіннің *n*-биттік блогын керішифрлайды. Шифрлау немесе керішифрлау алгоритмі *k*-биттік кілтті пайдаланады. Керішифрлау алгоритмі шифрлау алгоритмінің инверсиясы болу керек, жіберуші жіберген хабарды қабылдаушы қалпына келтіру үшін екеуіде жұмыста бір кілтті қолданады. 3.1 сурет заманауи блокты шифрда шифрлау және керішифрлаудың жалпы идеясын көрсетеді.

Егер хабардың өлшемі *n* биттен төмен болса, *n*-разрядты блокты құру үшін толықтыру қосу қажет; егер хабардың ұзындығы *n* биттен ұзын болса, ол *n*-разрядты блоктарға бөліну қажет, және

егер қажет болса соңғы блокқа сәйкес толықтыру қосу қажет. Жалпы мәндер *n* үшін әдетте *64*, *128*, *256* немесе *512* бит [12].



3.1 сурет – Заманауи блокты шифр

3.1 мысал. Егер шифр блок ұзындығын *64* бит қабылдаса және кодтау үшін *8* бит бойынша *ASCII* код қолданса, *100* символдан тұратын хабарға қанша қосымша бит қосу керек екенін анықтау.

Шешім. 100 символды кодтау үшін 8 бит бойынша *ASCII* кодтар қолданылады, онда хабар ұзындығы *800* бит болады. Алғашқы деректер *64* қалдықсыз бөліну қажет. Егер $|M|$ және $|Pad|$ – хабар ұзындығы және толықтыру ұзындығы болса, онда

$$(|M| + |Pad|) \bmod 64 = 0.$$

Бұл жерден

$$|Pad| = -|M| \bmod 64 = -800 \bmod 128 = -32 \bmod 64 = 32.$$

Ол деген хабарға *32* бит толықтыруды (мысалы, нөлдерді) қосу қажет екенін көрсетеді. Алғашқы деректер онда *832* биттерден немесе он үш *64*-разрядты блоктардан тұрады. Тек соңғы блокта толықтыру болады. Он үш шифрланған деректер блогын құру үшін шифратор шифрлау алгоритмін он үш рет қолданады.

3.1.1. Ауыстыру және орын ауыстыру шифрлары

Заманауи блокты шифр ауыстыру шифры немесе транспозиция (орын ауыстыру) шифры сияқты жұмыс істеу үшін жобалану

мүмкін. Бұндай идея дәстүрлі шифрларда да қолданылады, тек айырмашылығы ауыстырылатын немесе алмастырылатын символдарда символдың орнында биттер болады.

Егер шифр ауыстыру шифры сияқты жобаланса, алғашқы деректерде *1* немесе *0* биттер мәні *0* немесе *1* ауыстырылу мүмкін. Бұл алғашқы және шифрланған деректерде әртүрлі бірлер саны болу мүмкін болуын білдіреді. Мысалы, 64 биттік алғашқы деректер блогында *12* нөл және *52* бір болсын, шифланған түрде бұл блокта *34* нөл және *30* бір болу мүмкін. Егер шифр орын ауыстыру (транспозиция) шифры сияқты жобаланса, онда биттер тек кездесу ретін ғана өзгертеді, символдар көлемі өзгермейді. Кез келген жағдайда, мүмкін *n*-биттық алғашқы немесе шифрланған деректердің саны 2^n тең, себебі блокта қолданылған *n* биттің әрқайсысы *0* немесе *1* мәніне тең болу мүмкін.

Заманауи блокты шифрлар ауыстыру шифрлары сияқты жобаланған, себебі төменде көрсетілген мысалдардағыдай транспозиция қасиеттері (нөл және бір сандарының сақталауы) шифрды толық іздеу шабуылына осал болуына әкеледі.

3.2 мысал. $n = 64$ тең блокты шифр болсын. Шифрланған деректерде *10* бір болсын, ал қалған деректер - нөлдер (54 нөл). Шифрланған деректерді жолай ұстау жолымен алғашқы деректерді алу үшін қарсылас қанша *«сынау және қате»* сияқты сынауларды жасау керек әр келесі жағдайға:

а) шифр ауыстыру шифры сияқты жобаланған;

б) шифр транспозиция шифры сияқты жобаланған.

Шешім. Бірінші жағдайда (ауыстыру) қарсылас алғашқы деректерде қанша бір бар екенін білмейді. Сонда бір мәні бар блокты табу үшін ол *64* биттен барлық мүмкін 2^{64} блокты сынау керек. Егер қарсыласта секундына *1* миллиард блокты сынауға мүмкіндігі болса, онда жұмысы сәтті біту үшін оған жүздеген жыл керек болады.

Екінші жағдайда (орын ауыстыру) қарсылас алғашқы деректерде *10* бір бар екенін нақты біледі, себебі шифрланған деректерде транспозиция бірлер (немесе нөлдер) санын өзгертпейді. Қарсылас толық іздеу шабуылын тек *10* бірі бар *64*-биттік блоктарын қолданып бастау мүмкін. *64* биттен 2^{64} сөздерде тек

$$\frac{n!}{(m_1)!\cdot(n-m_1)!} = \frac{64!}{10!\cdot 54!} = 151\,473\,214\,816 \qquad (3.1)$$

нақты *10* бірі бар.

(3.1) өрнегінді m_1 – шифрланған деректерде бірлер саны.

Егер қарсылас секундына *1* миллиард сынау орындау мүмкін болса, онда оларды 3 минуттан аз уақытта тексеруге болады.

Сондықтан, толық іздеу шабуылына берік заманауи блокты шифр ауыстыру шифры сияқты жобалану керек.

3.1.2. Топтық математикалық орын ауыстыру сияқты блокты шифрлар

Заманауи блокты шифр математикалық топ болатынын анықтау қажет. Алдымен, кез келген мүмкінді алғашқы деректерді шифрлау үшін блокты шифрдың кілтінің ұзындығы жеткілікті дейік. Бұндай блокты шифрлар *толықөлшемді кілттік шифрлар* деп аталады. Тәжірибелік түрде кілт кішірек; ұзын кілтті тек кейбір шифрлауларда қолдануға болады. Жіберуші және қабылдаушы арасында алмасу кезінде блокты шифрда құпия кілт болу керек болса да, шифрда кілттен тәуелсіз компоненттер қолданылады [15, 16].

Толықөлшемді кілттік шифрлар

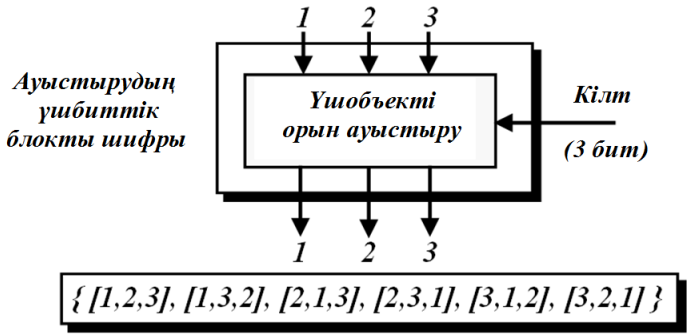
Толықөлшемді кілттік шифрлар тәжірибеде қолданбаса да, бөлшекті өлшемді кілтті шифрлар түсінікті болу үшін алдымен соларды қарастырамыз.

Толықөлшемді кілттік блокты транспозиция шифры мәндерін өзгертпей биттерді алмастырады, сондықтан ол *n!* орын ауыстыру кестелер көптігі бар орын ауыстыру шифры сияқты модельденген болу мүмкін. Бұл жерде жіберуші және қабылдаушы қай кестені қолданатынын кілт анықтайды. Ол үшін *n!* мүмкінді кілт болу керек, ал оның ұзындығы $[\log_2 n!]$ бит болу керек.

3.3 мысал. Блокты транспозиция шифры үшін үлгіні және *3* битке орын ауыстыру кестелер көптігін құру, бұл жерде блок өлшемі $– n = 3$ бит

Шешім. Орын ауыстыру кестелер көптігі 3.2 суретінде көрсетілгендей $n! = 3! = 6$ элементтен тұрады.

Кілттің ұзындығы $[\log_2 n!] = 3$ бит болу керек. Кілт *3* бит болса да, ол $2^3 = 8$ әртүрлі бейнені таңдау мүмкін, олардың ішінен *6* қолданамыз.



3.2 сурет – Блокты транспозиция шифры орын ауыстыру түрінде

Толықөлшемді кілттік блокты ауыстыру шифрлары бір қарағанда, орын ауыстыру ретінді модельденбеу мүмкін сияқты. Бірақ, егер кіріс ақпаратты керікодтап және шығыс ақпаратты кодтаса, онда ауыстыру шифры үшін де орын ауыстыру үлгісін қолдануға болады. Керікодтау бұл жерде ол *n*-разрядты тұтас санды бір бірі және 2^n*-1* нөлі бар 2^n-биттік жолға түрлендіру [4, 32]. Бір бірдің позициясы *0* ден 2^n*-1* дейінгі реттелген позициялар тізбегінде тұтас санның мәнін көрсетеді. Жаңа кіріс ақпаратта әрдайым бір бір болғандықтан, шифрды $2^n!$ объекттер орын ауыстыруы ретінде модельдеуге болады.

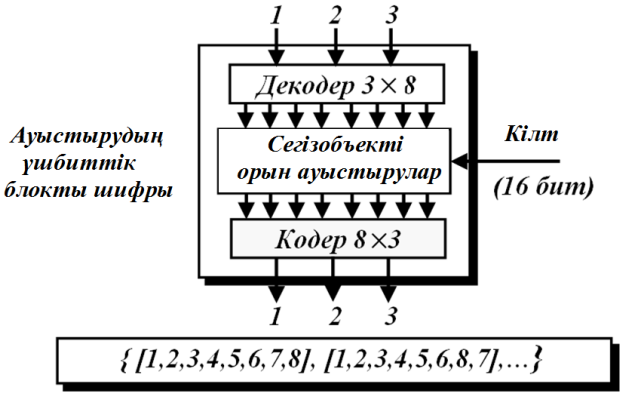
3.4 мысал. Блокты ауыстыру шифры үшін *3* битке үлгі және орын ауыстыру кестелер көптігін құру.

Шешім. Үш алғашқы кіріс деректер блоктары *0* ден *7* дейін тұтас сандармен белгілену мүмкін. Оларды бір бірі бар *8* биттік жол ретінде кодтауға болады. Мысалы, *000* комбинациясы *00000001* (оң жақтан бірінші бір) ретінде кодталу мүмкін; *101* комбинациясы *00100000* (оң жақтан алтыншы бір) ретінде. Үлгіні және орын ауыстыру кестелер көптігін 3.3 сурет көрсетеді.

Кодталған көптікте элементтер саны транспозиция шифрындағы элементтер санынан (*8! = = 40 320*) өте көп. Кілт ұзындығы да үлкенрек $[\log_2 40\,320] = 16$ бит. *16* биттік кілт *65 536* әртүрлі бейнелерді анықтау мүмкін болса да тек *40 320* қолданылады.

Толықөлшемді кілт – бұл *n*-разрядты транспозиция шифры немесе блокты ауыстыру шифры. Ол орын ауыстыру шифры сияқты модельдену мүмкін, бірақ олардың кілт көлемдері әртүрлі:

транпозиция шифры үшін - $[\log_2 n!]$, ауыстыру шифры үшін - $[\log_2 (2n)!]$.



3.3 сурет – Блокты ауыстыру шифрын орын ауыстыру шифры ретінде модельдеу

Толықөлшемді кілттік транспозиция немесе ауыстыру/орын ауыстыру шифры келесіні көрсетеді: егер шифрлау немесе керішифрлау осы шифрлардың кез келген комбинациясының бірден көбін қолданса, онда нәтиже топтық орын ауыстыру операциясына эквивалентті болады. Екі немесе одан көп каскадты орын ауыстыру бір орын ауыстырумен ауыстырылу мүмкін екені белгілі [43]. Бұл деген толықөлшемді кілттік шифрлардың бір каскадынан көп қолдану тиімсіз екенін көрсетеді, себебі ол бір қадам сияқты әсер қалдырады.

Бөлшекөлшемді шифр кілттері

Шын мәнде шифрлар толықөлшемді кілттерді қолдана алмайды, себебі кілт өлшемі өте үлкен болады, әсіресе блокты ауыстыру шифры үшін. Мысалы, *DES* жалпы ауыстыру шифры *64*-разрядты блокты шифрды қолданады. Егер *DES* жобалаушылары толықөлшемді кілтті қолданса, ол $\log_2(2^{64}!) = 2^{2^{70}}$ бит болатын еді. Тәжірибеде *DES* үшін кілт тек *56* бит, бұл толықөлшемді кілттің өте

қысқа үзіндісі. Бұл *DES* $2^{2^{70}}$ мүмкін бейнелерден тек 2^{56} қолданатынын көрсетеді.

Өзімізге сұрақ қояйық: бөлшекті кілті бар көпқадамды транспозиция немесе ауыстыру бұл операциялар композициясы бар орын ауыстыру тобы екенін айтуға болама? Бұл сұраққа жауап өте маңызды, себебі ол бөлшекті кілтпен көпқадамды нұсқа шифр сияқты шифрлау құралы болу мүмкінін анықтайды. Бұл факт қауіпсіздіктің үлкенрек деңгейіне жетуге мүмкіндік береді [19].

Бөлшекті кілтті шифр – бұл шифрдың сәйкес кілт өлшемінің (толықөлшемді) ішкі тобы. Басқа сөзбен, егер толықөлшемді кілттік шифр – топ $G = <Q, O>$ болса, бұл жерде Q – бейнелер көптігі және O – операциялар композициясы, онда бөлшекті өлшемді кілті бар шифр ішкі топ $H = = <U, O>$ болады, бұл жерде U – сол операциялары бар Q ішкікөптігі.

Мысалы, *56*-биттік кілті бар көпқадамды *DES* топ болмайтыны дәлелденген, себебі 2^{56} бейнесі бар ішкі топ $2^{64}!$ бейнесі бар топтан құрылу мүмкін емес.

Егер ол сәйкес толықөлшемді кілтті шифрдың ішкі тобы болса, онда *бөлшекті кілтті шифр* операциялар жинағы бар топ болады.

Кілтсіз шифр

Шифрды кілтсіз бөлек қолдану тиімсіз, оларды тек кілттік шифрлар компоненті ретінде қолдану мүмкін.

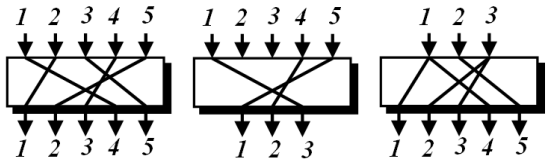
Кілтсіз транспозция шифрын (немесе тіркелген кілтпен) аппараттық құралдармен жүзеге асырылған транспозиция шифры ретінде қарастыруға болады. Тіркелген кілт (ауыстырудың жалғыз ережесі) бағдарламалық қамтамада жүзеге асырған жағдайда кесте ретінде көрсетілуі мүмкін. Ары қарай *орын ауыстырудың P-блоктары* деп аталатын кілтсіз транспозиция шифрлары қарастырылады, олар заманауи блокты шифрлардың стандартты блоктары ретінде қолданылады.

Кілтсіз ауыстыру шифрларын (немесе тіркелген кілтті) кіріс ақпаратты шығысқа алдын ала анықталған бейнелеу ретінде қарастыруға болады. Бейнелеу кесте, математикалық функция және басқа тәсілдерімен көрсетілу мүмкін. Ары қарай *ауыстырудың S-блоктары* деп аталатын кілтсіз ауыстыру шифрлары қарастырылады, олар заманауи блокты шифрлардың стандартты блоктары ретінде қолданылады.

3.1.3. Заманауи блокты шифрдың компоненттері

Заманауи блокты шифрлар әдетте ауыстырудың кілтті шифрлары болады, оларда кілт тек мүмкін кіріс ақпаратты шығыс ақпаратқа бөлшекті бейнелеуге мүмкіндік береді. Бірақ бұл шифрлар бір модуль ретінде жобаланбайды. Заманауи блокты шифрдың ақпаратты шашырату және аларастыру сияқты талап етілетін қасиеттерді қамтамасыз ету үшін, бұл шифр транспозиция (орын ауыстырудың *P*-блоктары деп аталатын) модульдері мен ауыстыру (ауыстырудың *S*-блоктары деп аталатын) модульдері және кейбір басқа модульдерінің комбинациясы ретінде қалыптастырылады.

Орын ауыстырудың *P*-блогы символдарды транспозициялаудың дәстүрлі шифрына ұқсас. Ол биттерді алмастырады. Заманауи блокты шифрларда орын ауыстырудың *P*-блоктарының үш типын табуға болады: түзу *P*-блоктары; кеңейту *P*-блоктары және қысу *P*-блоктары, бұл 3.4 суретінде көрсетілген.



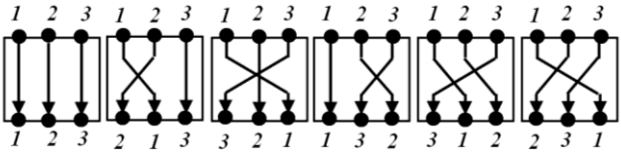
3.4 сурет - *P*-блоктардың үш типі: а) түзу *P*-блок;
б) қысу *P*-блогы; в) кеңейту *P*-блогы

3.4 сурет түзу *P*-блоты *5×5* (3.4, а сурет), қысу *P*-блогын *5×3* (3.4, б сурет) және кеңейту *P*-блогын *3×5* (3.4, в сурет) көрсетеді. Әрқайсысын тереңрек қарастырамыз.

Түзу P-блоктар

n кірісі және *n* шығысы бар *түзу P-блок* – бұл *n!* мүмкінді бейнесі бар орын ауыстыру.

3.5 мысал. 3.5 сурет *3×3* түзу *P*-блоктың барлық алты мүмкінді бейнесін көрсетеді.



3.5 сурет - 3×3 түзу *P*-блоктың мүмкінді бейнесі

Түзу *P*-блок *n!* бейненің біреуін анықтау үшін кілтті қолдану мүмкін, әдетте түзу *P*-блоктар – кілтті қолданусыз, яғни бейне алдын ала белгілі. Егер түзу *P*-блок алдын ала берілген және аппаратты құралдармен жүзеге асырылған болса, немесе егер ол бағдарламалық құралдармен жүзеге асырылса, онда орын ауыстыру кестелері бейнелеу ережесін анықтайды. Екінші жағдайда кесте кірістері шығыс позициялары көрсетілген позицияларға көрсетеді. 3.1 кесте $n=32$ жағдайда орын ауыстыру кестесінің мысалын береді.

3.1 кестеде *32* кестелік кіріс бар, олар *32* ақпараттық кіріске сәйкестікті тіркейді. Кіріс позициясы (индекс) шығысқа сәйкес. Мысалы, бірінші кестелік кірісте *26* нөмірі бар. Бұл бірінші шығыс *26*-шы кіріске сәйкес келетінін білдіреді. Сонғы кестелік кіріс – *7* деген *32*-ші шығыс жетінші ақпараттық кіріске сәйкес болатынын білдіреді, және ары қарай.

3.1 кесте

Түзу *P*-блок үшін орын ауыстыру кесте мысалы

	Биттер нөмірлері															
Кіріс	*26*	*18*	*10*	*02*	*28*	*20*	*12*	*04*	*30*	*22*	*14*	*06*	*32*	*24*	*16*	*08*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Кіріс	*25*	*17*	*09*	*01*	*27*	*19*	*11*	*03*	*29*	*21*	*13*	*05*	*31*	*23*	*15*	*07*
Шығыс	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*

Түзу *P*-блок қайтымды болады. Бұл түзу *P*-блокты шифрлау үшін және содан кейін керішифрлау үшін қолдануға болатынын білдіреді. Орын ауыстыру кестелері бір біріне қатысты қайтымды болу керек.

3.6 мысал. Кіріс сөздің екі орташа битын (*4* және *5* биттер) шығыс сөзінің шеткі биттеріне (*1* және *8* биттер) жылжытатын 8×8 түзі *P*-блок үшін орын ауыстыру кестесін құру қажет. Басқа биттерінің жанама позициялары өзгермейді.

Шешім. Түзу *P*-блокты *[4, 1, 2, 3, 6, 7, 8, 5]* кестемен құру қажет. *1, 2, 3, 6, 7* және *8* биттердің жанама позициялары өзгермейді, бірақ бірінші ақпараттық шығыс төртінші ақпараттық кіріспен

байланысты, сегізінші ақпараттық шығыс - бесінші ақпараттық кіріспен. Осындай түзу *P*-блок үшін орын ауыстыру 3.2 кестесінде келтірілген.

3.2 кесте

3.6 мысал үшін түзу *P*-блок үшін орын ауыстыру кестесі

	Биттер нөмірі							
Кіріс	*04*	*01*	*02*	*03*	*06*	*07*	*08*	*05*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*

Қысудың P-блоктары

Қысудың *P*-блогы деген *n* кірісі, *m* шығысы бар және $m < n$ *P*-блогы. Ақпараттық кіристердің кейбіреуі бұғатталған және шығыспен байланыспаған (3.4, б суретті қараңыз). Заманауи блокты шифрлардағы қысудың *P*-блоктары әдетте орын ауыстыру кестесі биттерді орын ауыстыру ережелерін көрсететін кілтсіз болады. Қысудың *P*-блогы үшін орын ауыстыру кестесінде *1* ден *n*-ға дейін *n* кестелік кірісі болады, бірақ әр кестелік шығыс мазмұнында кейбіреуі болмау мүмкін (бұғатталған ақпараттық кірістер) соны ескеру қажет. 3.3 кестесі 32×24 қысудың *P*-блогы үшін *7, 8, 9, 16, 23, 24* және *25* кірістері бұғатталған орын ауыстыру кесте мысалын көрсетеді.

Қысудың *P*-блоктары биттерді орын ауыстыру және сол уақытта келесі кезең үшін бит сандарын азайту үшін қолданылады.

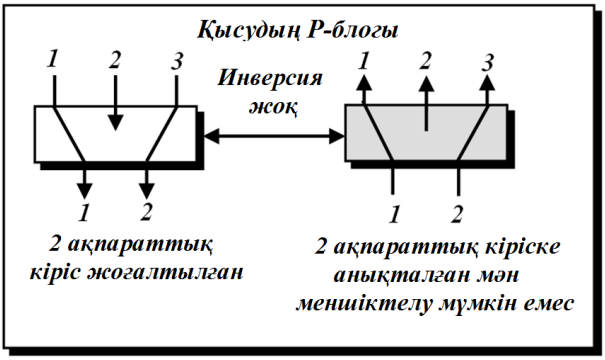
Қысудың *P*-блоктары қайтымды емес. Қысудың *P*-блоктарында бөлек кірістер шифрлау процессінде алып тасталуы мүмкін, бірақ керішифрлау алгоритмде алып тасталған битті қалпына келтіру үшін кілті жоқ.

3.3 кесте

32×24 қысу *P*-блогы үшін орын ауыстыру кесте мысалы

	Биттер нөмірі											
Кіріс	*01*	*02*	*03*	*21*	*22*	*26*	*27*	*28*	*29*	*13*	*14*	*17*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*
Кіріс	*18*	*19*	*20*	*04*	*05*	*06*	*10*	*11*	*12*	*30*	*31*	*32*
Шығыс	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*

3.6 сурет қысудың *P*-блогын шифрлау және керішифрлау үшін қолдану жағдайын көрсетеді.



3.6 сурет – Қайтымды емес компонент ретінде қысудың *P*-блогы

Кеңейтудің P-блогы

Кеңейтудің P-блогы деген *n* кірісі, *m* шығысы бар және $m < n$ *P*-блогы. Кейбір кірістер бірден көп шығыспен байланысқан (3.4, в суретін қараңыз). Заманауи блокты шифрларда қолданатын кеңейту *P*-блоктары әдетте кілтсіз. Биттерді орын ауыстыру ережелері кестеде көрсетіледі. Кеңейтудің *P*–блогы үшін орын ауыстыру кестесінде *m* кестелік шығыс болады, бірақ *m*–*n* кіріс (бірден көп ақпараттық шығыспен байланысқан кірістер) болады. 3.4 кестеде 12×16 кеңейтудің *P*–блогы үшін орын ауыстыру кестесінің мысалы көрсетілген. Әр *1, 3, 9* және *12* кіріс екі шығыспен байланысқанына назар аударыңыз.

Кеңейтудің *P*–блоктары биттердің орнын ауыстыру және сол уақытта шифрлаудың келесі каскады үшін биттер санын үлкейту үшін қолданылады.

3.4 кесте

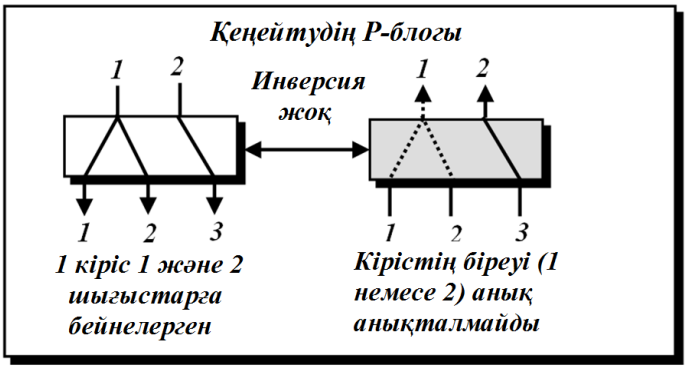
12× 16 кеңейту *P*-блогы үшін орын ауыстыру кестесінің мысалы

	Бит нөмірлері															
Кіріс	*01*	*09*	*10*	*11*	*12*	*01*	*02*	*03*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*12*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*

3.7 мысал. 3.7 суреті бірөлшемді кесте жағдайында орын ауыстыру кестесін қалай өзгерту керек екенін көрсетеді.

3.7 сурет – Орын ауыстыру кестесін өзгерту

Кеңейтудің *P*-блоктары да қайтымды емес. Кеңейтудің *P*-блогында шифрлау процессінде кейбір кірістер бірден көп шығысқа бейнелену мүмкін, бірақ шифрлау алгоритмінде кілт жоқ және ол өзі қай кірістер берілген шығыста бейнеленгенін анықтай алмайды. 3.8 сурет кеңейтудің *P*-блогын шифрлау және керішифрлау үшін қолдану жағдайын көрсетеді.



3.8 сурет – Кеңейтудің *P*-блогы қайтымды емес компонент ретінде

3.6 сурет және 3.8 сурет қысудың *P*-блогы кеңейтудің *P*-блогына қайтымды болмайтынын және керісіншені көрсетеді. Бұл деген, егер қысудың *P*-блогын шифрлау үшін қолданса, онда кеңейтудің *P*-блогын керішифрлау үшін қолдануға болмайтынын және керісіншені білдіреді. Бірақ, төменде көрсетілетіндей,

шифрлау үшін қысу немесе кеңейту *P*-блоктарын қолданатын шифрлар бар; бірақ олардың тиімділігі басқа тәсілдерге қарағанда нашарлау.

Ауыстырудың S-блоктары

Ауыстырудың S-блогын ауыстырудың кішкентай шифры ретінде қарастыруға болады. Бұл блокта кіріс және шығыс сандары әртүрлі болу мүмкін. Басқа сөзбен, ауыстырудың *S*-блогына кіріс *n*-битті сөз болу мүмкін, ал шығыс *m* разрядты сөз болу мүмкін, бұл жерде *m* және *n* бірдей болуы міндетті емес. Ауыстырудың *S*-блогы кілтті немесе кілтсіз болу мүмкін. Заманауи блокты шифрлар әдетте ауыстырудың *S*-блогын кілтсіз қолданады, бұл жерде ақпараттық кірістерден бейнелеу ақпараттық шығыстарға алдынала анықталған.

Ауыстырудың *S*-блогы – $m \times n$ ауыстыру модулі, бұл жерде *m* және *n* бірдей болу міндетті емес.

Ауыстырудың *S*-блогында *n* кірісі және *m* шығысы бар, кірістерді x_1, x_2,…, x_n деп және шығыстарды y_1, y_2,..., y_m деп белгілейміз. Кіріспен шығыс арасындағы қатынастар теңдеулер жүйесі ретінде көрсетілу мүмкін

$$y_1 = f_1 (x_1,x_2,\ldots,x_n);$$
$$y_2 = f_2 (x_1,x_2,\ldots,x_n);$$
$$\ldots$$
$$y_m = f_m (x_1,x_2,\ldots,x_n).$$

Ауыстырудың сызықты *S*-блогында жоғарыда көрсетілген арақатынастар келесідей болу мүмкін:

$$y_1 = a_{1,1} \cdot x_1 \oplus a_{1,2} \cdot x_2 \oplus \ldots \oplus a_{1,n} \cdot x_n;$$
$$y_2 = a2,1 \cdot x_1 \oplus a_{2,2} \cdot x_2 \oplus \ldots \oplus a_{2,n} \cdot x_n;$$
$$\ldots$$
$$y_m = a_{m,1} \cdot x_1 \oplus a_{m,2} \cdot x_2 \oplus \ldots \oplus a_{m,n} \cdot x_n.$$

Ауыстырудың сызықты емес *S*-блогында әр шығыс үшін жоғарыда көрсетілген арақатынастарды әрқашан беруге болады.

3.8 мысал. Үш кірісі және екі шығысы бар ауыстырудың *S*-блогында келесі болсын

$$y_1 = x_1 \oplus x_2 \oplus x_2 \oplus x_3; \quad y_2 = x_1.$$

Ауыстырудың *S*-блогы сызықты, себебі

$$a_{1,1} = a_{1,2} = a_{1,3} = a_{2,1} = 1 \quad \text{і} \quad a_{2,2} = a_{2,3} = 0.$$

Бұл арақатынастар төменде көрсетілгендей матрицалармен берілу мүмкін:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 111 \\ 100 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

3.9 мысал. Үш кірісі және екі шығысы бар ауыстырудың *S*-блогында болсын

$$y_1 = (x_1)^3 + x_2;$$

$$y_2 = (x_1)^2 + x_1 \cdot x_2 + x_3,$$

бұл жерде көбейту және қосу *Галуа GF(2)* өрісінде жүргізіледі. Ауыстырудың *S*-блогы сызықты емес, себебі кірістер және шығыстар арасында сызықты арақатынастар жоқ.

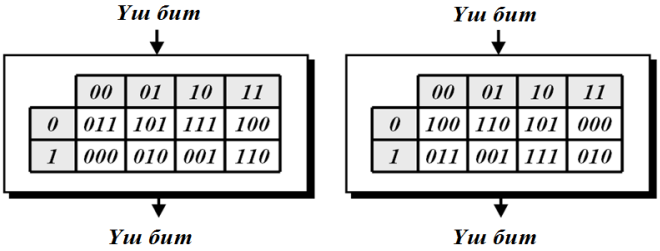
3.10 мысал. Келесі кесте (3.9 суретке қараңыз) 3×2 көлемді ауыстыру *S*-блогы үшін кірістер/шығыстар арасында қатынастарды анықтайды. Кірістің шеткі сол жақтағы биты жолды анықтайды; кірістің ең оң жақтағы екі биты бағанды анықтайды. Шығыстың екі биты – таңдалған жол және баған қиылысындағы мән.

3.9 сурет - 3.10 мысалы үшін ауыстырудың *S*-блогының кестесі

Ауыстырудың S-блогы – кіріс және шығыс арасындағы қатынастары кестемен немесе математикалық арақатынаспен анықталатын ауыстыру шифрлары. *S*-блогы қайтымды болу немесе

болмау мүмкін. Қайтымды ауыстыру *S*-блогында кіріс биттер саны шығыс биттер санына тең болу қажет.

3.11 мысал. 3.10 сурет қайтымды ауыстыру *S*-блогының мысалын көрсетеді. Кестенің біреуі шифрлау алгоритмінде қолданылады; екіншісі – керішифрлау алгоритмінде.



3.10 сурет – 3.11 мысал үшін ауыстыру *S*-блогының кестелері

Әр кестеде кірістің шеткі сол жақтағы биты жолды анықтайды; келесі екі бит бағанды анықтайды. Шығыс – кестенің жол және баған қыйылысындағы мән.

Мысалы, егер сол жақтағы блокқа кіріс – *001*, онда шығыс – *101*. Оң жақтағы кестедегі кіріс *101* шығыста *001* береді. Бұл екі кесте бір біріне қатысты қайтымды нәтиже алуға мүмкіндік беретінін көрсетеді.

Аластамалы НЕМЕСЕ

Шифрлау блоктарының көбісінде маңызды компонент - *аластамалы немесе (xor). Галуа GF(2^n)* өрісінде қосу және алу операциялары бір операциямен орындалады *аластамалы немесе (xor)* деп аталатын.

Галуа GF(2^n) өрісінде *xor* операцияның бес қасиеті бұл операцияны блокты шифрда қолдануын өте ыңғайлы қылады:

1. *Тұйықтық.* Бұл қасиет осы операция нәтижесінде екі *n*-биттік сөз басқа *n*-биттік сөзді беретінін кепілдейді.

2. *Ассоциативтік.* Бұл қасиет бірден көп *xor* қолдануға мүмкіндік береді, оларды кез келген ретпен есептеуге болады.

$$x \oplus (y \oplus z) \leftrightarrow (x \oplus y) \oplus z.$$

3. *Коммутативтік.* Бұл қасиет нәтижені (шығыс ақпаратты) өзгертпей, операторларды (кіріс ақпаратты) орындармен ауыстыруға мүмкіндік береді.

$$x \oplus y \leftrightarrow y \oplus x.$$

4. *Нөлдік (ұқсас) элементтің бар болуы. Xor* операциясы үшін нөлдік элемент – барлық нөлдерден тұратын сөз немесе (*00...0*). Бұны бейтарап элементтері бар сөз бар екенін түсінеді, ол операцияны орындау кезде сөзді өзгертпейді.

$$x \oplus (00\ldots00) = x.$$

Бұл қасиет *Фейстель* шифрында қолданылады, оны арықарай қарастырамыз.

5. *Инверсияның болуы. Галуа GF(2^n)* өрісінде әр сөз өзінің аддитивті инверсиясы болып табылады. Бұл сөздің өзімен өзін *xor* операциясын орындау нөлдік элементке алып келеді:

$$x \oplus x = (00\ldots00).$$

Бұл қасиетті *Фейстель* шифрында қолданамыз, оны арықарай қарастырамыз.

6. *Толықтыру.* Толықтыру операциясы – бірорынды операция (бір ақпараттық кіріс және бір ақпараттық шығыс), ол сөздің әр битын инверсиялайды. Нөлдік бит бірлік битке өзгереді, ал бірлік бит – нөлдік битке.

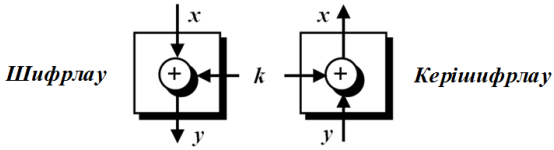
Шифрлау кезінде *xor* операциясына жанама толықтыруы бар операциялар қызығушылықты тұдырады. Егер $\overline{x}$ – x толықтыруы болса, онда келесі екі арақатынастар дұрыс болады:

$$x \oplus \overline{x} = (11\ldots11) \quad \text{и} \quad x \oplus (11\ldots11) = \overline{x}.$$

Бұл қасиетті кейбір шифрлардың қауіпсіздігін қарастырған кезде қолданамыз.

7. *Инверсия.* Егер компонент бірорынды операция болса (бір кіріс және бір шығыс), онда шифрда компонент инверсиясының мағынасы бар. Мысалы, кілтсіз *P*-блок немесе кілтсіз *S*-блок қайтымды болу мүмкін, сондықтан оларда бір кіріс және бір шығыс болады (3.11 сурет).

Ерекшелейтін немесе (*xor*) операциясы – бинарлық операция. *Xor* операцияның инверсиясы мағыналы болады тек егер кірістің біреуі тіркелген болса (шифрлау және керішифрлау кезінде өзгермейтін). Мысалы, егер кірістің біреуі әдетте шифрлау және керішифрлау кезінде өзгермейтін кілт болса, онда *xor* операциясы, 3.11 суретінде көрсетілгендей, қайтымды болады.



3.11 сурет – *Xor* операциясының қайтымдығы

3.11 суретінде аддитивті инверсия қасиеті келесіні түсінеді

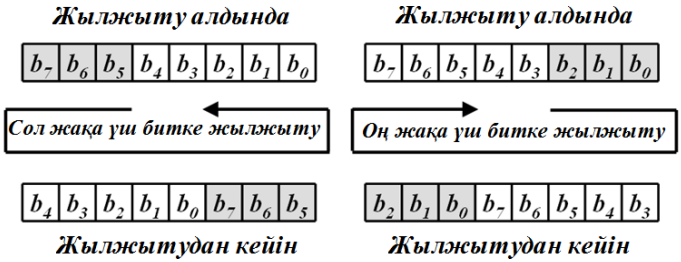
$$y = x \oplus k \quad \text{и} \quad x = k \oplus y.$$

Бұл қасиетті блокты шифрлардың құрылымын қарастырған кезде қолданамыз.

Циклды жылжыту

Кейбір заманауи блокты шифрларды қолданатын келесі компонент – *циклды жылжыту операциясы*. Жылжыту сол жақа және оң жақа болу мүмкін. Сол жақтың айналма жылжыту операциясы *n*-биттік сөзде әр битты *k* позицияға солға жылжытады; шеткі сол жақтағы *k*-биттер сол жақтан жойылады және оң жақа жазылады. Оң жақтың айналма жылжыту операциясы *n*-биттік сөзде әр битты *k* позицияға оңға жылжытады; шеткі оң жақтағы *k*-биттер оң жақтан жойылады және сол жақа жазылады. 3.12 сурет $n = 8$ және $k = 3$ болған жағдайда оң жақты және сол жақты жылжыту операцияларын көрсетеді.

Жылжытудың циклды операциясы сөзде биттерді араластырады және алғашқы сөзде үлгілерді жасыруға көмектеседі. Биттер жылжыйтын позициялар саны кілт ретінде қолданылуы мүмкін болса да, жылжытудың циклды операциясы әдетте кілтсіз; *k* мәні алдынала беріледі және орнатылады.



3.12 сурет – *8* биттік сөзді солға немесе оңға циклды жылжыту мысалы

Сол жақты жылжытудың циклды операциясы – оң жақтағы жылжыту операцияның инверсиясы, яғни қайтымды болады. Егер олардың біреуі шифрлау үшін қолданылса, онда екіншісі керішифрлауға қолданылу мүмкін.

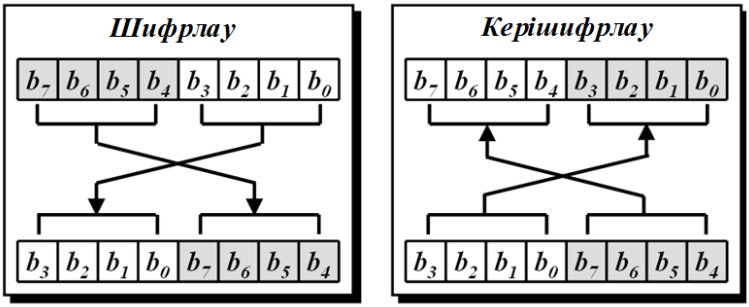
Циклды жылжыту операциясында екі қасиет бар.

Бірінші - *n* модулі бойынша жылжыту. Басқа сөзбен, егер $k = 0$ немесе $k = n$ болса, ешқандай жылжу болмайды. Егер $k > n$ болса, онда кіріс ақпарат $k \bmod n$ битке жылжыйды.

Екінші қасиет – операцияларды жалғауға циклды жылжыту операциясы – топ болады. Бұл, егер жылжыту бірнеше рет жүргізілсе, онда бір мән бірнеше рет шығу мүмкін екенін білдіреді.

Ауыстыру

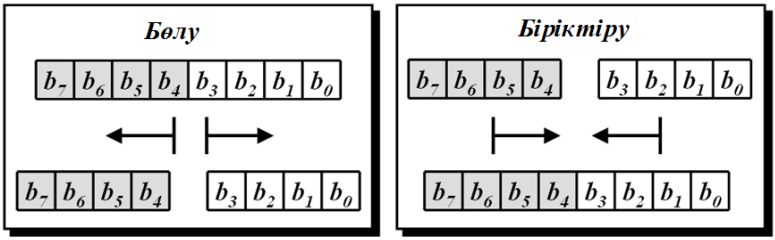
Ауыстыру операциясы – циклдық жылжыту операциясының арнайы жағдайы. Бұл жерде $k = n/2$ операция мүмкін тек егер n жұп нөмір болғанда. Сол жақа $n/2$ жылжыту оң жақа $n/2$ жылжытумен бірдей болғандықтан, бұл операция қайтымды болады. Шифрлау үшін ауыстыру операциясы керішифрлау кезіндегі ауыстыру операциясымен толық ашылу мүмкін. 3.13 суреті *8* биттен тұратын сөз үшін ауыстыру операциясын көрсетеді.



3.13 сурет - *8* биттік сөзде ауыстыру операциясы

Бөлу және біріктіру

Кейбір блокты шифрларда қолданылатын операциялар – *бөлу* және *біріктіру*. Бөлу әдетте *n*-биттік сөзді ортасынан бөледі, бірдей ұзындығы бар екі сөзді құрып. Біріктіру *n*-биттік сөзді құру үшін ұзындығы бірдей екі сөзді байланыстырады. Бұл екі операция бір біріне инверсты және бір бірін теңестіру үшін жұп ретінде қолдану мүмкін. Егер біреуі шифрлау үшін қолданылса, онда екіншісі – керішифрлау үшін. 3.14 сурет бұл екі операцияны *n = 8* жағдай үшін көрсетеді.



3.14 сурет - *8*-биттік сөз үшін бөлу және біріктіру операциялары

3.2. ҚҰРАМДАС ШИФРЛАР

Шеннон құрамдас шифрлар түсінігін енгізді. *Құрамдас шифр* – ауыструды, орын ауыстыруды және жоғарыда қарастырылған басқа компоненттерін біріктіретін кешен.

3.2.1. Шашырату және араластыру

Құрамдас шифрды ұсынуда *Шеннонның* идеясы блокты шифрларға келесі екі маңызды қасиетке ие болуына мүмкіндік беру: *шашырату* және *араластыру.*

Шашырату алғашқы және шифрланған деректер арасында қатынастарды жасыру керек. Бұл алғашқы деректерді табу үшін шифрланған деректер статистикасын қолданатын қасқойды *«есінен жаңылдырады»*. Шашырату шифрланған деректерде әр бит (символ) алғашқы деректерде бір немесе барлық биттерден (символдардан) тәуелді болатынын түсінеді. Басқа сөзбен, егер алғашқы деректерде бір бит өзгерсе, онда шифрланған деректерде бірнеше немесе барлық биттер өзгереді.

Араластырудың идеясы келесіде: ол шифрланған деректер және кілт арасында тәуелдікті жасыру. Бұл да кілтті табу үшін шифрланған деректерді қолдануды тырысатын қасқойды *«есінен жаңылдырады»*. Басқа сөзбен, егер кілтте бір бит өзгерсе, онда шифрланған деректерде барлық биттер өзгереді.

3.2.2. Раундтар

Шашырату және араластыруға құрамдас шифрларды қолдану арқылы жетуге болады, бұл жерде әр итерация деген *S*-блоктар, *P*-блоктар және басқа компоненттер комбинациясы. Әр итерация *раунд* деп аталады. Блокты шифр шифр кілтінен әр раунд үшін әртүрлі кілттерді құратын кілттік тізбекті немесе кілттер генераторын қолданады. *N*-раундтық шифрда шифрланған деректерді құру үшін алғашқы деректер *N* рет шифрланады; сәйкес, шифрланған деректер *N* рет керішифрланады. Аралық деңгейде (екі раунд ортасында) құрылған деректер *орташа деректер* деп аталады. 3.15 сурет екі раунды бар қарапайым құрамдас шифрды көрсетеді.

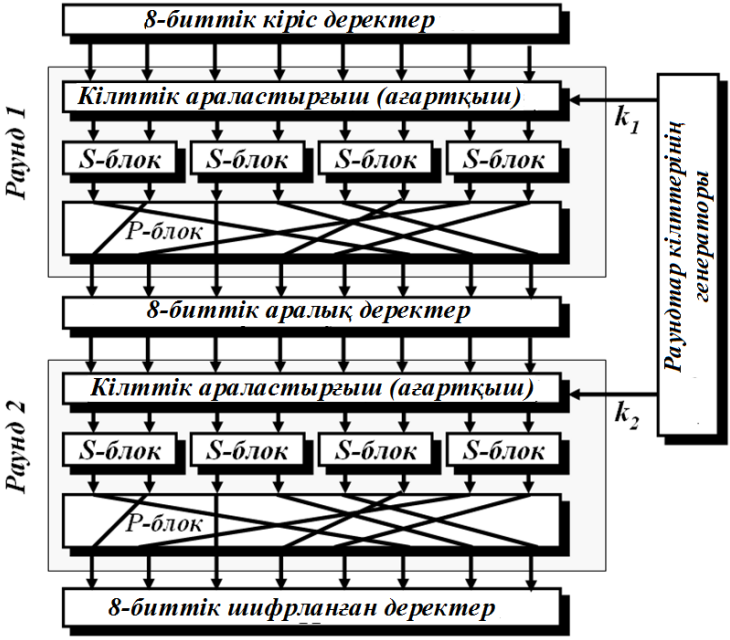
Тәжірибеде құрамдас шифрларда екіден көп раунд бар. 3.15 суретінде әр раундта үш түрлендіру өткізіледі:

1. Деректердің биттері теңықтималдықты болу үшін *8*-биттік деректер раундтық кілтпен араластырылады (кілтті қолданып, биттерді жасыру) – деректер *«ағартылады»* (*whiting*). Бұл әдетте *8* биттік сөзді *8* биттік раундты кілтті *xor* операция көмегімен өңдеу арқылы жасалады.

2. *«Ағартушы»* шығыстарынан деректер *2* биттік төрт топқа бөлінеді және төрт ауыстыру *S*-блогына жіберіледі. Биттер мәндері бұл түрлендіруде ауыстырудың *S*-блогын құруына сәйкес өзгереді.

3. Ауыстырудың *S*-блогының шығыстарынан деректер орын ауыстырудың *P*-блогына кіреді, сонымен бірге биттер келесі шартпен алмасу керек: келесі раундта әр блок шығысынан нәтиже әртүрлі кірістерге кіру қажет.

3.15 суретінде сұлбасы көрсетілген жеңілдетілген құрамдас шифр ауыстырудың *S*-блогы және орын ауыстырудың *P*-блогы комбинациясын қолданып шашыратуды кепілдейді.



3.15 сурет – Екі раунды бар құрамдас шифр

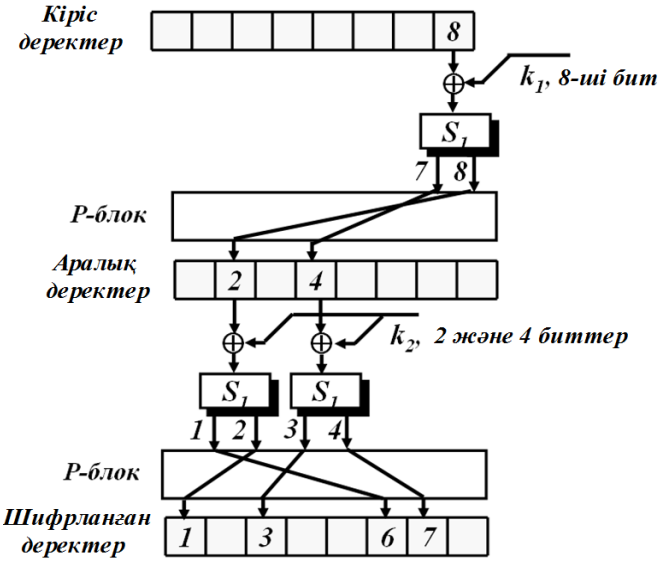
Бірінші раундта, k_1 раундтық кілттің сәйкес биттерімен *xor* операциясын өткізгеннең кейін, *S*-блок *4* арқылы екі бит өзгереді (*7* және *8* биттер). *7* бит орын ауыстырып *2* бит болады; *8* бит орын ауыстырып *4* бит болады. Бірінші раундтан кейін *8* бит *2* және *4* биттерді өзгертеді. Екінші раундта, k_2 раундтық кілттің сәйкес биттерімен *xor* операциясын өткізгеннең кейін, *S*-блок *1* арқылы екі бит өзгереді (*1* және *2* биттер). *1* бит орын ауыстырып *6* бит болады;

2 бит орын ауыстырып *1* бит болады. k_2 раундтық кілттің сәйкес биттерімен *xor* операциясын өткізгеннең кейін, *4* бит *3* және *4* биттерді өзгертеді. *3* бит қалады, *4* бит орын ауыстырып *7* бит болады. Екінші раундтан кейін *8* биттен *1, 3, 6* және *7* биттері өзгереді.

Бұл қадамдарды басқа бағытта орындау (шифрланған деректерден алғашқы деректерге дейін) шифрланған деректерде әр бит алғашқы деректерді бірнеше биттерге өзгертетінін көрсетеді.

3.16 суретінде алғашқы деректерде жалғыз бір битті өзгертуі шифрланған деректерде көптеген биттерді өзгертуді шақыратынын көрсетеді.

3.16 сурет араластыру қасиетіне құрамдас шифрдың көмегімен жетуге болатынын дәлелдейді. Шифрланған деректердің төрт биты (*1, 3, 6* және *7* биттер) раундтық кілттерде үш бит көмегімен түрлендірілген (k_1 - *8* бит және k_2 - *2* және *4* биттер). Қайта бағытта жүру әр раундта раундтық кілттің әр биты шифрланған деректердің бірнеше битын қозғалтатынын көрсетеді. Шифрланған деректер биттері және раундтық кілттер биттері арасындағы қатынастар көленкеленген тікбұрыштарда көрсетілген.



3.16 сурет – Блокты шифрда шашырату және араластыру

Шашырату мен араластыруды жақсарту үшін тәжрібелік шифрлар деректердің ірі блоктарын, көбірек ауыстырудың *S*-блоктарын және көбірек раундтарды қолданады. Ауыстырудың *S*-блоктарының үлкен санын қолдану кезінде раундтар санын кейбір үлкейту жақсырақ шифрды құруға мүмкіндік береді, оның ішінде шифрланған деректер кездейсоқ *n*-биттік сөз сияқты көрінеді. Сонымен, шифрланған және алғашқы деректер арасындағы қатынастар толық жасырылады (шашыратылады). Раундтар санын үлкейту раундтық кілттер санын үлкейтеді, бұл шифрланған деректер және кілт арасындағы қатынастарды жасырады.

3.2.3. Құрамдас шифрлардың екі класы

Заманауи блокты шифрлар – бәрі құрамдас, бірақ олар екі класқа бөлінеді. *Бірішнші кластың шифрлары* қайтымды және қайтымды емес компоненттерді қолданады. Бұл шифрлар әдетте *Фейстель шифрлары* деп аталады. *Екінші кластағы шифрлар* тек қайтымды компонентерді қолданады. Бұл шифрлар *Фейстельдікі-емес* деп аталады (басқа атау болмағандықтан).

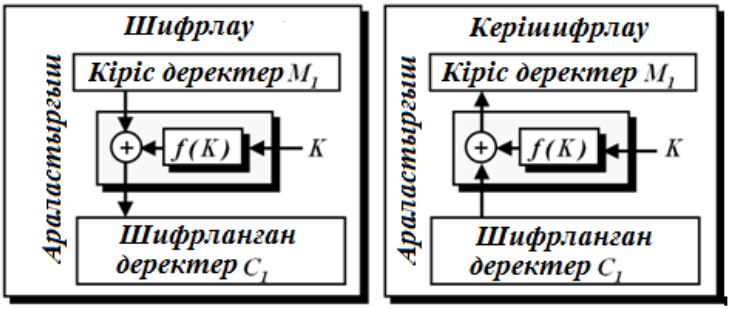
Фейстель өте зияткерлік және қызық шифрды құрды, ол көптеген жылдары қолданылды. *Фейстель* шифрында үш түрлі компонент болу мүмкін: *өзіқайтымды, қайтымды* және *қайтымды емес.*

Фейстель шифрында блоктарда барлық қайтымды емес элементтер бар және ол шифрлау және керішифрлау алгоритмдерде бір модульді қолданады. Сұрақ келесіде: егер шифрлау және керішифрлауда қайтымды емес модульдер болса, олар қалай ашық және жабық деректерді бір біріне инвертациялайды. *Фейстель* олар үйлестірілген болу мүмкін екенін көрсетті [8, 27].

Фейстель шифрын жақсырақ түсіну үшін бір қайтымды емес компонентті шифрлау және керішифрлау алгоритмдерде қалай қолдануға болатынын қарастырайық. Егер 3.17 суретте көрсетілгендей *xor* операциясын қолданса, онда шифрлау алгоритмдегі қайтымды емес компонент эффектісі керішифрлау алгоритмінде болмау мүмкін.

Шифрлау кезінде кілт *f(K)* қайтымды емес функция кірісіне барады, ол алғашқы деректерімен *xor* операторының бір қосылғышы болып табылады. Нәтижесінде шифрланған деректер болады. *f(K)* функция мен *xor* операцияның комбинациясын –

қоспалауыш деп атаймыз (басқа ат болмағандықтан). Қоспалауыш *Фейстель* шифрының жаңа нұсқаларында маңызды рөлді атқарады.



3.17 сурет – Фейстельдің шифрлау және керішифрлау алгоритмдерінде қайтымды емес компонентті қолдану

Шифрлау және керішифрлау кезінде кілт бір болғандықтан, екі алгоритм бір біріне инверсты екенін дәлелдеуге болады. Басқа сөзбен, егер $C_2 = C_1$ (жіберу кезінде шифрланған деректерде кез келген өзгеріс), онда $M_2 = M_1$.

Шифрлау:

$$C_1 = M_1 \oplus f(K).$$

Керішифрлау:

$$M_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = M_1 \oplus f(K) \oplus f(K) =$$
$$= M_1 \oplus (00...0) = M_1.$$

Есептеу үшін *xor* операцияның екі қасиеті қолданғанына назар аударыңыз (инверсияның болуы және нөлдік код болуы).

Жоғарыда қорсетілген теңдеулер қоспалауышта айырбасталымды элемент бар болсада, қоспалауыштың өзі өзайырбасталымды болатынын дәлелдейді.

3.12 мысал. Ұзындығы 4 биттен алғашқы және шифрланған деректер және 3 биттік кілт бар болсын. *f(K)* функциясы бірінші және үшінші биттерді алып ондық нөмір ретінде түсініп, оның екі дәрежесін табады және нәтижені 4-биттік екілік тізбек ретінде қабылдайды. Егер алғашқы деректер – *0111,* және кілт – *101* болса, деректердің шифрлау және керішифрлау нәтижелерін көрсету қажет.

Шешім. f(K) функциясы кілттің бірінші және үшінші битын алады. Нәтижесінде екілік түрде *11* немесе ондық бейнеде *3* болады. Екілік дәрежеге шығару нәтижесі – *9*, екілік бейнеде *1001*.

Шифрлау: $C = M \oplus f(K) = 0111 \oplus 1001 = 1110$.

Керішифрлау: $M = C \oplus f(K) = 1110 \oplus 1001 = 0111$.

Көрініп тұрғандай керішифрлау нітижесі *M* алғашқы деректерімен тең.

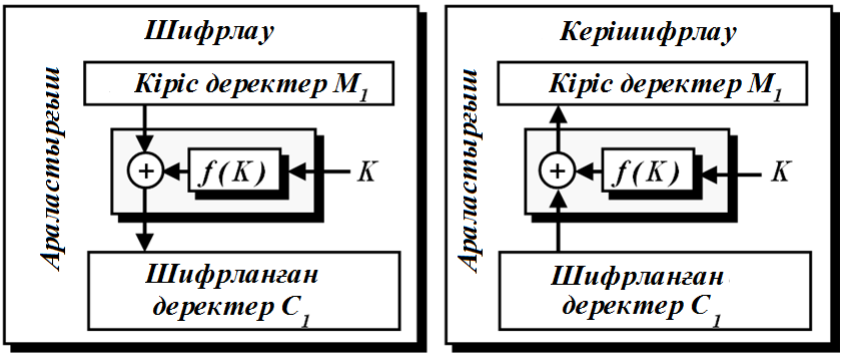
f(101) = 1001 функциясы айырбасталмайтын, бірақ *xor* операциясы *f(K)* функциясын шифрлау және керішифрлау алгоритмдерде қолдануға мүмкіндік береді. Басқа сөзбен, *f(K)* функциясы айырбасталмайтын, бірақ қоспалауыш өзіайырбастылымды болады.

Фейстель шифрына жақындау үшін 3.17 суретіндегі шифрды жетілдіру қажет. Ол үшін айырбасталмайтын элементтің (функцияның) кірісін қолдану қажет екені белгілі, бірақ бұл жерде кілт және тағы басқаны қолдану қажет. *f(K)* функцияның кірісін алғашқы деректер бөлігін шифрлау үшін және шифрланған деректердің бөлігін керішифрлау үшін қолданайық. Кілт функцияның екінші кірісі ретінде қолдану мүмкін. Осы тәсіл арқасында *f(K)* функциясы кейбір кілттік емес және кілттік элементтері бар күрделі элемент болады. Мақсатқа жету үшін, алғашқы және шифрланған деректерді ұзындығы бірдей екі блокқа бөлеміз – сол жақты (*L*) және оң жақты (*R*). Оң жақтағы блок *f(K)* функциясына енгізіледі және $f(R_1,K)$ болады, ал сол жақтағы блок *xor* операциясы көмегімен $f(R_1,K)$ функциясының шығысымен қосылады. Шифрлау және керішифрлау кезінде $f(R_1,K)$ функциясына келетін шығыстар нақты сәйкес болу керектігін еске сақтау қажет. Бұл деген, шифрлаудың алдында алғашқы деректердің R_1 оң жақ секциясы және шифрлаудан кейін шифрланған деректердің R_1 оң жақ секциясы сәйкес келетінін білдіреді. Басқа сөзбен, R_1 секциясы өзгерусіз шифрлауға кіру және керішифрлаудан шығу керек. 3.18 суреті бұл идеяны көрсетеді.

Шифрлау және керішифрлау алгоритмдері бір біріне инверсты. $L_3 = L_2$ және $R_3 = R_2$ (шифрланған деректерде жіберу кезінде өзгерістер болған жоқ) дейік. Онда

$$R_4 = R_3 = R_2 = R_1,$$

$$L_4 = L_3 \oplus f(R_3,K) = L_2 \oplus f(R_2,K) = L_1 \oplus f(R_1,K) \oplus f(R_1,K) = L_1.$$



3.18 сурет – *Фейстельдің* алдынғы сұлбасын жетілдіру

Шифрлау алгоритмінде қолданатын алғашқы деректер – бұл керішифрлау алгоритмімен дұрыс қалпына келтірілген деректер.

Алдынғы жетілдіруде бір кемшілік бар: алғашқы деректердің оң жағы ешқашан өзгермейді. Қасқой шифрланған деректерді бөліктерге бөліп, оның оң жағын ашып алғашқы деректердің оң жағын дереу таба алады. Жоба ары қарай жетілдіру қадамдарды талап етеді: *біріншіден*, раундар санын көбейту қажет; *екіншіден*, әр раундқа жаңа элемент қосу қажет – *ауыстыру құрылғысын*. Шифрлау раундында ауыстыру құрылғысының әсері керішифрлау раундында ауыстыру құрылғысының әсерімен орын толтырады. Бұл әр раундта оң және сол жақтарын орнымен ауыстыруға мүмкіндік береді. 3.19 сурет екі раунды бар *Фейстель* шифрының нұсқасын көрсетеді.

Раундтардың екі кілті k_1 және k_2 бар екеніне назар аударыңыз. Кілттер шифрлау және керішифрлау кезінде кері ретпен қолданады.

Қоспалауыш және ауыстыру құрылғылары бір біріне инверсты болғандықтан, шифрлау және керішифрлауда бір біріне инверсты болады. Бұл фактыны әр шифрда оң жақ және сол жақ секциялар арасындағы қатынастарды қолданумен дәлелдеуге болады. Басқа сөзбен, егер $L_6 = L_1$ және $R_6 = R_1$ болса, онда $L_4 = L_3$ және $R_4 = R_3$ (жіберу кезінде шифрланған деректер өзгермейді) деп болжайық. Алдымен бұны аралық деректер үшін дәлелдейік:

$$L_5 = R_4 \oplus f(L_4, K_2) = R_3 \oplus f(R_2, K_2) =$$

$$= L_2 \oplus f(R_2, K_2) \oplus f(R_2, K_2) = L_2,$$

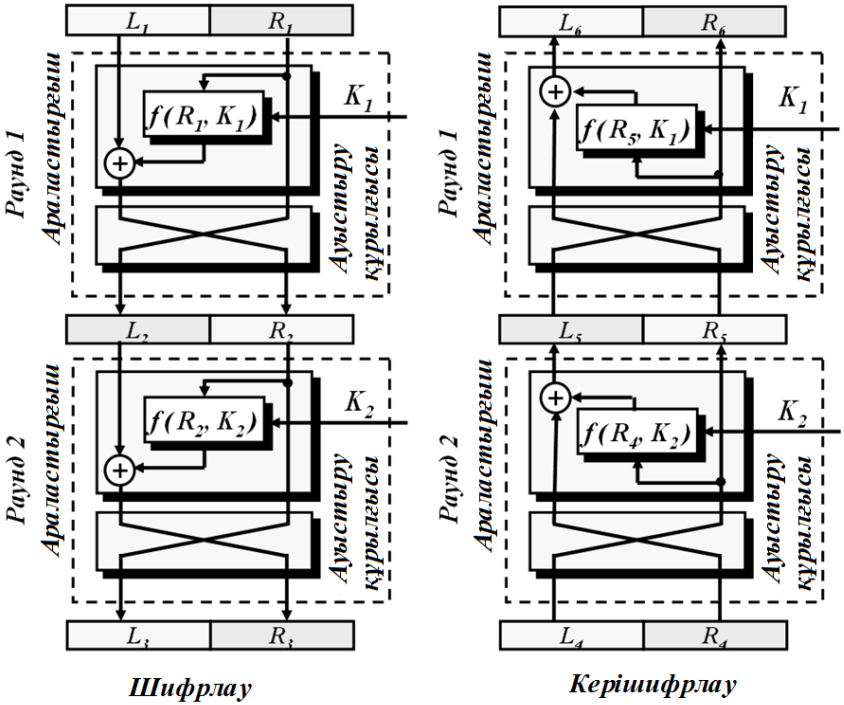
$$R_5 = L_4 = L_3 = R_2.$$

Онда алғашқы деректердің екі блогы үшін теңдікті дәлелдеу оңай:

$$L_6 = R_5 \oplus f(L_5,K_1) = R_2 \oplus f(L_2,K_1) =$$

$$= L_1 \oplus f(R_1,K_1) \oplus f(R_1,K_1) = L_1,$$

$$R_6 = L_5 = L_2 = R_1.$$



3.19 сурет – Екі раунды бар *Фейстель* шифрының сонғы нұсқасы

Фейстельдікі-емес шифрлар тек қайтымды компоненттерді қолданады. Алғашқы деректердегі компонентте шифрда сәйкес келетін компоненті болады. Мысалы, сәйкес (қайтымды) болу үшін ауыстырудың *S*-блоктарында кіріс және шығыстар саны бірдей болу керек. *Фейстельдікі*-емес шифрларда *P*-блоктарды ешқандай қысу немесе кеңейту болмау керек, себебі олар қайтымды емес болып

қалады. *Фейстельдікі*-емес шифрларда ашық мәтінді екі жартыға бөлу қажеттілігі жоқ.

3.19 суретін *Фейстельдікі*-емес шифрлар жұмысының принципін графикалық көрсету ретінде қарастыруға болады, себебі әр раундта бірлік компонеттер – өзқайтымды *xor* операциялары, қайтымды болатындай жасалу мүмкін 2×2 *S*-блоктары және егер сәйкес орын ауыстыру кестесі қолданса қайтымды болатын түзу *P*-блоктар. Әр компонент қайтымды болғандықтан, әр раунд қайтымды болатынын көрсетуге болады. Ол үшін раунд кілттерін кері ретпен қолдану қажет. Шифрлау k_1 және k_2 раунд кілттерін қолданады. Керішифрлау алгоритмі k_2 және k_1 раунд кілттерін қолдану қажет.

3.3. БЛОКТЫ ШИФРЛАРҒА ШАБУЫЛДАР

Дәстүрлі шифрлар шабуылдарын заманауи блокты шифрларға да қолдануға болады, бірақ бүгінгі блокты шифрлар шабуылдардың көбіне қарсы тұра алады. Мысалы, кілттің «қатқыл күш» шабуылы ереже бойынша, жүзеге асырымды емес, себебі әдетте кілттер өте ұзын. Бірақ жақында блокты шифрлар шабуылдарының жаңа түрлері шықты, олар заманауи блокты шифрлар құрылымдарына негізделген. Бұл шабуылдар дифференциалдық және сызықты криптоталдау әдістерін қолданады.

3.3.1. Дифференциалды криптографиялық талдау

Дифференциалды криптографиялық талдауға қатысты идеяны *Эли Бихам* және *Ади Шамир* ұсынды. Бұл – алғашқы деректерді таңдаумен шабуыл. Қасқой кейбір тәсілмен жіберуші компьютеріне қол жеткізіп алғашқы деректер бөлігін және оларға сәйкес келетін шифрланған деректерін алу мүмкін. Мақсат жіберуші шифрының кілтін табуда.

Талдау алгоритмі. Қасқой алғашқы деректерді таңдаумен шабуылды қолданардың алдында, ол алғашқы деректер және шифрланған деректер арасындағы қатынас туралы кейбір ақпаратты жинау үшін шифрлау алгоритмін талдау қажет. Қасқой шифр кілтін білмейтіні айқын. Бірақ кейбір шифрлардың құрылымында әлсіздіктер болады, олар қасқойға алғашқы деректер және

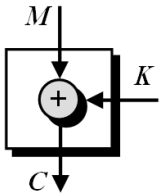
шифрланған деректер арасында айырмашылықтарын табуға мүмкіндік бере алады.

3.13 мысал. 3.20 суретінде көрсетілгендей, шифр тек бір *xor* операциясынан тұрсын дейік.

Кілт мәнін білмей, қасқой алғашқы және шифрланған деректер айырымы арасындағы қатынастарды оңай таба алады.

Егер алғашқы деректердің айырымын $M_1 \oplus M_2$ деп белгілесек, ал шифрланған деректер айырымын $C_1 \oplus C_2$ деп белгілесек, онда келтірілген келесі түрлендірулер келесіні дәлелдейді $C_1 \oplus C_2 = M_1 \oplus M_2$:

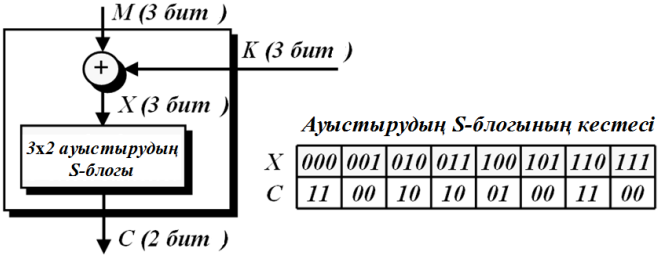
$$C_1 = M_1 \oplus K;\ C_2 = M_2 \oplus K;$$



3.20 сурет – 3.13 мысал үшін диаграмма

Алайда бұл мысал реалды емес; модемді блокты шифрлар соншалықты қарапайым емес.

3.14 мысал. 3.21 суретінде көрсетілгендей 3.13 мысалына бір ауыстыру *S*-блогын қосу қажет.



3.21 сурет – 3.14 мысалы үшін диаграмма

Екі X және екі $P\,(X_1 \oplus X_2 = M_1 \oplus M_2)$ арасындағы айырымды қолданған кезде кілтпен шифрлау әсері болмаса да, ауыстыру *S*-блогының бар болуы қасқойға алғашқы деректердің айырымы және шифрланған деректер айырымы арасындағы анықталған қатынастарды табуға кедергі жасайды. Бірақ ықтималдық қатынастарды анықтауға болады.

Қасқой алғашқы деректер айырымы үшін қанша шифрланған деректер айырымдарын құруға болатынын көрсететін кестені құрастыру мүмкін (3.5 кестені қараңыз) – шифр.

Кесте 3.21 сурет бойынша ауыстыру *S*-блогының кіріс-шығыс кестесін есепке алып құрылған ақпарат бойынша жасалғанына назар аударыңыз, себебі $M_1 \oplus M_2 = X_1 \oplus X_2$.

3.5кесте

3.14 мысалындағы шифр үшін кіру және шығулардың дифференциалды кестесі

	$C_1 \oplus C_2$				
$M_1 \oplus M_2$		*00*	*01*	*10*	*11*
	000	*8*			
	001	*2*	*2*		*4*
	010	*2*	*2*	*4*	
	011		*4*	*2*	*2*
	100	*2*	*2*	*4*	
	101		*4*	*2*	*2*
	110	*4*		*2*	*2*
	111			*2*	*6*

Кілт көлемі – 3 бит болғандықтан, енгізудің әр айырымы үшін сегіз күй болу мүмкін. Кесте егер кіріс алу нәтижесі – $(000)_2$, шығыс алу нәтижесі – үнемі $(00)_2$ болатынын көрсетеді. Басқа жақтан, егер кіріс айырымы – $(100)_2$, онда $(00)_2$ екі шығыс айырымдар күйлері, $(01)_2$ екі шығыс айырымдар күйлері және $(10)_2$ төрт шығыс айырымдар күйлері болатынын кесте көрсетеді.

3.15 мысал. 3.14 мысалдың эвристикалық нәтижесі 3.6 кестесінде көрсетілгендей, қасқой үшін ықтималдық ақпаратты құру мүмкін. Кестенің кірістері пайда болу ықтималдығына сәйкес. Нөлдік ықтималдығы бар айырымдар ешқашан болмайды.

Кешірек белгіленетіндей, қасқойда шабуылды бастау үшін жеткілікті ақпарат бар. Ауыстырудың *S*-блогының құрылымында әлсіздік бар болғандықтан ықтималдықтар әртекті бөлінгенін 3.6 кестесі көрсетеді. 3.6 кестесі кей кезде таратудың дифференциалды кестесі немесе *xor* профайы деп аталады.

Таңдалған алғашқы деректер шабуылын іске қосу. Эвристикалық талдау бір рет жасалғаннаң кейін, ол шифр құрылымы өзгергенше болашақ қолдану үшін сақталу мүмкін. Қасқой шабуылдар үшін алғашқы деректерді таңдау мүмкін. Ытималдықты таратудың дифференциалды кестесі (3.6 кесте) қасқойға оларды таңдауға көмектеседі – ол кестеде ең жоғары ықтималдығы барларды алады.

3.6 кесте

3.14 мысалда шифр үшін кіріс және шығыстардың дифференциалды кестесі

	00	*01*	*10*	*11*
000	*1*	*0*	*0*	*0*
001	*0,25*	*0,25*	*0*	*0,50*
010	*0,25*	*0,25*	*0,50*	*0*
011	*0*	*0,50*	*0,25*	*0,25*
100	*0,25*	*0,25*	*0,50*	*0*
101	*0*	*0,50*	*0,25*	*0,25*
110	*0,50*	*0*	*0,25*	*0,25*
111	*0*	*0*	*0,25*	*0,75*

Кілт мәнін болжау. Алғашқы деректерді сәйкес таңдаумен кейбір шабуылдарды іске қосқаннаң кейін қасқой кейбір «алғашқы деректер/шифрланған деректер» жұбын табу мүмкін, ол кілттің кейбір мәнін болжауға мүмкіндік береді. Процесс *C* басталады және *M* қарай жылжыйды.

3.16 мысал. 3.6 кестені қарастырып, қасқой біледі, егер $M_1 \oplus M_2 = 001$, онда *0,5* ықтималдығымен $C_1 \oplus C_2 = 11$. Ол $C_1 = 00$ алып көреді және $M_1 = 010$ алады (шифрланған деректерді таңдаумен шабуыл). Ол тағы $C_2 = 11$ байқайды және $M_2 = 011$ алады (шифрланған деректерді таңдаумен басқа шабуыл). Енді қасқой M_1 және C_1 бірінші жұпта негізделген талдауға қайтып келуді көреді (3.21 суретті қараңыз):

$$C_1 = 00 \rightarrow X_1 = 001 \text{ немесе } X_1 = 111.$$

Егер $X_1 = 001$, онда $K = X_1 \oplus M_1 = 011$.
Егер $X_1 = 111$, онда $K = X_1 \oplus M_1 = 101$.
M_2 және C_2 жұбын қолданып келесіні аламыз

$$C_2 = 11 \rightarrow X_2 = 000 \text{ өйткені } X_1 = 110.$$

Егер $X_2 = 000$, онда $K = X_2 \oplus M_2 = 011$.
Егер $X_2 = 110$, онда $K = X_2 \oplus M_2 = 101$.

Екі сынау $K = 011$ немесе $K = 101$ көрсетеді. Қасқой кілттің нақты мәні қандай екеніне сенімді болмасада, ол ең оң жақтағы бит – *1* (екі мән арасындағы жалпы бит) екенін біледі. Шабуылды

жалғастырып ең оң жақтағы бит – *1* есепке алуға болады. Сонымен, бұл кілттің басқа биттерін анықтауға болады.

Жалпы процедура. Заманауи шифрлардың қиындығы бұл тарауда қарастырылғаннан жоғары. Осымен қоса, оларда әртүрлі раундтар саны болу мүмкін. Қасқой келесі стратегияны қолдану мүмкін.

1. Әр раундта бірдей операциялар болғандықтан, қасқой әр раундқа таратуды құру үшін, әр ауыстыру *S*-блогы үшін дифференциалдық тарату кестесін (*xor* профайлын) құруға және оларды комбинациялауға мүмкіндік алады.

2. Әр раунд тәуелсіз (әділ болжам) деп болжайық. Қасқой сәйкес ықтималдықтарды көбейту арқылы бүкіл шифр үшін тарату кестесін құруға мүмкіндік алады.

3. Енді қасқой шабуыл үшін екінші қадамда құрылған тарату кестесінде негізделген алғашқы деректерді бере алады. Екінші қадамды орындау кезінде құрылған кесте қасқойға «алғашқы деректер/шифрланған деректер» жұптарының тек аз санын таңдауға көмектесетінін еске салайық.

4. Қасқой шифрланған деректерді таңдайды және сәйкес келетін алғашқы деректерді табады. Содан кейін ол кілттің кейбір биттерін табу үшін нәтижені талдайды.

5. Көбірек кілттің биттерін табу үшін қасқой 4 қадамның әрекетін қайталайды.

6. Кілттің жеткілікті биттер санын тапқаннан кейін толық кілтті табу үшін қасқой "*қатқыл күш*" шабуылын қолдану мүмкін.

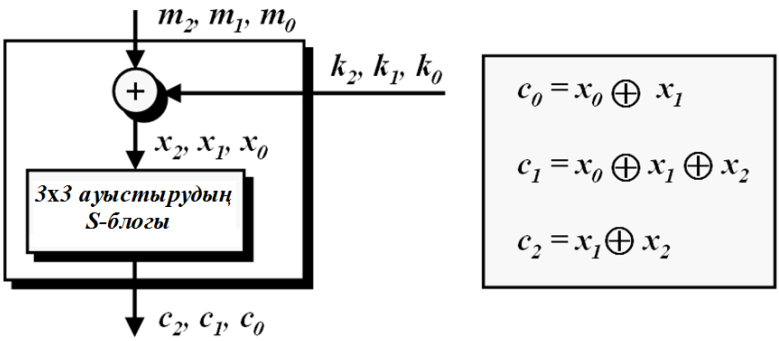
Дифференциалды криптографиялық талдау блокты шифрда ауыстыру *S*-блоктардың біркелкі емес дифференциалдық таратулар кестесінде базаланады.

Дифференциалды криптографиялық талдау толығырақ 5 тарауда келтіріледі.

3.3.2. Сызықты криптографиялық талдау

Сызықты криптографиялық талдауды 1993 жылы *Митцури Мацуи* (*Mitsuru Matsui*) ұсынды. Талдау белгілі алғашқы деректерге шабуылдарды қолданады (дифференциалды криптографиялық талдауда таңдалған алғашқы деректерге негізделген). Бұл шабуылдың толық қарастырылуы ықтималдық теорияның кейбір түсініктеріне негізделеді. Бұл шабуылдың басты идеясын қарастыру үшін, шифр

3.22 суретте көрсетілгендей бір раундтан тұрсын, c_0, c_1 және c_2 ауыстыру *S*-блогының шығысындағы үш бит, ал x_0, x_1 және x_2 – кірісіндегі үш бит.



3.22 сурет – Ауыстырудың сызықты *S*-блогымен қарапайым шифр

Ауыстырудың *S*-блогы – сызықты түрлендіру, оның ішінде жоғарыда көрсетілгендей, әр шығыс кірістің сызықты функциясы. Осы сызықты компонентпен, төменде көрсетілгендей, алғашқы деректер және шифрланған деректер биттері арасында үш сызықты теңдеуді құруға болды:

$$c_0 = m_0 \oplus k_0 \oplus m_1 \oplus k_1;$$

$$c_1 = m_0 \oplus k_0 \oplus m_1 \oplus k_1 \oplus m_2 \oplus k_2;$$

$$c_2 = m_1 \oplus k_1 \oplus m_2 \oplus k_2.$$

Үш белгісіз үшін теңдеулер жүйесін шешіп, келесіні аламыз

$$k_1 = (m_1) \oplus (c_0 \oplus c_1 \oplus c_2);$$

$$k_2 = (m_2) \oplus (c_0 \oplus c_1);$$

$$k_0 = (m_0) \oplus (c_1 \oplus c_2).$$

Бұл, белгілі алғашқы деректерге үш шабуыл k_1 және k_2 табуға мүмкіндік алатынын білдіреді. Бірақ реалды блокты шифрлар сонша лықты қарапайым емес; оларда компоненттер көбірек, және ауыстырудың *S*-блоктары сызықты емес.

Сызықты аппроксимация. Кейбір заманауи блокты шифрларда кейбір ауыстырудың *S*-блоктары толық сызықты емес болу мүмкін; онда олар ықтималдық мағынасында кейбір сызықты функцияларымен аппроксимацияланған болу мүмкін. Жалпы, алғашқы және шифрланған деректерді *n* битпен және кілтті *m* битпен беру келесі түрі бар кейбір теңдеулер ізделінеді.

$$(k_0 \oplus k_1 \oplus \ldots \oplus k_x) = (m_0 \oplus m_1 \oplus \ldots \oplus m_y) \oplus$$

$$\oplus (c_0 \oplus c_1 \oplus \ldots \oplus c_z).$$

Бақылау сұрақтары және тапсырмалары

1. Заманауи және дәстүрлі симметриялық кілті бар шифрлар арасындағы айырмашылықтарды көрсетіңіз.
2. Неге заманауи блокты шифрлар транспозиция шифрларын қолданудың орнына ауыстыру шифрлары сияқты жобаланғанын түсіндіріңіз.
3. Заманауи блокты шифрлардың компоненттерін айтыңыз.
4. *P*-блокты анықтаңыз және оның үш нұсқасын айтыңыз. Қай нұсқа қайтымды болады?
5. *S*-блокты анықтаңыз және қайтымды *S*-блоктардың қажетті шартын көрсетіңіз.
6. Құрамдас шифрды анықтаңыз және құрамдас шифрлардың екі класын айтыңыз.
7. Шашырату және араластыру арасындағы айырмашылықты түсіндіріңіз.
8. *Фейстель* және *Фейстельдікі емес* блокты шифрлар арасындағы айырмашылықты түсіндіріңіз.
9. Сызықты және дифференциалды криптографиялық талдаулар арасында қандай айырмашылық бар? Қай криптографиялық талдау таңдалған алғашқы деректерге шабуылды қолданады, ал қайсысы – белгілі алғашқы деректерге?
10. Хабарда *1500* символ бар, оларды бейнелеу үшін *ASCII* кодтар қолданылады. Бұл хабар *64* бит ұзындығы бар блокты шифрмен шифрланады. Толықтауыш көлемін және шифрланған блоктар санын табыңыз.

11. Транспозиция блогында *4* кіріс және *4* шығыс бар. Орын ауыстыру тобының реті және биттермен берілген кілттік тізбек көлемі қандай?

12. Ауыстыру блогында *4* кіріс және *4* шығыс бар. Орын ауыстыру тобының реті және биттермен берілген кілттік тізбек көлемі қандай?

13. 10011011_2 сөзін қолданып, *3* битке солға циклдық жылжудың нәтижесін көрсетіңіз.

14. 10011011_2 сөзін қолданып, *3* битке оңға циклдық жылжудың нәтижесін көрсетіңіз.

15. Келесі операциялар нәтижелерін табыңыз:

а) $(01001101)_2 \oplus (01001101)_2$; б) $(01001101)_2 \oplus (10110010)_2$;

в) $(01001101)_2 \oplus (00000000)_2$; г) $(01001101)_2 \oplus (11111111)_2$.

16. 3.4 суретінде көрсетілген түзу *P*-блок үшін, егер оның кірісінде $(10110)_2$ болса, биттер орын ауыстыруын орындаңыз.

17. 3.4 суретінде көрсетілген қысу *P*-блок үшін, егер оның кірісінде $(10110)_2$ болса, биттер орын ауыстыруын орындаңыз.

18. 3.4 суретінде көрсетілген кеңейту *P*-блок үшін, егер оның кірісінде $(101)_2$ болса, биттер орын ауыстыруын орындаңыз.

19. Келесі кестемен анықталған *P*-блокты көрсетіңіз:

8	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Көрсетілген *P*-блок қандай блоктар типіне жатады: түзу; қысу немесе кеңейту?

20. Келесі орын ауыстыру кестесі бар *P*-блок:

1	*1*	*2*	*3*	*4*	*4*

түзу *P*-блок, қысу *P*-блок немесе кеңейту *P*-блокқа жататынын анықтаңыз.

21. Келесі орын ауыстыру кестесі бар *P*-блок:

1	*3*	*5*	*6*	*7*

түзу *P*-блок, қысу *P*-блок немесе кеңейту *P*-блокқа жататынын анықтаңыз.

22. Келесі орын ауыстыру кестесі бар *P*-блок:

7	*3*	*1*	*4*	*8*	*5*	*2*	*6*

түзу *P*-блок, қысу *P*-блок немесе кеңейту *P*-блокқа жататынын анықтаңыз.

23. *2×2 S*-блоктың кіріс-шығыс қатынастары келесі кестеде көрсетілген:

Кіріс: сол жақ бит \ Кіріс: оң жақ бит	0	1
0	01	11
1	00	10

Инверсты блок үшін кестені көрсетіңіз.

24. Ауыстырудың *S*-блогы шығыстың сол жақ битын алу үшін тақ биттермен *xor* операциясын орындайды, шығыстың оң жақ битын алу үшін жұп биттермен *xor* операциясын орындайды. Егер кіріс – $(110010)_2$, онда шығыс қандай болады? Егер кіріс – $(101101)_2$, онда шығыс қандай болады?

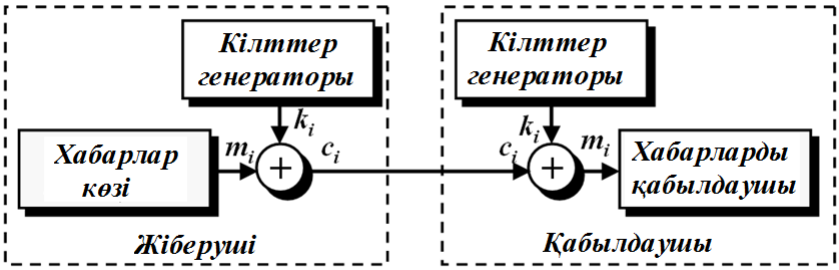
25. *4 × 3* көлемі бар ауыстырудың *S*-блогының щеткі сол жақтағы биты басқа үш биттің жылжуын анықтайды. Егер щеткі сол жақтағы бит 0 тең болса, онда басқа үш бит бір битке оң жаққа жылжыйды. Егер щеткі сол жақтағы бит 1 тең болса, онда басқа үш бит сол жаққа бір битке жылжыйды. Егер кіріс – *1011*, онда шығыста қандай нәтиже болады? Егер кіріс – *0110*, онда шығыста қандай нәтиже болады?

4 тарау
АҒЫНДЫ ШИФРЛАР ЖӘНЕ ПСЕВДОКЕЗДЕЙСОҚ САНДАР ГЕНЕРАТОРЛАРЫ

4.1. АҒЫНДЫ ШИФРЛАР ТУРАЛЫ ЖАЛПЫ МӘЛІМЕТТЕР

Блокты алгоритм анықталған ұзындығы бар блоктарды шифрлау үшін арналған. Бірақ деректерді блоктармен емес символдармен шифрлауға қажеттілік болу мүмкін. Бұндай талаптарға сәйкес *ағынды шифр (stream cipher)*, ол кіріс хабарды операцияда бір битпен (немесе байтпен) түрлендіруді орындайды. Ағынды шифрлау алгоритмі хабарды жеткілікті ұзындығы бар тұтас сандар блоктарына бөлуді талап етпейді, сондықтан, ол реалды уақытта жұмыс істей алады. Сонымен, егер символдар ағыны жіберілсе, онда әр символ шифрланады да жиберіледі.

Типтік ағынды шифрдың жұмысы 4.1 суретінде көрсетілген.



4.1 сурет - Ағынды шифрдың жұмыс істеу принципі

Кілттер генераторы k_i биттер ағынын береді, олар гамма ретінде қолданылады. Хабар көзі m_i ашық мәтін биттерін генерациялайды, олар гаммамен 2 модулі бойынша қосылады, нәтижесінде c_i шифрланған мәтін символдары шығады:

$$c_i = m_i \oplus k_i, \ i = 1, 2, \ldots, n.$$

Шифрланған $c_1, c_2, \ldots, c_n$ хабарынан $m_1, m_2, \ldots, m_n$ хабарды қалпына келтіру үшін шифрланған кезде қолданған $k_1, k_k, \ldots, k_n$ кілттік тізбекті генерациялау қажет және кері шифрлау үшін өрнекті қолдану қажет

$$m_i = c_i \oplus k_i, \ i = 1, 2, \ldots, n,$$

ойткені *2* модулі бойынша разрядтарды қосу операциясында қайтымдылық қасиеті бар.

Әдетте алғашқы хабар және кілттік тізбек тәуелсіз биттер ағыны болады. Сонымен, барлық ағынды шифрлар үшін шифрлау және керішифрлау түрлендірулері бірдей болғандықтан, олар тек кілттер генераторларын құру тәсілімен ғана айырылады. Жүйенің қауіпсіздігі кілттер ағыны генераторының қасиетінен толық тәуелді болады. Егер кілттер ағыны генераторы тек нөлдерден (немесе бірлерден) тұратын тізбекті берсе, онда шифрланған хабар алғашқы биттер ағыны сияқты болады (бірлерден тұратын кілттер жағдайда хабар алғашқының инверсиясы болады). Егер гамма ретінде бір символ қолданса, мысалы, сегіз биттен тұратын, онда шифрланған хабар алғашқымен бірдей болмасада жүйе қауіпсіздігі төмен болады. Бұл жағдайда – бүкіл хабар ұзындығына кілт кодының көп рет қайталанғаны оны статистикалық әдіспен ашуға қауіп тұдырады. Бұны қарапайым екілік хабарды қысқа екілік кодты кілтті қолданып гамма әдісімен шифрлау мысалында түсіндіреміз.

4.1 мысал. Алғашқы хабар екілік-ондық сан болсын, яғни санның әр тетрадасы (төрт биты) ондық санды *0...9* екілік түрге ауыстырумен алынған. Шифрланған *M* хабардың *24* биты жолай ұсталсын, яғни алты тетрада m_1, m_2, m_3, m_4, m_5 және m_6, нақтырақ $C = 1100\ 1101\ 1110\ 1111\ 0000\ 0001$ мәні. Шифрлау кілті төрт биттен тұратыны белгілі, оларда бірмәнді ондық сан, яғни бір мән $0 \le K \le 9$ алғашқы хабардың әр төрт битын шифрлауға қолданылды. Сонымен, *K* кілтімен шифрланған сандарды m_1, m_2, m_3, m_4, m_5 және m_6 теңдеулер жүйесімен көрсетуге болады:

$$m_1 \oplus K = 1100; \quad m_2 \oplus K = 1101; \quad m_3 \oplus K = 1110;$$

$$m_4 \oplus K = 1111; \quad m_5 \oplus K = 0000; \quad m_6 \oplus K = 0001.$$

m_i *0* ден *9* дейін ондық мәндерді қабылдайтын шарттан, белгісіз *K* табу үшін *2* модулі бойынша қосындысы *1100* нәтижеге алып келетін барлық мүмкінді m'_1 және *K* мәндерін табамыз:

$$\begin{array}{ll} & K = \ 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001 \\ \oplus & c_1 = \ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100 \\ \hline & m'_1 = \ 1100\ 1101\ 1110\ 1111\ 1000\ 1001\ 1010\ 1011\ 0100\ 0101 \end{array}.$$

Алғашқы хабар *0* ден *9* дейін сандардан тұрғандықтан, қарастырудан *0000, 0001, 0010, 0011, 0110, 0111* кілт мәндерін алып тастауға болады, олармен қосу нәтижесінде ондық эквивалентте *9* үлкен сандар шығады. Бұндай мәндер ашық хабарда болу мүмкін емес. Сонымен, талдаудың бірінші кезеңі мүмкінді кілттер санын оннан төртке дейін қысқартуға мүмкіндік берді.

Белгісіз *K* ары қарай іздеу үшін m_2' және қалған кілт варианттарының барлық мүмкінді мәндерін анықтаймыз, олардың *2* модулі бойынша қосындысы $c_2 = 1101$ нәтижесіне алып келу қажет:

$$\begin{array}{ll} & K = \ 0100\ 0101\ 1000\ 1001 \\ \oplus & c_2 = \ 1101\ 1101\ 1101\ 1101 \\ \hline & m'_2 = 1001\ 1000\ 0101\ 0100 \end{array}.$$

Бұл кезең қалған кілттер варианттарының біреуінде алып тастауға мүмкіндік бермегені көрініп тұр. Бұны $c_3=1110$ қолданып орындауға тырысамыз:

$$\begin{array}{ll} & K = \ 0100010110001001 \\ \oplus & C_3 = 1110111011101110 \\ \hline & m'_3 = 1010101101100111 \end{array}.$$

Бұл кезеңді өткізген соң кілт болып *0100* және *0101* мәндері болмайтыны белгілі болды. Кілттің екі мүмкінді мәні қалды: $1000_{(2)}=8_{(10)}$ и $1001_{(2)}=9_{(10)}$.

Берілген әдістеме бойынша арықарай талдау өкінішке орай қалған кілттің қайсысы шифрлау кезінде қолданғаны туралы бірмәнді айтуға мүмкіндік бермейді. Бірақ кілт кеңістігі оннан екіге дейін төмендегені жетістік деп есептеуге болады. Енді екі табылған кілтті хабарды керішифрлау үшін қолданып, ашылған деректердің мәнін талдау қажет.

Реалды жағдайларда, алғашқы мәтін тек сандардан емес, басқа символдарды қолданумен құрылған болса, құпия деректерді жапқан қысқа кілтті статистикалық талдауды қолдану арқылы тез және нақты қалпына келтіруге мүмкіндік береді.

4.2. АҒЫНДЫ ШИФРЛАУ КЕЗІНДЕ ПСЕВДОКЕЗДЕЙСОҚ САНДАР ГЕНЕРАТОРЛАРЫН ҚОЛДАНУ ПРИНЦИПТЕРІ

Заманауи ақпараттану псевдокездейсоқ сандарды әртүрлі қосымшаларда кең қолданады – математикалық статистика және имитациялық модельдеу әдістерінен криптографияға дейін. Бұл жерде қолданатын *псевдокездейсоқ сандар генераторының (ПКСГ)* сапасынан алынатын нәтижелер сапасы тікелей тәуелді болады.

ПКСГ ағынды шифрларда кілттер генераторлары ретінде қолдану мүмкін. *ПКСГ* қолдану мақсаты кілттің өзінің қысқа ұзындығымен *«шексіз»* кілттік сөзді алу. *ПКСГ* кездейсоққа ұқсас биттер тізбегін құрады. Шынында, бұндай тізбектер анықталған ережелер бойынша есептеледі және кездейсоқ болмайды, сондықтан олар жіберетін және қабылдайтын жақта нақты қалпына келтірілу мүмкін. Шифрлау кезінде қолданатын кілттік символдар тізбегі тек жеткілікті ұзын ғана болмау керек. Егер кілттер генераторы әр қосылған сайын бір тізбекті берсе, онда осындай жүйені бұзу мүмкін болады. Демек, кілттер ағынының генераторының шығысы кілттің функциясы болу қажет. Бұл жағдайда хабарды керішифрлау және оқу тек шифрлау кезінде қолданған кілттің қолдануымен болады.

Криптографиялық мақсаттарға *ПКСГ* қолдану үшін оның келесі қасиеттері болу қажет:

- тізбектің периоды өте үлкен болу керек;
- тұындалатын тізбек расында кездейсоқтан айырмашылығы *«жоқ дерлік»* болу керек;

- әртүрлі мәндердің пайда болу (тұындау) ықтималдықтары нақты тең болу қажет;

- заңды пайдаланушы хабарды керішифрлау үшін, k_i кілттік биттер ағынын алған кезде кейбір құпия кілтті қолдану және есепке алу қажет, және де k_{i+1} санын k_i тізбегінің белгілі алдынғы элементтер бойынша есептеу кілтті білмесе қиын есеп болу керек.

Көрсетілген қасиеттері бар кезде псевдокездейсоқ сандар тізбектері ағынды шифрларда қолдану мүмкін.

4.2.1. Сызықты конгруэнтты псевдокездейсоқ сандар генераторы

Псевдокездейсоқ сандар генераторлары әртүрлі алгоритмдер бойынша істеу мүмкін. Қарапайым генераторларының бірі *сызықты конгруэнтты генератор,* ол кезекті k_i санын есептеу үшін келесі өрнекті қолданады:

$$k_i = (a*k_{i-1}+b) \bmod c,$$

бұл жерде a, b, c – кейбір тұрақтылар, ал k_{i-1} – алдынғы псевдокездейсоқ сан.

k_1 санын табу үшін алғашқы мән k_0 беріледі. Мысал ретінде $a = 5$, $b = 3$, $c = 11$ және $k_0 = 1$ аламыз. Бұл жағдайда жоғарыда келтірілген өрнек бойынша 0 ден 10 дейін ($c = 11$ болғандықтан) мәндерді алуға болады. Тізбектің бірнеше элементін есептейміз:

$$k_1 = (5 * 1 + 3) \bmod 11 = 8; \quad k_2 = (5 * 8 + 3) \bmod 11 = 10;$$

$$k_3 = (5 * 10 + 3) \bmod 11 = 9; \quad k_4 = (5 * 9 + 3) \bmod 11 = 4;$$

$$k_5 = (5 * 4 + 3) \bmod 11 = 1.$$

Алынған мәндер (8, 10, 9, 4, 1) кездейсоқ сандарға ұқсас көрінеді. Алайда келесі мән k_6 қайтадан 8 тең болады:

$$k_6 = (5 * 1 + 3) \bmod 11 = 8,$$

ал k_7 және k_8 мәндері сәйкес 10 және 9 тең болады:

$$k_7 = (5 * 8 + 3) \bmod 11 = 10; \quad k_8 = (5 * 10 + 3) \bmod 11 = 9.$$

ПКСГ қайталанатын сандар тізбегін қалыптастырады, *8, 10, 9, 4, 1* сандарды периодты түрде тұындатып. Өкінішке орай, бұл қасиет барлық сызықты конгруэнтты генераторларда бар. Негізгі параметрлер *a, b* және *c* мәндерін өзгертіп, период ұзындығына және тұындайтын мәндер k_i әсер етуге болады. Мысалы, *c* санын үлкейту периодты үлкейтуге алып келеді. Егер *a, b* және *c* параметрлері дұрыс таңдалса, онда генератор максималды периоды *c* тең кездейсоқ сандарды тұындатады. Бағдарламалық жүзеге асыру кезінде *c* әдетте 2^{b-1} немесе 2^b тең болып орнатылады, бұл жерде *b* – *электронды есептеу машинаның (ЭЕМ)* немесе *автоматтандырылған жүйенің (АЖ)* битпен берілген сөз ұзындығы.

Сызықты конгруэнтты *ПКСГ* артықшылықтары – олардың қарапайымдылығы және псевдокездейсоқ мәндерді алудың жоғары жылдамдығы. Сызықты конгруэнтты генераторлар модельдеу және математикалық статистика есептерін шешуде қолданылады, бірақ криптографиялық мақсаттарда оларды қолдануды ұсыныға болмайды, өйткені криптографиялық талдау бойынша мамандар бірнеше мәндер бойынша *псевдокездейсоқ сандардың (ПКС)* барлық тізбегін қалпына келтіруді үйренді. Мысалы, қасқой k_0, k_1, k_2, k_3 мәндерін анықтай алады дейік. Сонда:

$$k_1 = (a*k_0+b) \bmod c; \quad k_2 = (a*k_1+b) \bmod c; \quad k_3 = (a*k_2+b) \bmod c.$$

Үш теңдеуден тұратын жүйені шешіп, *a, b* және *c* табуға болады.

ПКС алу үшін квадрадтты және кубты генераторларды қолдану ұсынылған:

$$k_i = (a_1^2*k_{i-1}+a_2*k_{i-1}+b) \bmod c;$$

$$k_i = (a_1^3*k_{i-1}+a_2^2*k_{i-1}+a_3*k_{i-1}+b) \bmod c.$$

Бірақ бұндай генераторларда криптографиялық мақсаттарға тұра сол *«болжамды»* себебімен жарамсыз болып қалды. .

4.2.2. Кешігумен Фибоначчи әдісі негізінде псевдокездейсоқ сандар генераторы

Кешігумен *Фибоначчи әдісі* (*Lagged Fibonacci Generator*) - *ПКС* сандарын генерациялаудың әдісінің бірі. Ол псевдокездейсоқ сандардың жоғарырақ *«сапасын»* алуға мүмкіндік береді.

Фибоначчи датчиктерінің жоғары әйгілілігі бүтін емес сандармен арифметикалық операцияларды орындау жылдамдығы бүтін сандық арифметика жылдамдығымен теңесумен байланысты, ал Фибоначчи датчиктері бүтін емес арифметикада табиғи жүзеге асырылады.

Кешігумен *Фибоначчи* әдісін қолданудың әртүрлі сұлбалары белгілі. Кең тараған Фибоначчи датчиктерінің бірі келесі рекурентті өрнекте негізделген:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{егер} \quad k_{i-a} \geq k_{i-b}; \\ k_{i-a} - k_{i-b} + 1, & \text{егер} \quad k_{i\ a} < k_{i\ b}, \end{cases}$$

бұл жерде k_i – *[0,1]* диапазонынан бүтін емес сандар, *a, b* – бүтін оң сандар, генератор параметрлері.

Фибоначчи дитчигіне жұмыс үшін *max {a,b}* алдынғы генерацияланған кездейсоқ сандарды білу қажет. Бағдарламалық жүзеге асыру кезде генерацияланған кездейсоқ сандарды сақтау үшін кейбір *a* және *b* параметрлерінен тәуелді жады көлемі қажет.

4.2 мысал. Келесі шығыс деректері бар: $a = 4$, $b = 1$, $k_0 = 0{,}1$; $k_1 = 0{,}7$; $k_2 = 0{,}3$; $k_3 = 0{,}9$; $k_4 = 0{,}5$ k_5 бастап кешігумен Фибоначчи әдісімен генерацияланатын бірінші он саннан тұратын тізбекті есептейміз:

$$k_5 = k_1 - k_4 = 0{,}7 - 0{,}5 = 0{,}2;$$

$$k_6 = k_2 - k_5 = 0{,}3 - 0{,}2 = 0{,}1;$$

$$k_7 = k_3 - k_6 = 0{,}9 - 0{,}1 = 0{,}8;$$

$$k_8 = k_4 - k_7 + 1 = 0{,}5 - 0{,}8 + 1 = 0{,}7;$$

$$k_9 = k_5 - k_8 + 1 = 0{,}2 - 0{,}7 + 1 = 0{,}5;$$

$$k_{10} = k_6 – k_9 + 1 = 0,1 – 0,5 + 1 = 0,6;$$

$$k_{11} = k_7 – k_{10} = 0,8 – 0,6 = 0,2;$$

$$k_{12} = k_8 – k_{11} = 0,7 – 0,2 = 0,5;$$

$$k_{13} = k_9 – k_{12} + 1 = 0,5 – 0,5 + 1 = 1;$$

$$k_{14} = k_{10} – k_{13} + 1 = 0,6 – 1 + 1 = 0,6,$$

Генерацияланған сандар тізбегі сырттай кездейсоққа ұқсас екенін көреміз. Расында, алынатын кездейсоқ сандарда жақсы статистикалық қасиеттері бар екенін зерттеулер растайды.

Кешугімен *Фибоначчи* әдісі бойынша генераторлар үшін ұсынылатын *a* және *b* параметрлері бар, олар сапаға тестіленген деп айтуға болады. Мысалы, зерттеушілер келесі мәндерді ұсынады: (*a, b*) = (*55, 24*), (*17, 5*) немесе (*97, 33*). Алынатын кездейсоқ сандардың сапасы *a* тұрақты мәнінен тәуелді: ол үлкейген сайын, алынған кездейсоқ сандардан құрылатын кездейсоқ векторлардың біркелкілігі сақталатын кеңістік өлшемі жоғары болады. Сол уақытта *a* тұрақтының мәні үлкейген сайын алгоритм қолданатын жады көлемі де жоғарлайды.

Нәтижесінде (*a, b*) = (*17, 5*) мәндері қарапайым қосымшалар үшін ұсынылады. Кездейсоқ сандар сапасына қатал көптеген криптографиялық алгоритмдерге (*a, b*) = (*55, 24*) мәндері қанағаттандырлық сандарды алуға мүмкіндік береді. Өте жоғары сапасы бар кездейсоқ сандарды алуға (*a, b*) = (*97, 33*) мәндері мүмкіндік береді, оларды жоғары мөлшерлі кездейсоқ векторлармен жұмыс істейтін алгоритмдерде қолданады [32].

Кешігумен Фибоначчи әдісінде негізделген *ПКСГ* криптография мақсаттары үшін қолданған. Сонымен қатар, олар математикалық және статистикалық есептерде, кездейсоқ процесстерді модельдеуде қолданылады. Кешігумен *Фибоначчи* әдісі негізінде жасалған *ПКСГ* кең тараған *Matlab* жүйесінде қолданылды.

4.2.3. BBS алгоритм негізінде псевдокездейсоқ сандар генераторы

Кең тарауды *псевдокездейсоқ тізбектерді (ПКТ)* генерациялайтын *BBS алгоритм* (авторлар тегінен – *L. Blum, M.*

Blum, M. Shub) деп немесе *квадратты қалдығымен генератор* деп аталатын алгоритм алды. Криптография мақсаттары үшін бұл әдіс 1986 жылы ұсынылды [3, 32].

Алгоритмнің маңызы келесіде. Алдымен p және q екі үлкен жай сан таңдалады. Бұл сандар p және q *4* модулі бойынша *3 салыстырмалы* болу қажет, яғни p және q *4* бөлген кезде бірдей қалдық *3* болу қажет. Арықарай $M = p \cdot q$ саны есептеледі, ол *Блюмның бүтін саны* деп аталады. Содан кейін басқа кездейсоқ бүтін сан таңдалады, ол M өзара жай сан (бірден басқа ортақ бөлгіштері болмайтын) болады. $x_0 = x^2 \bmod M$ есептеледі, бұл жерде x_0 *генератордың сөрелік саны* деп аталады.

Генератор жұмысының әр n қадамында есептеледі

$$x_{n+1} = x_n^2 \bmod M.$$

Орындаудың n-ші қадам нәтижесі x_{n+1} санның бір (әдетте кіші) биты. Кейбір кезде нәтиже ретінде жұптық битын қабылдайды, яғни элементтің екілік түріндегі бірліктер саны. Егер санның жазуында бірліктер саны жұп боса – жұптық биты *0* деп қабылданады, егер тақ болса – жұптық биты *1* деп қабылданады.

4.2 мысал. $p = 11$, $q = 19$ болсын (тексереміз $11 \bmod 4 = 3$, $19 \bmod 4 = 3$). Онда $M = p \cdot q = 11 \cdot 19 = 209$. X-ты M өзара жай сан деп қабылдаймыз: $x = 3$ болсын. Генератордың сөрелік санын есептейміз x_0:

$$x_0 = x^2 \bmod M = 3^2 \bmod 209 = 9 \bmod 209 = 9.$$

BBS алгоритмы бойынша бірінші он санды есептейміз x_i. Кездейсоқ биттер ретінде x_i екілік түріндегі кіші битын аламыз:

$x_1 = 9^2 \bmod 209 = 81 \bmod 209 = 81$ – кіші бит: *1*;

$x_2 = 81^2 \bmod 209 = 6561 \bmod 209 = 82$ – кіші бит: *0*;

$x_3 = 82^2 \bmod 209 = 6724 \bmod 209 = 36$ – кіші бит: *0*;

$x_4 = 36^2 \bmod 209 = 1296 \bmod 209 = 42$ – кіші бит: *0*;

$x_5 = 42^2 \bmod 209 = 1764 \bmod 209 = 92$ – кіші бит: *0;*

$x_6 = 92^2 \bmod 209 = 8464 \bmod 209 = 104$ – кіші бит: *0;*

$x_7 = 104^2 \bmod 209 = 10816 \bmod 209 = 157$ – кіші бит: *1*;

$x_8 = 157^2 \bmod 209 = 24649 \bmod 209 = 196$ – кіші бит: *0*;

$x_9 = 196^2 \bmod 209 = 38416 \bmod 209 = 169$ – кіші бит: *1*;

$x_{10} = 169^2 \bmod 209 = 28561 \bmod 209 = 137$ – кіші бит: *1*.

Бұл әдістің тәжірибелік мақсаттар үшін ең қызықты қасиеті тізбектің *n*-ші санын табу үшін x_i алдынғы *n* санын есептеу қажет емес. x_n тікелей өрнек арқылы есептеуге болады екен

$$x_n = x_0^{2^{n \bmod ((p-1)(q-1))}} \bmod M .$$

Мысалы, x_0 – ден x_{10} есептейік:

$$x_{10} = x_0^{2^{10 \bmod ((11-1)(19-1))}} \bmod 209 = 9^{1024 \bmod 180} \bmod 209 = 137.$$

Нәтижесінде реттік есептеумен алынған мәнге тең мәнді алдық, – *137*. Есептеулер қыйын сияқты көрінеді, бірақ оларды кейбір процедура немесе бағдарлама ретінде қалыптастырып қажет жағдайда қолдануға болады.

«Тікелей» x_n алу мүмкіндігі *BBS* алгоритмін ағынды шифрлау кезінде қолдануға мүмкіндік береді, мысалы, ерікті қол жеткізуі бар файлдар немесе дерекқор жазбалары бар файл үзінділері үшін.

BBS алгоритмінің қауіпсіздігі үлкен *M* санын көбейткіштерге бөлу қиындығында негізделген. Егер *M* жеткілікті үлкен болса, оны құпия сақтау қажет емес деп айтылады; *M* көбейткіштерге бөлмегенше *ПКСГ* шығысын ешкім болжай алмайды. Бұл $n = p \cdot q$ (*p* және *q* – жай сандар) сандарын көбейткіштерге бөлу есебі өте қиын

болғанымен байланысты, егер тек n белгілі болса, ал p және q – бірнеше ондық немесе жүздік биттерден тұратын үлкен сан болса (бұл *факторизациялау есебі* деп аталады).

Осыған қоса, қасқой *BBS* генераторымен генерацияланған кейбір тізбекті білсе, алдынғы және келесі биттерді анықтай алмайтынын дәлелдеуге болады. *BBS* генераторы оң және сол жақ бағытта болжамды емес. Бұл қасиет криптография мақсаттары үшін өте қажетті және ол M санын көбейткіштерге бөлу ерекшеліктерімен байланысты.

BBS алгоритмінің ең үлкен кемшілігі – ол қажетті деңгейде жылдам емес, бұл оны көптеген салаларда қолдануға болмайтынына алып келеді, мысалы, реалды уақытта есептеулерде, сонымен қатар ағынды шифрлауда.

Алайда бұл алгоритм үлкен периоды (алғашқы параметрлерін сәйкес таңдауда) бар жақсы *ПКТ* береді. Бұл оны криптографиялық мақсаттарда шифрлау үшін кілттерді генерациялау кезінде қолдануға мүмкіндік береді.

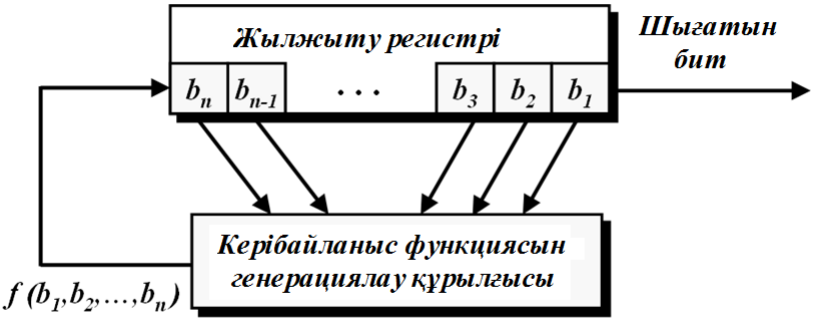
4.2.4. Кері байланысы бар жылжымалы регистрлер негізінде псевдокездейсоқ сандар генераторы

Кодтау теориясында және криптографияда *кері байланысы бар жылжымалы регистрлер* кең қолданылады. Олар шифрлау аппараттарында ЭЕМ және заманау жоғары жылдамдықты бағдарламалық шифрлауыштар кең қолданудың алдында қолданылды.

Кері байланысы бар жылжымалы регистрлер псевдокездейсоқ биттер ағынын алу үшін қолдану мүмкін. Кері байланысы бар жылжымалы регистр екі бөліктен тұрады: n-биттік жылжымалы регистрден және кері байланыс құрылғысынан (4.2 сурет).

Регистрден биттерді тек бір бірден (ретпен) алуға болады. Егер келесі битті алу керек болса, онда регистрдің барлық биттері бір разрядқа оң жаққа жылжыйды. Регистрдің кірісіне сол жақтан жаңа бит кіреді, ол кері байланыс құрылғысымен қалыптастырылады және жылжымалы регистрдің барлық қалған биттерінен тәуелді. Осының есебінен регистрдің биттері анықталған заң бойынша өзгереді, ол *ПКТ* алу сұлбасын анықтайды. Регистр жұмысының тактарының кейбір санынан кейін биттер тізбегі қайталанып

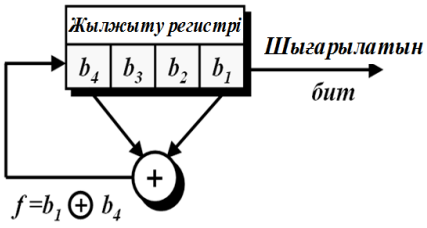
бастайтыны түсінікті. Қайталанудың алдындағы алынатын тізбек ұзындығы *жылжымалы регистрдің периоды* деп аталады.



4.2 сурет – Кері байланысы бар жылжымалы регистр

Жылжымалы регистрлерін қолданумен ағынды шифрлар тәжірибеде ұзақ қолданды. Бұл олар цифрлық құрылғылар көмегімен өте жақсы жүзеге асатынымен байланысты.

Кері байланысы бар жылжымалы регистрдің қарапайым түрі *кері байланысы бар сызықты жылжымалы региср (КБСЖР)* (*linear feedback shift register – LFSR*). Бұл құрылғының кері байланысы регистрдің барлық (немесе кейбір) биттерінің *2* модулі бойынша қосындысы ретінде жүзеге асады. Кері байланысқа қатысатын биттер *отводтық тізбекті* құрайды. Кері байланысы бар сызықты жылжымалы регистрлер немесе олардың модификациялары криптографияда жиі қолданылады.



4.3 сурет - *4*-разрядтық сызықты жылжымалы регистр мысалы

Кері байланысы бар жылжымалы регистр жұмысы түсінікті болу үшін, 4.3 суретінде көрсетілген бірінші және төртінші разрядтардан отводы бар *4*-биттік *LFSR* қарастырайық.

4.3 суретінде көрсетілген регистрге *1011* – алғашқы мәнді жазамыз. Регистрдің іш-

кі күйлер тізбегін есептеу 4.1 кестеде көрсетілген кесте көмегімен ыңғайлы. Кестеде регистрдің бірінші тоғыз күйі көрсетілген.

4.1 кесте

Сызықты жылжымалы регистрдің жұмыс реті

Күй нөмірі	Регистрдің ішкі күйі b_4, b_3, b_2, b_1	Кері байланыс функциясын есептеу нәтижесі $f = b_1 \oplus b_4$	Шығатын бит (b_1)
0	*1 0 1 1*	*0*	*1*
1	*0 1 0 1*	*1*	*1*
2	*1 0 1 0*	*1*	*0*
3	*1 1 0 1*	*0*	*1*
4	*0 1 1 0*	*0*	*0*
5	*0 0 1 1*	*1*	*1*
6	*1 0 0 1*	*0*	*1*
7	*0 1 0 0*	*0*	*0*
8	*0 0 1 0*	*0*	*0*

Әр қадамда регисрдегі барлық сандар оң жақа бір разрядқа жылжыйды. Нәтиже ретінде бір битті алуға болады. Сол жақта босаған орынға кері байланыс функциясын $f = b_1 \oplus b_4$ есептеу нәтижесіне тең бит кіреді. Псевдокездейсоқ биттер генераторының шығыс тізбегін кестенің соңғы бағаны (шығатын бит) құрайды.

Көлемі *n* бит сызықты жылжымалы регистр $2^n–1$ күйлерінің біреуінде болу мүмкін. Сондықтан теориялық түрде бұндай регистр генерациялайтын псевдокездейсоқ тізбегінің максималды периоды $2^n–1$.

Кері байланысы бар сызықты жылжымалы регистр максималды периоды бар биттер циклдық тізбегін тек отводтық тізбек ретінде анықталған биттерді таңдаса ғана генерациялайды. Отводтық тізбек биттері үшін қажетті разрядтар нөмірін таңдау мүмкіндігін беретін математикалық теория құрылған [3, 8, 9].

Кері байланысы бар сызықты жылжымалы регистрлер деректер ағынын шифрлау үшін жиі қолданылды және кәзірге дейін қолданылады. Осындай шифрлау құрылғыларында криптографиялық беріктілікті жоғарлату үшін бірнеше кері байланысы бар сызықты жылжымалы регистр комбинациясын қолданады, сонымен қатар қосымша араластыру операциялар

енгізіледі. Осындай электронды сұлбалар Екінші әлемдік соғыс алдында ұсынылған және шығарылған. XX ғасырдың соңында құрылған кейбір ағынды шифрларға ұқсас принциптер салынған, мысалы, *GSM* стандартты ұялы цифрлық арналарын шифрлау үшін Еуропада қолданған *A5* алгоритміне. Кейбір криптографиялық талдаушылар кері байланысы бар сызықты жылжымалы регистрлерді қолданумен ағынды шифрлардың сенімділігіне сенімсіздік білдіргеніне қарамастан, олар кәзіргі уақытқа дейін қолданатын әртүрлі әскери және азаматтық байланыс құрылғыларының жұмыс істеу негізіне орнатылған [32].

Сызықты жылжымалы регистрлер негізінде *ПКСГ* негізгі кемшілігі бағдарламалық жүзеге асырудың қыйындығы. Жылжытулар және биттік операциялар электронды құрылғыларда тез әрі оңай орындалады, сондықтан әртүрлі мемлекеттерде кері байланысы бар жылжымалы регистрлерді қолданумен алгоритмдер негізінде ағынды шифрлау үшін құрылғылар және шағынсұлбалар шығарылады.

4.3. АҒЫНДЫ ШИФРЛАРЫН ЖІКТЕУ

Мысалы, ағынды шифрлар үшін гаммалау режимінде байланыс арнасы бойынша жіберу кезінде шифрланған хабарда бір белгіде қате кетсін. Бұл жағдайда қатесіз қабылданған барлық белгілер дұрыс керішифрланатыны айқын. Хабардың тек бір белгісін жоғалту болады. Ал енді байланыс арнасы бойынша жіберу кезінде шифрланған хабардың белгілерінің біреуі жоғалсын дейік. Бұл, жоғалған белгіден кейін барлық хабардың дұрыс емес керішифрлауына алып келеді. Ағынды шифрлау жүйелері үшін барлық деректерді жіберу арналарында бөгеттер болады. Сондықтан ақпаратты жоғалтпау үшін хабарды шифрлауын және керішифрлауын синхрондау проблемасын шешеді. Бұл проблеманы шешу тәсілі бойынша криптографиялық жүйелерді синхронды жүйелер және өзісинхрондалатын жүйелерге бөледі.

4.3.1. Синхронды ағынды шифрлар

Синхронды ағынды шифрлар (САШ) – бұл шифрларда кілттер ағыны ашық және шифрланған хабарға тәуелсіз генерацияланады.

Шифрлау кезінде кілттер ағынының генераторы кілттер ағынының биттерін береді, олар керішифрлау кезіндегі кілттер ағынының биттеріне тең. Шифрланған хабардың белгісінің жоғалуы осы екі генераторының арасында синхрондалуының бұзылуына және қалған хабарды керішифрлауға мүмкіндік бермеуіне алып келеді. Бұл жағдайда жіберуші және қабылдаушы жұмысты жалғастыру үшін қайтадан синхрондалу қажет.

Әдетте синхрондау жіберілетін хабарға арнайы маркерлерді енгізумен жасалады. Нәтижесінде жіберу кезінде түсіп қалған белгі маркердің біреуі қабылданғанша дұрыс емес керішифрлауға алып келеді.

Синхрондау кілт ағынының бір де бір бөлігі қайталанбау шартымен өткізілу керек. Сондықтан генераторды алдынғы күйге ауыстырудың маңызы жоқ.

САШ негізгі артықшылықтары:

- қателерді тарату болмау (тек қатесі бар бит дұрыс емес керішифрланады);

- шифрланған хабарды кез келген ендіру мен жоюдан қорғайды, олар синхрондаудың жоғалуына алып келеді және көрініп қалады.

САШ кемшіліктері: шифрланған хабардың бөлек биттерін өзгертуіне қарсы тұра алмайды. Егер қасқойға ашық хабар белгілі болса, ол биттерді өзіне керек түрде керішифрланатындай өзгерту мүмкін.

4.3.2. Өзісинхрондалатын ағынды шифрлар

Өзісинхрондалатын ағынды шифрлар (асинхронды ағынды шифрлар (ААШ)) – бұл шифрларда кілт ағыны кілт және шифрланған хабардың белгіленген тіркелген саны функциясымен құрылады.

Сонымен, кілт ағыны генераторының ішкі күйі шифрланған хабардың алдынғы *n* бит функциясы болады. Сондықтан *n* битті қабылдап, керішифрлауды орындайтын кілттер ағыны генераторы шифрлауды орындайтын кілттер ағыны генераторымен автоматты түрде синхрондалады.

Бұл режимді жүзеге асыру келесі ретпен жасалады: әр хабар *n* бит ұзындығы бар кездейсоқ атаумен (заголовок) басталады; атау шифрланады, жіберіледі және керішифрланады; керішифрлау дұрыс

емес болады, бірақ содан кейін екі генератор да синхрондалған болады.

ААШ негізгі артықшылығы ашық хабардың статистикасын араластыру болады. Ашық хабардың әр белгісі келесі шифрланған хабарға әсер еткендіктен, ашық хабардың статистикалық қасиеттері барлық шифрланған хабарға таралады. Сондықтан, *ААШ САШ-қа* қарағанда ашық мәтін артықтығы негізіндегі шабуылдарға беріктірек болады.

ААШ кемшіліктері:

- қате таралауы (шифрланған хабардың әр дұрыс емес битіне ашық хабарда *n* бит сәйкес келеді);
- қайта жіберумен ашуға сезімшіл.

4.4. А5 АҒЫНДЫ ШИФР

4.4.1. А5 ағынды шифрын құру тарихы

А5 деген *GSM* (*Group Special Mobile*) еуропалық мобильды цифрлық байланысында телефон және базалық бекет арасында жіберілетін деректерінің конфиденциалдығын қамтамасыз етуге қолданатын ағынды шифрлау алгоритмі.

Шифр шифрланатын ақпаратты және генерацияланатын псевдокездейсоқ тізбекті биттер бойынша *2* модулімен (*xor* буль операциясы) қосуда негізделген. *А5* псевдокездейсоқ тізбек үш кері байланысы бар сызықты жылжымалы регистрлер негізінде жүзеге асырылады. Регистрлердің разрядтар саны (ұзындығы) *19, 22* және *23* бит сәйкес. Әр қадам сайын кем дегенде екі регистрдің жылжуын ұйымдастыратын арнайы сұлба басқарады. Бұл ол тізбектер мазмұнын теңөлшемді емес жылжуына алып келеді. Тізбек регистрлердің шығыс биттеріне *xor* операцияны қолданумен қалыптастырылады.

Алдында француз әскерінің криптограф-мамандары ағынды шифрды тек әскері мақсаттарында қолдану үшін құрды. XX ғасырдың 80-ші жылдарының аяғында *GSM* стандарты үшін жаңа, заманауи қауіпсіздік жүйесін құру талап етілді. Оның негізіне үш құпия алгоритм жатты: аутентификация – *А3*; ағынды шифрлау – *А5*, сеанс кілтін генерациялау – *А8*. *А5* алгоритм ретінде француздық жүзеге асырылыу қабылданды. Бұл шифр ағынның қорғалғандығын жеткілікті жақсы қамтамасыз етті, ол деген

сөйлесудің конфиденциалдығын қамтамасыз етті. Алдында Еуропадан стандартты экспорттау ойда болмаған, бірақ уақыт өте келе ода қажеттілік туындады. Сондықтан, *A5* атын *A5/1* ауыстырды және Еуропа және АҚШ тарату басталды. Басқа мемлекеттер үшін алгоритм өзгертілді, оның криптографиялық беріктілігін төмендетті. *A5/2* Еуропалық одаққа кірмейтін мемлекеттер үшін экспортты нұсқа ретінде арнайы құрылған. *A5/0* шифрлау мүлдем жоқ. Кәзіргі уақытта *Касуми алгоритміне* негізделген *A5/3* алгоритм құрылған және *3G* желісінде қолдануға бекітілген. Бұл өзгертулерді *A5/x* деп белгілейді.

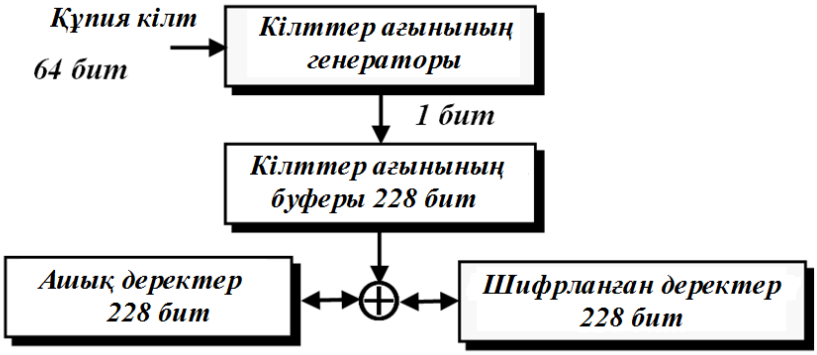
Ресмі түрде берілген криптографиялық сұлба жарияланымға шықпаған және оның құрылымы жариялыққа шығарылмаған. Бұл құрушылар белгісіздік есебінде қауіпсіздікке сүйенгеніне байланысты, яғни егер алгоритмдер жариялыққа шықпаса оларды бұзу қыйын. Деректер *GSM* операторларына тек қажеттілік болғанда ғана берілді. Осыған қарамастан, 1994 жылға *A5* алгоритмінің барлық егжей-тегжейлері белгілі болды: британдық телефондық компания (*British Telecom*) стандартқа тиесілі барлық құжаттаманы ақпаратты жарияламау келімісі сіз Брэдфордтық университетке талдау үшін берді. Осыған қоса, стандарт туралы материалдар Қытайдың бір конференциясында пайда болды. Нәтижесінде, оның сұлбасы бірте-бірте кең шеңберге таралды. Сол жылы кембриджідік ғалымдар *Росс Андерсон* (*Ross Anderson*) және *Майкл Рое* (*Michael Roe*) сол деректер бойынша қалпына келтірілген криптографиялық сұлбаны жариялап, оның криптографиялық беріктілігіне баға берді. Ақыры алгоритм *Йован Голич* жұмысында *Eurocrypt'97* конференциясында ұсынылды [3, 7].

4.4.2. A5 көмегімен деректерді ағынды шифрлау

Кәзіргі уақытта *A5* алгоритмі – шифрлардың тұтас тұқымдасы. Қарастыру үшін *A5/1* (4.4 сурет) осы тұқымның басы ретінде аламыз. Алгоритмнің басқа нұсқалардағы өзгерістерін бөлек көрсетеміз.

A5/1 ауқымды мобильді байланыс жүйесінде (*GSM*) қолданылады. *GSM* телефондық байланыс *228* битке кадрлар тізбегі ретінде жүзеге асырылады, әр кадр – *4,6* миллисекунд. *A5/1 64* биттік кілтке негізделіп бит ағынын құрады. Разрядты ағындар *228*

бит бойынша буферлерге жиналады, оларды 4.4 суретінде көрсетілгендей *228* биттік кадрмен *2* модулі бойынша қосу үшін.



4.4 сурет - *A5/1* ағынды шифрлау алгоритмін құру принципі

4.4 суретінде екі жақа қараған тілдер шифрлау кезінде ашық деректер қабылданатынын, ал кілт ағынымен *2* модулі бойынша қосудан кейін шифрланған деректер шығатынын көрсетеді; керішифрлау кезінде шифрланған деректерді алады, ал кілт ағыны мен *2* модулі бойынша қосудан кейін ашық деректер шығады.

Бұл алгоритмде әр ашық кілт символына шифрланған хабар символы сәйкес келеді. Хабар блоктарға бөлінбейді (блокты шифрлау кезіндегідей) және көлемі өзгермейді. Аппаратты жүзеге асыруды жеңілдету үшін, сонымен қатар жылдамдығын жоғарлату үшін тек қарапайым операциялар орындалады: *2* модулі бойынша қосу (*xor*) және регистр биттерін жылжыту.

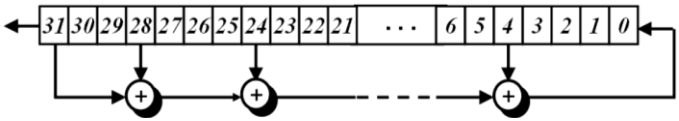
Шығыс тізбекті қалыптастыру алғашқы хабар ағынын генерацияланатын тізбекпен (гаммамен) қосу жолымен өтеді. *Xor* операцияның ерекшелігі келесіде: егер оны жұп саңға тең рет қолданса, ол алғашқы мәнге алып келеді. Осыдан, керішифрлау шифрланған хабарды белгілі тізбекпен (гаммамен) қосу жолымен өтеді.

Сонымен, *A5* шифрдың қауіпсіздігі тізбек қасиеттерінен толық тәуелді. Идеалды жағдайда гамманың әр биты – тәуелсіз кездейсоқ шама, ал тізбектің өзі кездейсоқ болады. Осындай сұлбаны 1917 жылы *Вернам* ойлап тапты және соның атымен аталды. 1949 жылы *Клод Шеннон* дәлелдегендей бұл абсолютті криптографиялық

беріктілікті қамтамасыз етеді. Бірақ кездейсоқ тізбекті қолдану оның көлемі ашық мәтінге тең болатынын және қорғалған арнамен жіберу керектігін білдіреді, ол деген есепті қатты қиындатады және тәжірибелік түрде еш жерде қолданылмайды.

Реалды жүйелерде берілген өлшемді кілт құрылады, ол қиындықсыз жабық арнамен жіберіледі. Оның негізінде тізбек генерацияланады және ол псевдокездейсоқ болады. Ағынды шифрлардың үлкен класын (соның ішінде *A5*) псевдокездейсоқ тізбек генераторы сызықты кері байланысы бар жылжымалы регистрлерде (*LFSR*) негізделген шифрлар құрады.

LFSR регистрден (берілген ұзындығымен биттер тізбегі) және кері байланысынан тұрады (4.5 сурет).



4.5 сурет – Кері байланыстың көпмүшесімен $x^{32}+x^{29}+x^{25}+x^5+1$ *LFSR*

Әр такта келесі әрекеттер өткізіледі: шеткі сол жақтағы бит (жоғарғы бит) алынады, регистрдегі тізбек сол жаққа жылжыйды және босаған оң жақтағы ұяшыққа (кіші бит) кері байланыс функцияның нәтижесі жазылады. Бұл функция регистрлердің анықталған биттерін *2* модулі бойынша қосу болады және ол көпмүше ретінде жазылады, дәреже бит нөмірін көрсетеді. Алынған биттер шығыс тізбекті қалыптастырады.

LFSR үшін негізгі көрсеткіш псевдокездейсоқ тізбегінің периоды. Егер кері байланыс функциясының көпмүшесі *2* модулі бойынша қарапайым болса, онда ол максималды болады (және 2^n-1 тең). Бұл жағдайда шығыс тізбегі *M*-тізбегі деп аталады (максималды мүмкінді қайталанбайтын ұзындықты тізбек).

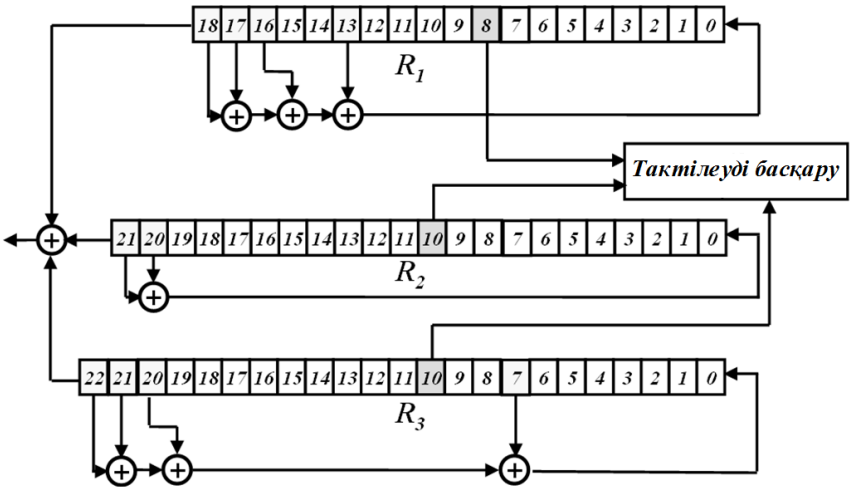
КБСЖР өзі криптографиялық талдауға оңай беріледі және шифрлауда қолдану үшін жеткілікті сенімді емес. Тәжірибелік қолдануда айнымалы тактілеу регистрлер жүйелері, оларда әртүрлі ұзындықтары және кері байланыс функциялары бар.

A5 ағынды шифр сұлбасына кіреді (4.6 сурет):

- *19, 22* және *23* разряд ұзындықтары бар үш регистр, олар үшін кері байланыс көпмүшелері:

- R_1 үшін – $x^{19} + x^{18} + x^{17} + x^{14} + 1$;

- R_2 үшін – $x^{22} + x^{21} + 1$;
- R_3 үшін – $x^{23} + x^{22} + x^{21} + x^8 + 1$;
- тактілеуді басқару сұлбасы.



4.6 сурет - *A5/1* алгоритмінде регистр жүйесінің сұлбасы

Тактілеуді басқару үшін әр регистрде синхрондау биттері бар: *8* (R_1), *10* (R_2), *10* (R_3). Осы биттерді қолданумен келесі функция есептеледі:

$$f = x \,\&\, y \mid x \,\&\, z \mid y \,\&\, z,$$

бұл жерде *&* – бульдік *and*; | – бульдік *or*; *x*, *y* және *z* – R_1, R_2 және R_3 синхрондау биттері.

Функция *f* есептегеннен кейін тек синхрондау биты *f* тең регистрлердің ғана разрядтары жылжыйды. Шынында, синхробиттері көбісіне жататын регистрлердің ғана биттері жылжыйды.

Жүйенің шығыс биты регистрлердің шығыс биттеріне *xor* операцияның нәтижесі.

Белгілі сұлба негізінде (4.6 сурет) алгоритм жұмысының ерекшеліктерін қарастырамыз. Деректерді жіберу құрылымдастырылған түрде жүзеге асады – кадрларға бөлумен (*114* бит). Инициализациялаудың алдында регистрлер нөлденеді,

алгоритм кірісіне келеді: *A8* алгоритмімен қалыптастырылған сеанс кілті (*K – 64* бит) және кадр нөмірі (*Fn – 22* бит). Ары қарай келесі әрекеттер ретімен орындалады: шифрлау немесе керішифрлау үшін қолданатын бірбиттік шығыс тактілік импульстармен *228* биттік буферді қамтамасыз етеді.

Инициализациялау. Инициализациялау шифрлау немесе керішифрлаудың әр кадырына орындалады, *64* биттік кілт және сәйкес кадр нөмірінің *22* биті қолданады. Инициализациялау кезінде келесі қадамдар орындалады:

1. Алдымен үш сызықты жылжымалы регистрлерінің барлық биттеріне *0* жазылады (нөлденеді).

2. *64* биттік кілт регистр мәнімен келесі псевдокодқа сәйкес араластырылады (әр сызықты регистр бір қадамға жылжыйды, яғни синхрондау қамтамасыз етіледі):

```
for (i = 0 to 63)
{
    K[i] кілтін барлық үш регистрінің шеткі сол жақтағы
    биттерімен 2 модулі бойынша қосу.
    Барлық үш сызықты жылжу регистрлерін синхрондауы
}
```

3. Алдынғы процессті қайталау, бірақ псевдокодқа сәйкес *22*-биттік кадрды қолдану:

```
for (i = 0 to 21)
{
    Кадр нөмірін (i)  барлық үш регистрінің шеткі сол жақтағы
    биттерімен 2 модулі бойынша қосу.
    Барлық үш сызықты жылжу регистрлерін синхрондауы
    }
```

4. *100* циклда осы псевдокодқа сәйкес барлық генератор синхрондалады, сонымен бірге қай сызықты жылжу регистрі синхрондалатынын анықтау үшін *мажоритарлық функция* қолданылады.

```
for (i = 0 to 99)
{
    Мажоритарлық функция негізінде барлық генераторды
    синхрондау
}
```

Кей кезде бұл жерде синхрондау екі немесе барлық үш сызықты жылжу регистрі жылжытынын білдіреді.

Мажоритарлық функция. Егер биттердің көбісінің мәндері *1* тең болса, онда (*x, y, z*) параметрлері бар мажоритарлық функция (*f*) мәні *1* тең болады; егер бұл – *0,* онда *0* болады. Мысалы, *f(1,0,1) = 1*, бірақ *f(0,0,1) = 0*. Мажоритарлық функцияның мәні тактілік импульстің келу алдында анықталады; үш кіріс бит синхрондайтын биттер деп аталады: егер оң жақтағы шеткі бит нөлге тең болса, онда олар сызықты регистрлер $R_1[10]$, $R_2[11]$ және $R_3[11]$ биттері. Әдебиетте бұл биттер $R_1[8]$, $R_2[10]$ және $R_3[10]$ сол жақтан саналатынына (4.6 суретінде көрсетілгендей) назар аудару қажет. Сызықты регистрлер $R_1[10]$, $R_2[11]$ және $R_3[11]$ биттерін оң жақтан қарастырамыз. Бұл шарт биттің сипаттамалық полиномдағы орнына сәйкес.

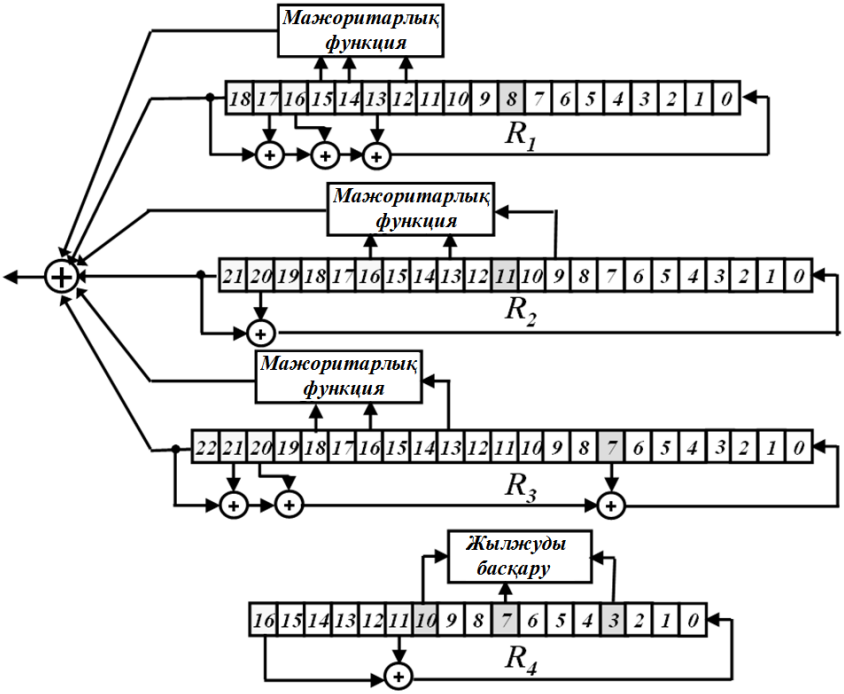
Ағынның кілттік биттері. Кілт генераторы әр тактілік импульс бойынша бір биттік кілттік ағынды құрады. Кілтті құру алдында мажоритарлық функция есептеледі. Содан кейін егер сызықты жылжымалы регистрдің синхрондау биты мажоритарлық функция нәтижесіне сәйкес болса, онда әр регистр синхрондалады; басқа уақытта синхрондалмайды.

4.4 мысал. Кейбір уақыт сәтінде $R_1[10]$, $R_2[11]$ және $R_3[11]$ синхрондау биттері *1, 0* және *1* тең. Жылжымалы регистрлер мазмұны қандай болу керек?

Шешім: Мажоритарлық функция нәтижесі: *f(1,0,1) = 1*. Сондықтан, R_1 және R_3 регистрлердің мазмұны жылжыйды, ал R_2 – жоқ.

Шифрлау/керішифрлау. Кілттер генераторымен құрылған разрядтты ағындар ары қарай *228* биттік кілтті қалыптастыру үшін буферге жазылады. Содан кейін шифрланған деректер кадрын құру үшін кілт алғашқы деректер кадрымен *2* модулі бойынша қосылады. Сонымен бірге бір кадрды шифрлау/керішифрлау орындалады.

A5/2 алгоритміне тағы бір *17* биттік (R_4) регистр қосылған, ол басқалардың жылжыуын (тактілеуін) басқарады (4.7 сурет).



4.7 сурет - *A5/2* алгоритмінде регистрлер жүйесінің құрылымы

Құрылымның өзгерулері келесі:

- *17* бит ұзындығы бар кері байланыстың көпмүшесі R_4 үшін – $x^{17} + x^{10} + 1$ тең R_4 регистрі қосылған;

- тактілеуді басқаратын R_4 регистрі, ол үшін *3, 7, 10* биттері – синхрондау биттері;

- есептелетін мажоритарлық функция

$$f = x \,\&\, y \mid x \,\&\, z \mid y \,\&\, z,$$

бұл жерде *&* – бульдік *and*; | - бульдік *or*; *x*, *y* және *z* – синхрондау биттері R_4 (*3*-ші бит), R_4 (*7*-ші бит) и R_4 (*10*-ші бит).

Регистрдегі биттер жылжыуы келесі жағдайда орындалады:

- егер $R_4\ (10) = f$, онда R_1 регистрінің биттері жылжыйды;

- егер $R_4\ (3) = f$, онда R_2 регистрінің биттері жылжыйды;

- егер $R_4\ (7) = f$, онда R_3 регистрінің биттері жылжыйды.

Шынында, тек синхробиты көпшілікке жататын регистрлердің биттері жылжыйды.

Жүйенің шығыс биты регистрлердің жоғарғы биттері мен регистрлердің анықталған биттерінен мажоритарлық функциялардың *xor* операцияның нәтижесі болады:

- R_1 регистрі – *12*, *14* және *15* биттері;
- R_2 регистрі – *9*, *13* және *16* биттері;
- R_3 регистрі – *13*, *16* және *18* биттері.

Жұмыс жасауда өзгерістер мәнді емес және тек инициализацияға қатысты:

- *64* + *22* такт сеанстық кілтпен және кадр нөмірімен толтырылады солай R_4;
- бір такт: $R_4(3)$, $R_4(7)$ және $R_4(10)$ бірлермен толтырылады;
- *99* такт регистрлер жылжыун басқарумен, бірақ тізбекті генерациялаусыз.

Инициализация сондай уақыт алатыны көрініп тұр (*100* такт генерациялау сыз екі бөлікке бөлінген).

A5/3 алгоритмі 2001 жылы құрылған және мобильді жүйелердің үшінші буынында *A5/1* орнын басу керек. Ол *Касуми* алгоритмі деп аталады. Оны құру кезінде негіз ретінде *Mitsubishi* корпорацияның *MISTY* шифры алынған. Кәзіргі уақытта *A5/3* талап етілген беріктілікті қамтамасыз етеді деп есептеледі.

A5/0 алгоритмінде шифрлау жоқ.

4.4.3. A5 ағынды шифрдың криптографиялық беріктілігі

GSM стандартын құру бұзуға бейімделмеген (әсіресе реалды уақытта) шифрлаудың құатты аппаратын түсінетін. Қолданатын зерттемелер қажетті жүзеге асыру кезінде жіберілетін деректердің сапалы шифрлауын қамтамасыз еткен. Осындай ақпаратты осы стандартты тарататын компаниялардан алуға болады. Бірақ маңызды нюансты белгілеу қажет: сөйлесулерді тыңдау – арнайы қазметтермен қолданатын ажырамас атрибуты. Олар өз мақсаттары үшін телефонды сөйлесулерді тыңдау мүмкіндігінде қызығушылық білдірген. Сондықтан алгоритмге қолайлы уақытта бұзу мүмкіндігін беретін өзгерістер енгізілді. Сонымен қоса, экспорт үшін *A5-mi A5/2* түрлендірді. *MoU-да* (*Memorandum of Understand Group Special Mobile standard*) *A5/2* құру мақсаты шифрлаудың криптографиялық беріктілігін төмендету екенін мойындайды, бірақ ресмі тестілеу

нәтижелерінде алгоритмнің кемшіліктері туралы белгісіз деп айтылады [32].

A5 стандарты туралы ақпараттың пайда болуымен, алгоритмді бұзу әрекеттер, сонымен қатар әлсіздіктерді іздеу басталды. Үлкен рөлді қорғауды шұғыл әлсіздендіретін стандарттың ерекшеліктері берді, олар:

- кілттің *10* биты ықтиярсыз нөлденген;
- регистрлер арасында айқасқан байланыстардың болмауы (жылжуды басқарудан басқа);
- криптографиялық талдаушыға белгілі шифрланатын қызметтік ақпараттың артық артықшылығы;
- кілттің 40% астамы генерациялайтын тізбек периодының минималды ұзындығына алып келеді, ол деген $(2^{23}–1)\cdot 3/4$ [32];
- сеанстың басында нөлдік хабарлармен (бір кадрдан) алмасу жүргізіледі;
- *A5/2* жылжу *10* бит ұзындығы бар бөлек регистрмен орындалады.

Алгоритмде осы «тесіктер» негізінде бұзу сұлбалары құрылған.

Кілт болып *64* бит ұзындығы бар сессиялық кілт есептеледі, кадр нөмірі белгілі деп есептеледі. Сонымен, түзу тексеруге негізделген шабуылдың қыйындығы 2^{64} тең.

Шифрдың бірінші шолулары (*Росс Андерсон* жұмысы) алгоритмнің әлсіздігін бірден көрсетті – кілттің тиімді ұзындығын кішірейтуге (*10* битті нөлдеу) байланысты қыйындық 2^{45} (бірден *6* деңгейге) дейін төмендеді. *Андерсон* шабуылы қысқа регистрлердің алғашқы толтыру туралы болжамға және шығу деректері бойынша үшінші толтыруды алуға негізделген.

1997 жылы *Йован Голич A4* талдау нәтижелерін жариялады. Ол регистрлердің алғашқы толтыруларын белгілі *64* биттік гамма кесіндісі бойынша анықтау тәсілін ұсынды. Бұл кесіндіні нөлдік хабарлардан алады. Шабуылдың орташа қыйындығы 2^{40} [32].

1999 жылы *Вагнер* және *Голдберг* жүйені ашу үшін талдау арқылы тек R_4 алғашқы толтыруын анықтау керектігін қиындықсыз көрсетті. Тексеру нөлдік кадрлар арқылы жүзеге асырылды. Бұл шабуылдың қыйындығы 2^{17}тең, сонымен, заманауи компьютерде шифрды ашу бірнеше секундта орындалады.

1999 жылдың желтоқсанында израильдік ғалымдар тобы (*Ади Шамир*, *Алекс Бирюков*, ал содан кейін американдық *Дэвид Вагнер*

(ағыл.)) өте оңай емес, бірақ теориялық өте тиімді *A5/1* ашу алгоритмін жариялады.

4.5. RC4 АҒЫНДЫ ШИФР

4.5.1. RC4 шифрын құру тарихы

RC4 алгоритмі 1987 жылы *Р. Ривестпен* арнайы айнымалы ұзындықты кілтімен кілттік ақпарат ағыны генераторы ретінде құрылған. Ресми қысқартыуы – *Rivest Cipher 4* болса да, оны жиі *Ron's Code* дегеннің қысқартуы деп есептейді [3, 19].

Шифр коммерциялық құпия болған, бірақ 1994 жылдың қыркүйегінде оның сипаттамасы *Cypherpunks* таратуына белгісіз адаммен жіберілген [3, 19]. Көп кешікпей *RC4* сипаттамасы *sci.crypt* ньюс-тобында жарияланды. Сол жақтан кіріс коды *Интернет* желісінің көптеген сайттарына шықты. Жарияланған шифр шығыста шынайы *RC4* беретін шифрланған деректерді берді. Бұл деректер орындау кодтарын талдау нәтижесінде алынған шығар. Жарияланған шифр *RC4* қолданатын түліктермен үйлесімді, ал телеконференцияның кейбір қатысушылары, олардың айтуынша *RC4* кіріс кодына құқықтары бар, алгоритмдердің белгілері және бағдарлама құрылымында айырмашылығы болсада сәйкестігін растады.

Бұл алгоритм белгілі болғандықтан, ол ары қарай коммерциялық құпия болмайды. Бірақ, *RC4* аты *RSA* компаниясының сауда маркасы. Сондықтан кей кезде сауда маркасының иемденушісі жағынан назарлық болмау үшін, шифрды *ARCFOUR* немесе *ARC4* (*RSA* ресми түрде алгоритмді жарияламаған соң *Alleged RC4* – болжамды *RC4*) деп атайды.

RC4 кең қолдануға ие болғанның басты факторлары аппараттық және бағдарламалық жүзеге асырудың қарапайымдылығы, сонымен қатар екі жағдайда да алгоритмнің жоғары жұмыс істеу жылдамдығы.

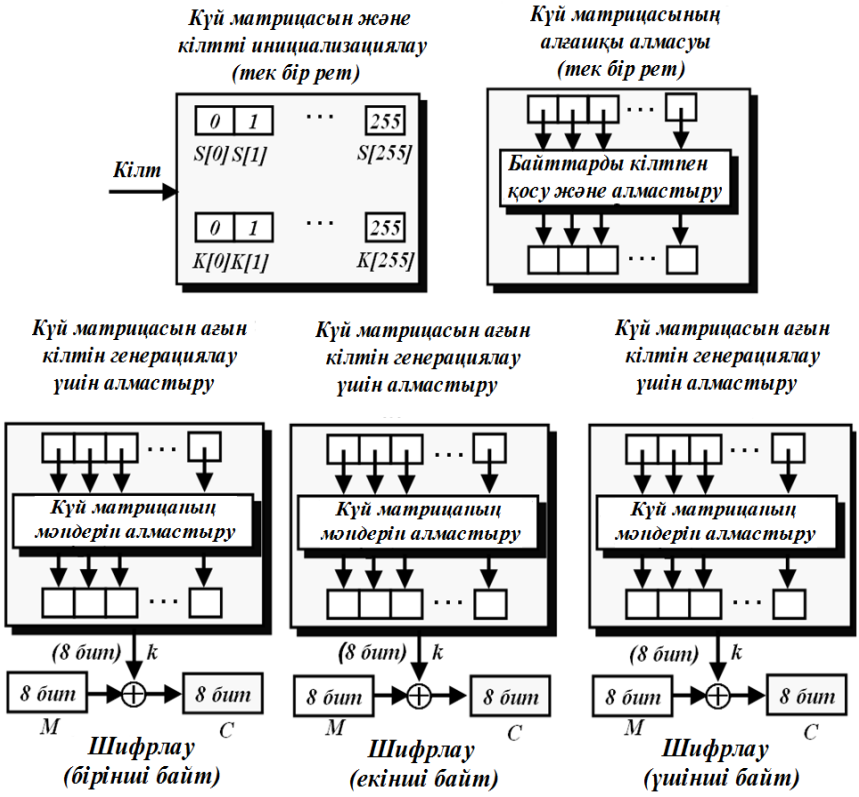
АҚШ-та мемлекет ішінде қолдану үшін кілт ұзындығы *128* бит тең болып ұсынылады, бірақ *бағдарламалық қамтама баспашы ассациация (Software Publishers Association – SPA*) және АҚШ үкіметі арасындағы бекітілген келісім бойынша *RC4* арнайы статус береді. Ол деген тек *40* битқа дейін кілт ұзындығы бар шифрларды экспорттауға рұқсат етілгенін білдіреді. *56*-биттік кілттерді

американдық компаниялардың шекараның сыртындағы бөлімдеріне қолдануға рұқсат.

4.5.2. RC4 алгоритмінің сипаттамасы

RC4 күйлер матрицалары түсінігіне базаланады. Әр сәтте күйлер матрицасы (*256* байт) активтендіріледі, одан кездейсоқ бір байт таңдалады, ол шифрлау үшін кілт болады. Идея байттар массив ретінде көрсетілу мүмкін.

S[0], S[1], S[2], ..., S[254], S[255].



4.8 сурет - *RC4* ағынды шифрды құрастыру принципі

Элементтер диапазонның индекстері – *0* және *255*аралығында. Әр элементтің мазмұны - байт (*8* бит), ол *0* ден *255* дейін бүтін сан ретінде қарастырылу мүмкін.

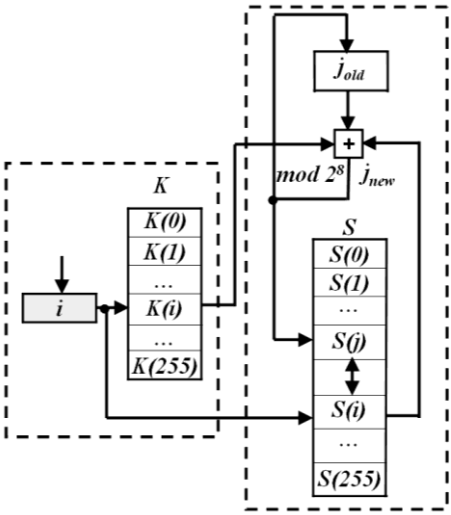
4.8 сурет *RC4* ағынды шифрды құрастыру принципін көрсетеді.

Бірінші екі блок тек бір рет орындалады (инициализациялау); ағын кілтін құру үшін алмастырулар шифрлауға арналған кіріс деректер байттары болғанша қайталанады.

RC4 алгоритмінде екі кезең бар. Бірінші, дайындау, кезеңінде ауыстыру кестесінің *S* (күй матрицалары), кілт матрицаның *K* инициализациясы орындалады, сонымен қатар *K[i]* байттар мәндеріне негізделген күй матрицаның алғашқы алмастыруы орындалады. Екінші, негізгі, кезеңде *k* псевдокездейсоқ сандары есептеледі.

Күй матрица және кілт массивінің инициализациясы

RC4 кілт ерікті ұзындығы бар байттар тізбегі, оның негізінде шифрдың алғашқы күйі *S* құрылады – барлық *256* байттарының алмасуы.



4.9 сурет – Күй матрицасы *S* және кілт массиві *K* инициализациялау сұлбасы

RC4 инициализациялау алгоритмі, сонымен қатар кілттік кесте алгоритмі деп аталатын (ағыл. *Key-Scheduling Algorithm or KSA*) 4.9 суретінде көрсетілген.

Бұл алгоритм *Key* сақталған кілтті қолданады және ұзындығы *L* байт. Инициализациялау *S* массивін толтырудан басталады, ары қарай бұл массив кілтпен анықталған алмастырулар арқылы араластырылады. Бір әрекет *S* орындалғандықтан келесі пікір орындалу керек: *S-ma* әрқашан кодтық сөздің барлық мәндері болады.

Күй матрицасы *0, 1, 2, ..., 254, 255* мәндері үшін инициализацияланады. Кілттер массиві *K[0], K[1], K[2], ..., K[254], K[255]* құрылады. Егер шифрлау кілтінде нақты *256* байт болса, байттар массивке көшіріледі; болмаса – байттар *K* массиві толғанша қайталанады.

Алдымен *S* күй матрицасы *0* ден *255* дейін тізбекті мәндермен толтырылады. Содан кейін *S* кезекті элементі *K* кілт элементімен анықталған элементпен орын ауыстырады, элементтің өзімен және алдынғы итерацияларда орын ауыстырған элементтердің нөмірлер қосындысымен. Санағыштар *i* және *j* мәндері алдымен *0* тең.

Төменде *S* күйлер матрицасын және *K* кілттер массивін инициализациялайтын, псевдокодта жазылған бағдарлама көрсетілген:

```
for ( i = 0 to 255 )
{
    S[i] ← i
    K[i] ← Key[ i mod LengthKey ]
}
```

Ары қарай күйлер матрицасында *K[i]* байттар мәндерінде негізделген алмастыру (элементтерді скремблерлеу) өткізіледі. Кілттік байт тек осы қадамда қандай элементтерді ауыстыруды анықтау үшін ғана қолданылады. Осы қадамнаң кейін матрица байттары толық араластырылған болады.

Төменде күйлер матрицаның байттарын алмастыруды жүзеге асыратын псевдокодта жазылған бағдарлама көрсетілген.

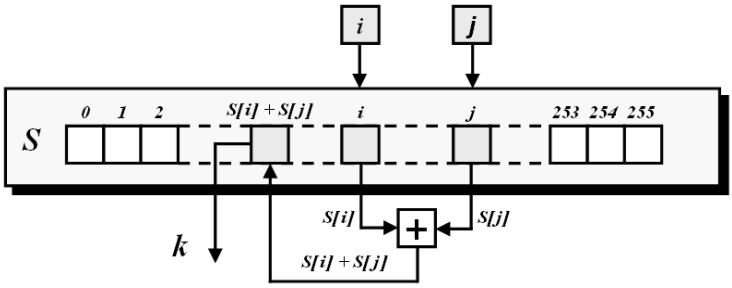
```
j ← 0
for ( i = 0 to 255 )
{
```

```
        j ← ( j + S[i] + K[i] ) mod 256
        swap ( S[i], S[j] )
}
```

Күй матрицаны алмастыру және кілттік ағынды генерациялау

Кілттік ағында k кілттер бірінің артынан бірі генерацияланады. Алдымен күйлер матрицаның элементтері өз элементтер мәні және екі жеке айнымалы і және j мәндері негізінде алмастырылады. Содан кейін күйлер матрицаның і және j позицияларында екі элементтердің мәндері k кілт ретінде қолданатын күйлер матрицаның индексін анықтау үшін қолданады. Келесі код мәтіннің алғашқы деректердің әр байты үшін кілттік ағында жаңа кілттік элементті құру үшін қайталанады.

Алгоритм өзегі кілттік ағынды генерациялау функциясынан тұрады (4.10 сурет).



4.10 сурет - *RC4* алгоритмнің өзегі

Ауыстыру кестесі (*S* күйлер матрицасы) қолдану уақытында баяу өзгереді, *i* санағышы (*i* айнымаласы) кестенің әр элементінің өзгеруін қамтамасыз етеді, ал *j* санағышы (*j* айнымалысы) кесте элементтерінің кездейсоқ өзгеретінін кепілдейді.

Бірінші итерацияның алдында *i* және *j* айнымалылары *0* инициализацияланады, бірақ мән бір итерациядан екіншіге көшіріледі.

Төменде *S[i]* күйлер матрицаның байтарын алмастыратын және *k* кілттік ағынды генерациялайтын псевдокодта жазылған бағдарлама көрсетілген:

```
i ← 0;        j ← 0
```

```
i ← ( i + 1 ) mod 256
j ← ( j + S[i] ) mod 256
swap ( S[i], S[j] )
k ← S[ (S[i] + S[j]) mod 256 ]
```

Деректерді шифрлау (керішифрлау)

Кілттік ағын *k* құрылғаннан кейін,шифрланған деректер байтын құру үшін ашық деректер байты *k* көмегімен шифрланады. Керішифрлау кері процесс болады.

Төменде *RC4* үшін деректерді шифрлау процессін көрсететін псевдокодта жазылған бағдарлама көрсетілген:

```
RC4_Encryption ( Key )
{
    // Алғашқы күйлер матрицасын
    // және кілттік байттарды құру
    for ( i = 0 to 255 )
    {
            S[i] ← i;          K[i] ← Key[ i mod LengthKey ]
    }
    // Кілт байты негізінде
    // күй матрицасының байттарын алмастыру
    j ← 0
    for ( i = 0 to 255 )
    {
            j ← ( j + S[i] + K[i] ) mod 256;     swap ( S[i], S[j] )
    }
    // Күйлер матрицасының байттарын үздіксіз алмастыру,
    // кілттік ағынды генерациялау және деректерді шифрлау
    i ← 0;   j ← 0
    while (шифрлау үшін деректер бар болғанша)
    {
            i ← ( i + 1 ) mod 256;     j ← ( j + S[i] ) mod 256
            swap ( S[i], S[j] );       k ← S[ (S[i] + S[j]) mod 256 ]
            // Кілт дайын, деректерді шифрлау
            input M
            C ← M ⊕ k
            output C
    }
}
```

Деректерді керішифрлау келесіде: кілттік ағынды (*k*) регенерациялау және оны шифрланған деректермен (*C*) *2* модулі бойынша қосу. *2* модулі бойынша қосу қасиеттері арқасында шығыста алғашқы деректерді (*M*) аламыз.

4.5 мысал. Ағын кілтінің кездейсоқтығын көрсету үшін, барлық нөлдік байттармен құпиялау кілтін қолданамыз. Бұл жағдайда *20* мән үшін кілттік ағын тең болады: *129, 163, 68, 228, 89, 71, 76, 51, 165, 213, 200, 18, 6, 239, 229, 236, 63, 52, 170, 117.*

4.6 мысал. 4.5 мысалын қайталаймыз, бірақ құпиялау кілті бес байт (*15, 202, 33, 6, 8*) болсын. Кілттік ағын: *154, 188, 65, 159, 144, 167, 200, 166, 202, 243, 119, 75, 104, 161, 206, 24, 34, 216, 114, 89.* Кілттік ағындағы кездейсоқтық сірә.

RC4 алгоритмінде деректерді түрлендіру процесстерін талдау

RC4 – блок немесе сөзбен – *n* параметрімен анықталатын алгоритмдер класына жатады. Әдетте $n = 8$, бірақ басқа да мәндерді қолдануға болады. Алгоритмді талдауды жеңілдету үшін $n = 4$ деп қабылдаймыз. *RC4* ішкі күйі 2^n сөздер массивінен және әрқайсының бір сөз көлемі бар екі санағыштан тұрады. Екі санағыш, екеуіде *4*-биттік ($n = 4$ тең кезде). Оларды *i* және *j* деп белгілейміз. Барлық есептеулер 2^n модулі бойынша өткізіледі.

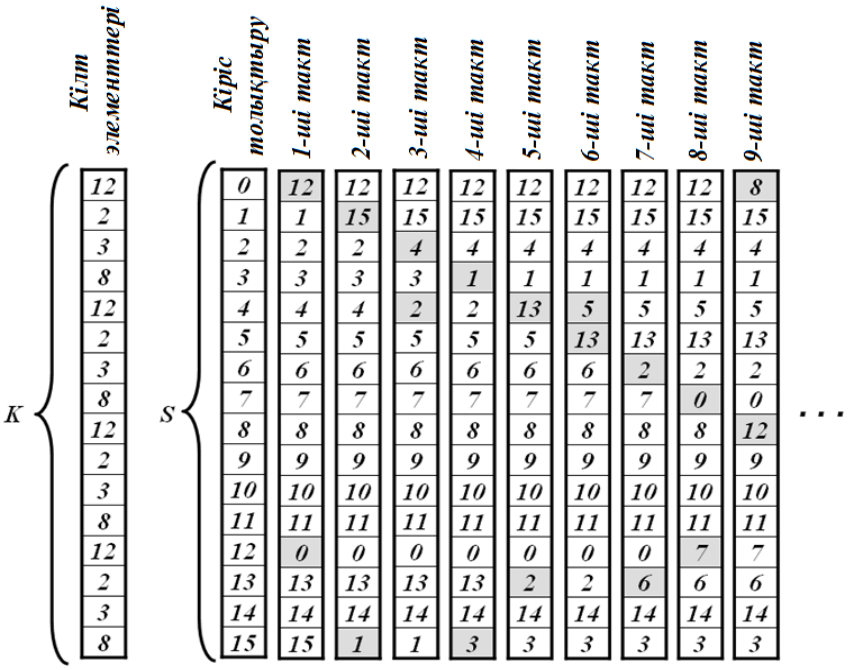
4.7 мысал. Кілттің алғашқы мәні $Key = \{12, 2, 3, 8\}$ болсын. Күйлер матрицасының және кілттер массивінің инициализациясын, сонымен қатар күйлер матрицасының алғашқы алмастыруын көрсету қажет.

Шешім. 4.11 суретінде *4*-разрядты күйлер матрицасының және кілттер массивінің инициализация және күйлер матрицаның алғашқы алмастыру (*16-дан* 9 такт) мысалы көрсетілген.

4.8 мысал. RC4 4-разрядты *ПКС* генераторы жұмысын *i* және *j* берілген мәндері және *4*-разрядты S ауыстыру кестенің берілген толтыруымен көрсету.

Шешім. RC4 4-разрядты *ПКС* генераторы жұмысының мысалы 4.12 суретінде көрсетілген.

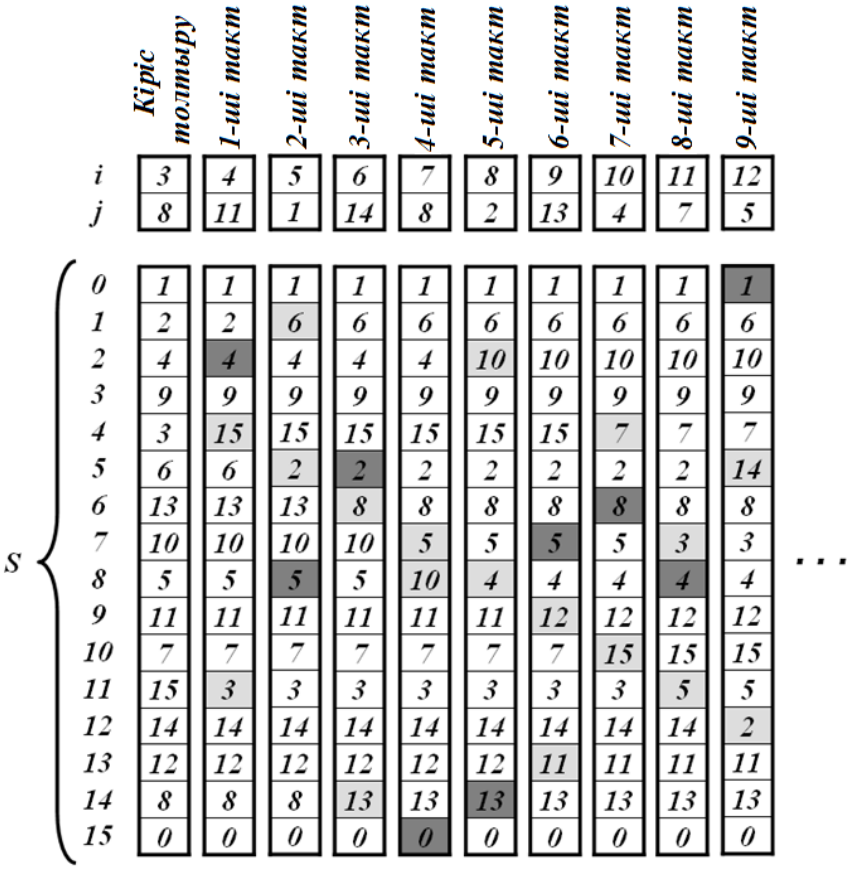
RC4 ПКС генераторы шығысында *4*-разрядты тізбек қалыптастырылады: *4, 5, 2, 0, 13, 5, 8, 4, 1*, … .



4.11 сурет – Күйлер матрицаның алмастырудың тактар тізбегі

Қарастырылған мысалда сөздің немесе блоктың көлемі n төртке тең болып алынды. Бұл мәнді басқа қылып алуға болады, мысалы, *8* немесе *16*. Егер $n = 8$ қолданса, онда S ауыстыру кестесі $2^8 = 256$ мәндерден тұру керек, ал ауыстыру кесте элементтері *0* ден *255* дейін сандар болу керек. Санағыштардың i және j көлемдерінде сегіз битке дейін өзгерту қажет (максималды мән – *255*).

Сонымен қатар, $n = 8$ жағдайда барлық есептеулер *256* модулі бойынша орындау қажет. Осындай өзгерістерді алгоритмде n параметрінің басқа да мәндері үшін орындау қажет.



4.12 сурет - *RC4 4*-разрядты *ПКС* генераторы жұмысының мысалы

4.5.3. RC4 ағынды шифрдың криптографиялық беріктілігі

RC4 алгоритмі криптографиялық талдаушылармен өте терең зерттелді. Оның ішінде ешқандай әлсіз орындар табылмады. Криптографиялық талдауға жоғары беріктіліктен басқа бұл алгоритм өте жылдам және ағынды шифрлау кезінде кілттік тізбекті генерациялау үшін қолдану мүмкін.

Ағынды шифрларды криптографиялық талдаудың барлық әдістерін үш класқа бөледі:

1. *Күштік* (*“қатқыл күш” шабуылы*). Толық талдау жолымен шабуылдар (барлық мүмкін нұсқаларды талдау). Толық талдау қиындығы есептің барлық мүмкін шешімдер санынан тәуелді (кілт кеңістігінің немесе ашық деректер кеңістігінің көлемінен). Бұл шабуыл түрін ағынды шифрлау жүйелерінің барлық түрлеріне қолдануға болады. Шифрлау жүйелерін құру кезінде құрушылар басқа бұзу әдістерімен салыстырғанда бұл шабуыл түрі ең тиімді емес болғандығына алып келуге тырысады.

2. *Статистикалық*. Екі бағынқы класқа бөлінеді:

- шифрлаушы гамманың статистикалық сипаттамаларын криптографиялық талдау әдісі: криптографиялық жүйенің алғашқы тізбегін зерттеуге бағытталған (криптографиялық талдаушы тізбектің келесі битын әртүрлі статистикалық тестер көмегімен кездейсоқ таңдаудың ықтималдығынан жоғары ықтималдықпен анықтауға талпынады);
- тізбектің қиындығын криптографиялық талдау әдісі: криптографиялық талдаушы гаммаға ұқсас, бірақ жүзеге оңайрақ асырылатын тәсілмен тізбекті генерациялау тәсілін табуға талпынады.

Екі әдісте сызықты қиындық принципын қолданады.

3. *Аналитикалық әдістер*. Бұл шабуыл түрі криптографиялық талдаушыға генератордың сипаттамасы, ашық және сәйкес жабық деректер белгілі деген болжаммен қарастырылады. Криптографиялық талдаушының мақсаты – қолданатын кілтті анықтау (регистрлердің алғашқы мәндерін).

Бақылау сұрақтары және тапсырмалар

1. Ағынды шифрдың блоктыдан қандай айырмашылығы бар?
2. Айнымалы ұзындығы бар деректер ағынын шифрлау қалай ұйымдастырылады?
3. Қандай сандар псевдокездейсоқ деп аталады?
4. Белгілі *ПКСГ* атаңыз. Криптографиялық мақсатпен қолдану үшін *ПКСГ* қандай қасиеттері болу керек?
5. Қарастырылған *ПКСГ* әр қайсысы үшін негізгі сипаттамаларын, кемшіліктерін және артықшылықтарын атаңыз.
6. Псевдокездейсоқ сандарды алу үшін кері байланысы бар жылжымалы регистрлер қалай қолдану мүмкін? Олардың жұмыс істеу принциптерін түсіндіріңіз.

7. Кездейсоқ және псевдокездейсоқ сандар генераторлары арасында қандай айырмашылықтар бар?

8. Ағынды шифрлау кезінде гамманы алу үшін шын кездейсоқ сандар генераторын қолдануға бола ма?

9. Қандай криптографиялық мақсатпен шын кездейсоқ сандар генераторларын қолдануға болады?

10. Әртүрлі параметрлер үшін *a*, *b* және *c* ($k_0 - 0$ тең деп қабылдау) сызықты конгруэнтты *ПКСГ* бірінші он сандардан тұратын тізбегін және периодын анықтаңыз:

а) $a = 5, b = 7, c = 17;$

б) $a = 6, b = 3$ және $c = 23$.

11. Келесі алғашқы деректер үшін k_a бастап кешігуі бар *Фибоначчи* әдісімен генерацияланатын он сандар тізбегін анықтаңыз:

а) $a = 3, b = 1, k_0 = 0,6; k_1 = 0,3; k_2 = 0,5$;

б) $a = 4, b = 2, k_0 = 0,9; k_1 = 0,3; k_2 = 0,5; k_3 = 0,9$.

12. Сызықты конгруэнтты генератор көмегімен алынған k_0, k_1, k_2, k_3 мәндері $k_0 = 1, k_1 = 8, k_2 = 10, k_3 = 9$ тең. *ПКСГ* *a*, *b* және *c* параметрлерін анықтаңыз.

13. *ПКС BBS* әдісі бойынша x_{11} есептеу, егер: а) $p = 19, q = 23, x = 3$; б) $p = 23, q = 31, x = 3$.

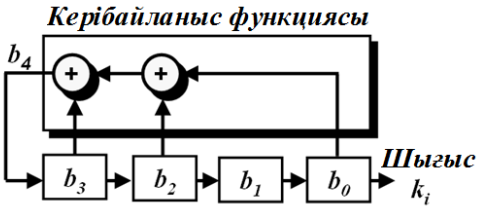
14. *ПКС BBS* әдісі бойынша *12* бит ұзындығы бар псевдокездейсоқ екілік тізбекті есептеу, егер: а) $p = 19, q = 23, x = 3$; б) $p = 23, q = 31, x = 3$. Кездейсоқ биттер ретінде санның екілік жазбасында x_0 бастап кіші битті алу.

15. *ПКС BBS* генерациялау әдісі бойынша $x_1 - x_8$ мәндерін есептеу, егер: а) $p = 19, q = 23, x = 3$; б) $p = 23, q = 31, x = 3$.

16. Ағынды алгоритм *RC4* үшін егер сеанс кілті *1, 2, 3, 4, 5, 6* және *7* мәндері бар *7* байт болса, кілттік ағынның бірінші *20* элементін көрсету.

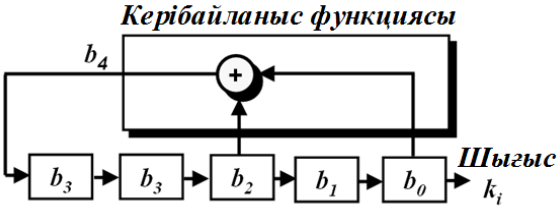
17. Сипаттамалық полиномы $x^5 + x^2 + + 1$ үшін *LFSR* көрсетіңіз. Берілген *LFSR* қандай максималды тізбек периоды қалыптастырылады?

18. Берілген *LFSR* сипаттамалық полиномын анықтаңыз:



Берілген *LFSR* қандай максималды тізбек периоды қалыптастырылады?

19. Егер тізбектің алғашқы мәні $(1110)_2$ болса, онда берілген *LFSR* генерациялайтын *20* битке кілттік ағынды анықтаңыз:



20. *LFSR* максималды период ұзындығы – *32* разряд. Жылжымалы регистр *LFSR* қанша рязряд бар?

21. *A5/1* ағынды алгоритм үшін әр сызықты жылжу регистры үшін екілік тізбектің максималды периодын табыңыз.

22. *A5/1* ағынды алгоритм үшін функция мәндерін табыңыз: а) $f(x, y, z) = f(1, 0, 0)$; б) $f(x, y, z) = f(0, 1, 1)$; в) $f(x, y, z) = f(0, 0, 0)$; г) $f(x, y, z) = f(1, 1, 1)$. Әр жағдай үшін қанша жылжу регистрлері синхрондалатынын көрсетіңіз.

5 тарау

ДЕРЕКТЕРДІ ШИФРЛАУДЫҢ БЛОКТЫ СИММЕТРИЯЛЫҚ СТАНДАРТЫ DATA ENCRIPTION STANDARD

5.1. DES ДЕРЕКТЕРДІ ШИФРЛАУ АЛГОРИТМІН ҚҰРУ ТАРИХЫ

Деректерді шифрлау стандарты (*Data Encryption Standard – DES*) – симметриялық кілттері бар блокты шифр, Ұлттық Стандарттар және Технологиялар Институтымен құрылған (*National Institute of Standards and Technology – NIST*) [4, 29, 43].

1973 жылы *NIST* симметриялық кілттері бар ұлттық криптографиялық жүйе ұсынысын құру үшін сұраныс жариялады.

IBM ұсынылған "*Люцифер*" (*Lucifer*) деп аталатын жобаның түрі *DES* ретінде қабылданды. *DES* эскиздік түрде Ақпаратты Өңдеудің Федералды Стандарты (*Federal Information Processing Standard – FIPS*) ретінде Федералды Регистрде 1975 жылы жарияланды [4, 29, 43].

Жараялаудан кейін эскиз қатаң сынға ұшырады екі себебпен: *бірінші* – кілттің қысқа ұзындығы (тек 56 бит), ол *«қатқыл күш»* шабуылына шифрды әлсіздендіру мүмкін; *екінші* себеп – критика берушілер *DES* ішкі құрылымының кейбір жасырылған құрылуына сенімсіздік білдірді. Олар құрылымның кейбір бөлігінде (ауыстырудың *S*-блоктары) жасырын «жол» болу мүмкін, ол хабарды кілтсіз керішифрлауға мүмкіндік береді деп ойлады. Кейбір уақыттан кейін *IBM* жобалаушылары ішкі құрылым криптографиялық талдауды болдырмау үшін қайта жасалды деп хабарлады.

1977 жылының ақпанында *DES* Федералдық Регистрде *FIPS 46* ретінде баспаға шықты. Бірақ *FIPS DES*–ті ресми емес қосымшаларда қолданатын стандарт деп хабарлады.

Стандарт негізіне қойылған алгоритм жеткілікті тез таралды және 1980 жылы *АҚШ NIST* қабылданды. Сол сәттен *DES* стандарт тек атымен (*Data Encryption Standard*) ғана емес, нақты түрде болды. Ақпаратты деректерді тарату желілерінде шифрлау және керішифрлауға арналған бағдарламалық қамтама және арнайы *шағын-ЭЕМ* пайда болды.

DES – Ақпаратты Өңдеудің Федералды Стандартының аты (*FIPS 46-3*), ол деректерді шифрлау алгоритмін сипаттайды (*Data Encryption Algorithm – DEA*). *ANSI* терминдерінде *DEA* стандарт *X9.32* деп анықталған.

NIST құжаттарында *DES* криптографиялық жүйесі деректерді шифрлау алгоритмі (*DEA*) деп аталады, ал стандарттау бойынша халықаралық ұйым *DES* шифрына сілтеме жасап, *DEA-1* аббревиатурасын қолданады [4, 29, 43].

Бұл алгоритм жиырма жылдан астам уақытта әлемдік стандарт болды және барлық тілеушілерге қолжетімді ресми стандарт ретінде бекітілді. Сондықтан оны әскері қолданудан кеңмасштабты қолдануға дейінгі криптография жолында маңызды қадам ретінде белгілеуге болады.

DES оны жариялаудан бастап ең кең қолданыста болған симметриялық кілттері бар блокты алгоритм болды. Уақыт өткеннең кейін *NIST* жаңа стандартты ұсынды – *FIPS 46-3*, ол болашақ қосымшалар үшін үшеселі *DES* (үш рет қайталанған *DES* шифры) қолдануды ұсынды.

DES алгоритмін құрастырушыларына қойылған негізгі талаптар келесі:

- алгоритм қауіпсіздіктің жоғары деңгейін қамтамасыз ету керек;
- алгоритм толық анықталған және түсінгенге жеңіл болу керек;
- алгоритмнің қауіпсіздігі кілтке негізделену керек және алгоритмнің өзін құпия сақтауға тәуелді болмау керек;
- алгоритм барлық пайдаланушыларға қолжетімді болу керек;
- алгоритмді әртүрлі қолдануларға адаптация жасау мүмкіндігі болу керек;
- алгоритмді электронды аспап түрінде тиімді жүзеге асыру мүмкіндігі болу керек;

- алгоритм қолдануда тиімді болу керек;
- алгоритм тексеру мүмкіндіктерін беру керек;
- алгоритм экспорт үшін рұқсатталу керек.

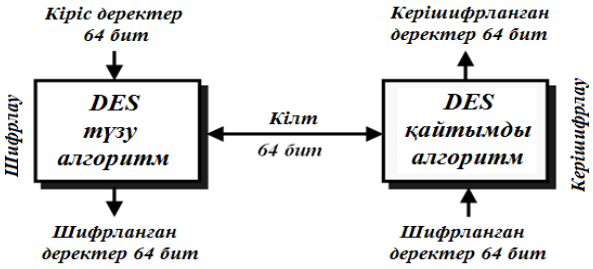
DES алгоритмнің негізгі артықшылықтары:

- ұзындығы *64* (*56* бит кілт ұзындығы және *8* бақылау разрядтары) бит тек бір кілт қолданылады;
- хабарды бір бағдарлама пакетімен шифрлап, керішифрлау үшін *DES* стандартына сәйкес келетін басқа кез келген бағдарлама пакеттерін қолдануға болады;
- алгоритмнің қарапайымдылығы өңдеудің жоғары жылдамдығын қамтамасыз етеді;
- алгоритмнің жеткілікті жоғары беріктілігі.

Алдымен *DES* стандарты негізінде жатқан әдіс, *IBM* фирмасымен өз мақсаттары үшін құрылған және "*Люцифер*" жүйесі түрінде жүзеге асырылған. "*Люцифер*" жүйесі ауыстыру және алмастыру әдістерін құрамдастыруда негізделген және ол ауыстыру және алмастыру блоктарының кезектескен тізбектен тұрады. Оның ішінде ауыстыру және алмастыру блоктарын басқаратын *128* биттік кілт қолданды. "*Люцифер*" жүйесі шифрлаудың кішкентай жылдамдығы (*2190 байт/с* – бағдарламалық жүзеге асыру және *96970 байт/с* – аппараттық жүзеге асыру) үшін тәжірибелік жүзеге асыру үшін өте қыйын болды.

5.2. DES ДЕРЕКТЕРДІ ШИФРЛАУ АЛГОРИТМІН ҚҰРУ ПРИНЦИПТЕРІ

5.1 суретінде көрсетілгендей, *DES* – деректерді шифрлау және керішифрлаудың блокты симметриялық криптографиялық жүйесі.



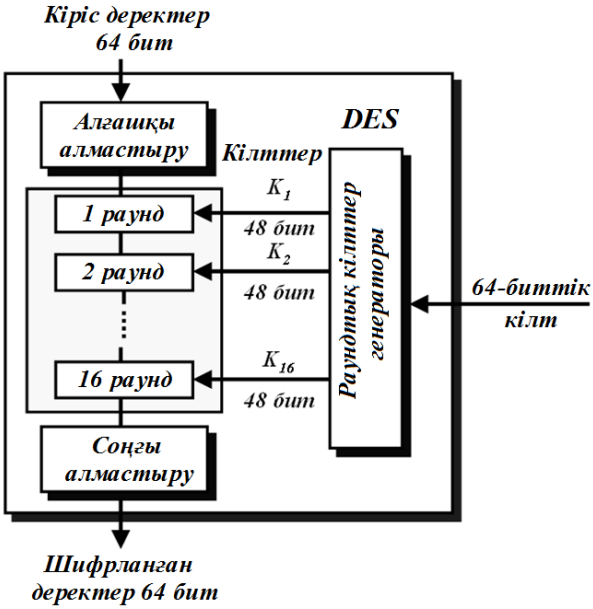
5.1 сурет - *DES* стандартын қолданумен деректерді шифрлау және керішифрлау

Шифрлау жағында *DES* *64*-биттік алғашқы деректер блогын қабылдайды және *64*-биттік шифрланған деректер блогын жасайды; Керішифрлау жағында *DES* *64*-биттік шифрланған деректер блогын қабылдайды және *64*-биттік алғашқы деректер блогын шығарады. Екі жақтада шифрлау және керішифрлау үшін бір *64*-биттік кілт қолданылады.

DES алгоритмі алмастыру және ауыстыру комбинацияларын қолданады. *DES* *64*-биттік деректер блоктарын *64*-биттік кілт көмегімен шифрлайды. Кілтте мәнді болып *56* бит есептеледі (әр сегізінші бит кілттің қалған биттерінің жұптығын бақылау үшін қолданылады) [12]. *DES* керішифрлау операциясы шифрлау операциясының кері операциясы болады, және шифрлау операцияларын қайта ретімен қайталау жолымен орындалады.

5.3. DES ДЕРЕКТЕРДІ ШИФРЛАУ АЛГОРИТМІНІҢ ҚҰРЫЛЫМЫ

DES алгоритімінде шифрлау процессінің жалпылама сұлбасы 5.2 суретінде көрсетілген.



5.2 сурет - *DES* алгоритімінде шифрлау процессінің жалпылама сұлбасы

Шифрлау процессінде *64*-биттік блоктың алғашқы алмастыруы, шифрлаудың он алты раунды және соңғы алмастыруы орындалады.

Шифрлаудың он алты раундын қолдануға келесі себептер бар:

- криптографиялық қорғаудың керекті деңгейін қамтамасыз ету үшін минималды қажетті раундтар саны он екі;
- аппаратты жүзеге асыру кезінде он алты раундты қолдану келесі түрлендірулер үшін түрлендірілген кілтті алғашқы күйге қайтаруға мүмкіндік береді;
- берілген раундтар саны сонымен қатар шифрланған деректер блогына екі жақтан шабуыл жасауды аластату үшін қажет.

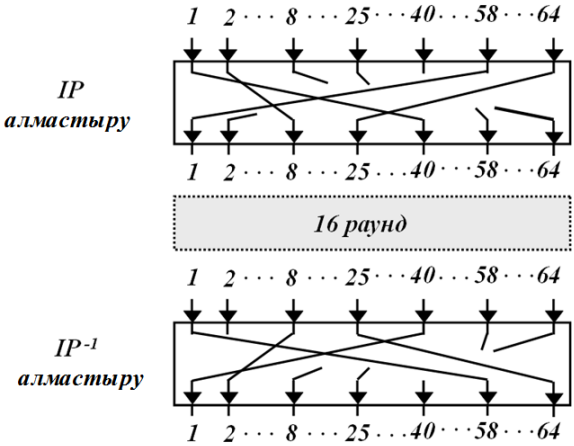
Әр раунд әртүрлі (он алты) генерацияланған *48*-биттік кілттерді қолданады.

DES кейбір жүзеге асыруларында ашық мәтін блоктары шифрлау құрылғысының регистрлеріне жүктеу алдында алғашқы алмастыру процедурасын өтеді. Бұл процедураны хабардың статистикалық құрылымын алғашқы шашыратуын өткізу үшін қолданылады [12, 29].

Алғашқы алмастыруды қолдану жағдайда (*Initial Permutation – IP*), он алты раундты орындағаннаң кейін алынған блокқа кері алмастыру қолданылады (IP^{-1}).

IP деректер биттерін алмастыру IP^{-1} инверсиясы бар сым коммутация деп аталады, яғни соңғы алмастыру алғашқысына кері болады. Қолданылатын екі алмастырудың *DES* криптографиясына ешқандай мәні жоқ. Екі алмастыру кілтсіз және алдыналы белгілі. Бұл процесстің екі жағына да бір бағдарламалық немесе аппараттық қамтаманы қолдануға мүмкіндік береді: шифрлау және керішифрлау үшін. 5.3 суреті алғашқы және соңғы алмастыруды көрсетеді.

Әр алмастыру *64*-биттік деректер блоктарын қабылдайды және оның элементтерін берілген ереже бойынша ауыстырады. 5.3 суретінде кіріс және сәйкес келетін шығыс порттарының кейбіреулері ғана көрсетілген. Бұл алмастырулар – кілтсіз түзу алмастырулар, олар бір біріне инверсты. Мысалы, алғашқы алмастыруда (*IP*) кірісте *58*-ші бит шығыста бірінші битке ауысады. Тұра солай, соңғы алмастыруда (IP^{-1}) бірінші кіріс биты шығыста *58*-ші битке ауысады. Басқа сөзбен, егер осы екі алмастыру арасында раунд болмаса, алғашқы алмастыру құрылғысының кірісіне келге *58*-ші бит соңғы алмастырудың *58*-ші шығысына жіберіледі.



5.3 сурет – *DES* алмастырудың алғашқы және соңғы қадамдары

64-биттік деректер блогы үшін алғашқы *IP* және соңғы IP^{-1} алмастыру ережелері 5.1 және 5.2 кестелерінде көрсетілген.

5.1 кесте

Алғашқы *IP* алмастыру ережелері

	Бит нөмірлері															
Кіріс	*58*	*50*	*42*	*34*	*26*	*18*	*10*	*02*	*60*	*52*	*44*	*36*	*28*	*20*	*12*	*04*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Кіріс	*62*	*54*	*46*	*38*	*30*	*22*	*14*	*06*	*64*	*56*	*48*	*40*	*32*	*24*	*16*	*08*
Шығыс	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
Кіріс	*57*	*49*	*41*	*33*	*25*	*17*	*09*	*01*	*59*	*51*	*43*	*35*	*27*	*19*	*11*	*03*
Шығыс	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*
Кіріс	*61*	*53*	*45*	*37*	*29*	*21*	*13*	*05*	*63*	*55*	*47*	*39*	*31*	*23*	*15*	*07*
Шығыс	*49*	*50*	*51*	*52*	*53*	*54*	*55*	*56*	*57*	*58*	*59*	*60*	*61*	*62*	*63*	*64*

Бұл ауыстыру (*IP* және IP^{-1}) шифрдың беріктілігіне әсер етпейді және пайдаланушылар оны орындау қажет па деп сұрақ қояды. *DES* құрған шығармашылық ұжымның мүшесінің бірі олар шифрлау процедурасын жүзеге асыруды жеңілдетеді деп айтты [27, 29, 41].

5.2 кесте

Кері алмастыру IP^{-1} ережелері

	Бит нөмірлері															
Кіріс	*40*	*08*	*48*	*16*	*56*	*24*	*64*	*32*	*39*	*07*	*47*	*15*	*55*	*23*	*63*	*31*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Кіріс	*38*	*06*	*46*	*14*	*54*	*22*	*62*	*30*	*37*	*05*	*45*	*13*	*53*	*21*	*61*	*29*
Шығыс	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
Кіріс	*36*	*04*	*44*	*12*	*52*	*20*	*60*	*28*	*35*	*03*	*43*	*11*	*51*	*19*	*59*	*27*
Шығыс	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*
Кіріс	*34*	*02*	*42*	*10*	*50*	*18*	*58*	*26*	*33*	*01*	*41*	*09*	*49*	*17*	*57*	*25*
Шығыс	*49*	*50*	*51*	*52*	*53*	*54*	*55*	*56*	*57*	*58*	*59*	*60*	*61*	*62*	*63*	*64*

Кестелердегі барлық алмастырулар және кодтар кілтті іріктеу жолымен керішифрлау процессін максималды қыйындату үшін құрушылармен арнайы таңдалған.

Барлық көрсетілген кестелер стандартты болып табылады және *DES* алгоритмін жүзеге асыруға өзгертулерсіз қосылу керек екенін айта кету қажет.

5.1 мысал. Алмастырудың алғашқы блок шығысындағы нәтижесін анықтау қажет егер оның кірісіне келесідей он алтылық тізбек кірсе:

$$0002\ 0000\ 0000\ 0001_{16}.$$

Шешім. Кірісте тек екі бір болады (*15* бит және *64* бит); түзу алмастыру қолданғандықтан шығыстада тек екі бір болу керек. 5.1 кестені қолданып, осы екі битпен байланысты шығысты табуға болады. Кірісте *15* бит шығыста *63* бит болады. Кірісте *64* бит шығыста *25* бит болады. Шығыста тек екі бір болады - *25* бит және *63* бит.

Сондықтан, он алтылық санақ жүйесінде нәтиже келесі болады:

$$0000\ 0080\ 0000\ 0002_{16}.$$

5.2 мысал. 5.1 мысалында алынған алғашқы және соңғы алмастырулар бір біріне инверсты екенін дәлелдеу.

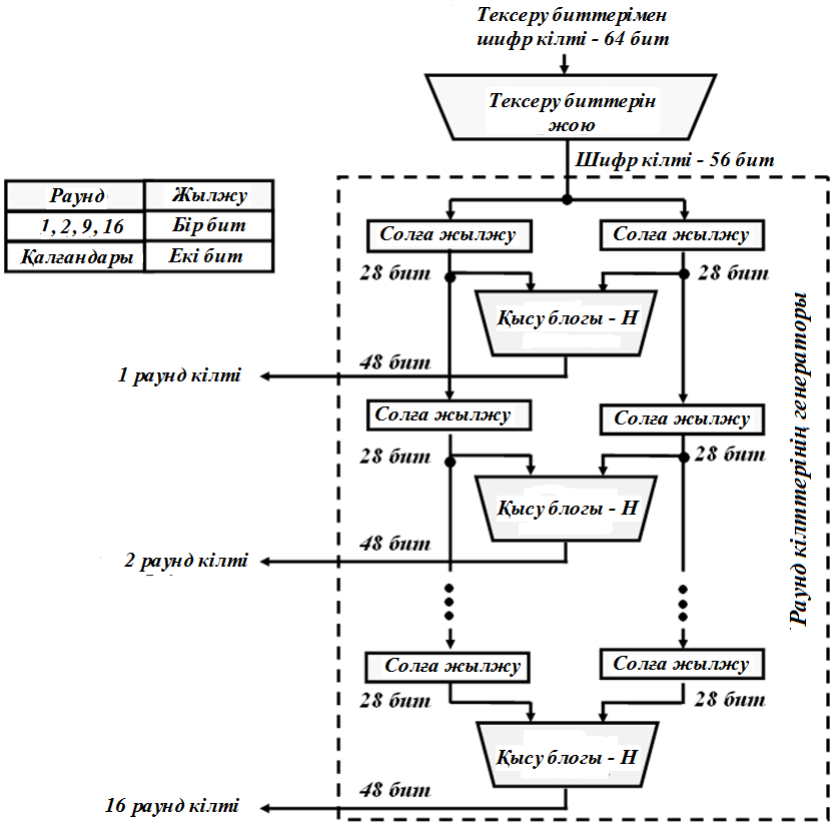
Шешім. Алынған шығыс тізбекті $0000\ 0080\ 0000\ 0002_{16}$ кіріске түрлендіреміз. Бірлік биттер – *25* және *63*, басқа биттер нөлге тең. Соңғы алмастыруда *25*-ші бит *64*-шіге ауысады, ал *63*-ші – *15*-шіге. Нәтиже:

0002000000000001_{16}.

Нәтижеден көрінетіндей алғашқы және соңғы алмастырулар – бұл алмастырудың түзу блоктары, олар бір біріне инверсты.

5.4. DES ДЕРЕКТЕРДІ ШИФРЛАУ ҮШІН РАУНДТЫҚ КІЛТТЕРДІ ГЕНЕРАЦИЯЛАУ

Кілт генераторы *56* бит ұзындығы бар шифр кілтінен он алты *48* биттік кілттерді құрады. Бірақ шифр кілті әдетте *64* биттік кілт ретінде беріледі, оның ішінде *8* қосымша биттер тексеру биттері. Олар раундық кілттерді генерациялау процессі (5.4 сурет) алдында алынып тасталады.



5.4 сурет – Раундық кілттерді генерациялаудың құрылымдық сұлбасы

5.4.1. Шифрдың кілтті тексеру биттерін өшіру

Кілттерді кеңейту алдындағы процесс – қысуды алмастыру, оны шифр кілтінің тексеру биттерін жою деп атайды. Ол *64*-биттік кілттен жұптық биттерін (*08, 16, 24, 32, 40, 48, 56, 64* биттері) жояды. Бұл биттер шифрлауға әсер етпейді. Кілттің көрсетілген биттері кілттің әр байтында бірлер тақ болғанына жауап береді.

Сонымен, шифр кілті нақты *56*- бит. Бақылау биттерін жою үшін және кілтті жұмысқа дайындау үшін *G* функциясы қолданылады. *G* функциясы – кілтті алғашқы дайындау функциясы (5.3 кесте). *G* функциясын орындау нәтижесі алмастыру болады, ол әдебиетте *PC-1* деп аталады [12, 29].

5.3кесте

Кілтті алғашқы дайындаудың *G* функциясы

	Бит нөмірлері													
К кіріс	*57*	*49*	*41*	*33*	*25*	*17*	*09*	*01*	*58*	*50*	*42*	*34*	*26*	*18*
Шығыс	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*
К кіріс	*10*	*02*	*59*	*51*	*43*	*35*	*27*	*19*	*11*	*03*	*60*	*52*	*44*	*36*
Шығыс	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*
К кіріс	*63*	*55*	*47*	*39*	*31*	*23*	*15*	*07*	*62*	*54*	*46*	*38*	*30*	*22*
Шығыс	*29*	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*
К кіріс	*14*	*06*	*61*	*53*	*45*	*37*	*29*	*21*	*13*	*05*	*28*	*20*	*12*	*04*
Шығыс	*43*	*44*	*45*	*46*	*47*	*48*	*49*	*50*	*51*	*52*	*53*	*54*	*55*	*56*

5.3. кесте екі бөлімге бөлінеді. Түрлендіру нәтижесі *G(K)* әр қайсысы *28* биттен тұратын екі бөлімге C_0 және D_0 бөлінеді. 5.3 кестенің бірінші екі жолы C_0 тізбегінің биттері қалай таңдалатынын анықтайды (C_0 бірінші биты – шифр кілтінің *57*-ші биты, содан кейін *49*-ші бит және ары қарай, ал сонғы биттер – кілттің *44*-ші және *36*-ші биттері). 5.3 кестенің келесі екі жолы D_0 тізбегінің биттері қалай таңдалатынын анықтайды (яғни D_0 тізбегі шифр кілтінің *63, 55, 47*, … , *12, 04* биттерінен тұрады).

5.3 мысал. Кілтті дайындау функцияның *G* шығысында болатын нәтижені анықтау қажет, егер шифр кілті

$$K = abab19283746cdcd_{16}.$$

Шешім. Шифр кілтінің мәндерін екілікке ауыстырып

$$1010\ 1011\ 1010\ 1011\ 0001\ 1001\ 0010\ 1000_2$$
$$0011\ 0111\ 0100\ 0110\ 1100\ 1101\ 1100\ 1101_2$$

және кілтті дайындау функциясын G қолданып (5.3 кесте) келесі тізбектерді аламыз:

$$C_0 = 1100\ 0011\ 1100\ 0000\ 0011\ 0011\ 1010_2 = c3c033a_{16};$$

$$D_0 = 0011\ 0011\ 1111\ 0000\ 1100\ 1111\ 1010_2 = 33f0cfa_{16}.$$

5.4.2. C_i және D_i тізбектерін сол жақа циклды жылжыту

Түзу алмастырудан кейін кілт *28* биттен екі бөлікке бөлінген. C_0 және D_0 тізбектерді қалыптастырудан кейін *16* раундтың әр қайсысына ($i = 1,2,...,16$) C_i және D_i тізбектері рекурсивті анықталады. Ол үшін раундтың нөміріне тәуелді, 5.4 кетеде көрсетілгендей, бір немесе екі биттке солға циклдық жылжыту операциялары қолданылады.

5.4 кесте

C_i и D_i тізбектерді солға циклдық жылжытудың биттер саны

Раундтар нөмірлері															
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
1	*1*	*2*	*2*	*2*	*2*	*2*	*2*	*1*	*2*	*2*	*2*	*2*	*2*	*2*	*1*
Солға жылжытудың биттер саны															

Солға циклды жылжыту операциялары C_i және D_i тізбектері үшін тәуелсіз орындалады. Мысалы, C_3 тізбегін алу үшін C_2 тізбегін екі позицияға солға циклды жылжытады, ал D_3 тізбегін алу үшін D_2 тізбегін екі позицияға солға циклды жылжытады. C_{16} және D_{16} тізбектерін алу үшін сәйкес C_{15} және D_{15} тізбектерін бір позицияға солға циклды жылжытады.

Раундтық кілттерді генерациялаудың сұлбасында циклды жылжытудың жалпы саны (C_i және D_i тізбектерінің жалпы циклдық жылжытуы *28* бит) арнайы келесі үшін таңдалған: соңғы раундтық кілтті қалыптастырғаннаң кейін C_{16} және D_{16} тізбектерінің мәндері C_0 и D_0 тізбектерінің мәндеріне тең болу үшін.

5.4 мысал. C_0 және D_0 тізбектерін солға циклды жылжыту операциядан кейін нәтижені екінші раундтық кілтті k_2

қалыптастыру үшін анықтау қажет, егер C_0 және D_0 мәндері келесіге тең:

$$C_0 = 1100\ 0011\ 1100\ 0000\ 0011\ 0011\ 1010_2 = c3c033a_{16};$$

$$D_0 = 0011\ 0011\ 1111\ 0000\ 1100\ 1111\ 1010_2 = 33f0cfa_{16}.$$

Шешім. Екінші раундтық кілтті k_2 қалыптастыру үшін 5.4 кестеге сәйкес C_0 және D_0 тізбектері екі разрядқа сол жаққа циклды жылжу қажет. C_0 және D_0 тізбектерін екі разрядқа сол жаққа циклды жылжыту нәтижесінде аламыз:

$$C_2 = 0000\ 1111\ 0000\ 0000\ 1100\ 1110\ 1011_2 = 0f00ceb_{16};$$

$$D_2 = 1100\ 1111\ 1100\ 0011\ 0011\ 1110\ 1000_2 = cfc33e8_{16}.$$

5.4.3. C_i және D_i тізбектерін біріктіру, оларды алмастыру және қысу

C_i және D_i тізбектерін сол жаққа циклды жылжытудан кейін олар *56* биттік блокты құру үшін біріктіріледі, яғни $C_i||D_i$ тізбегі болады, бұл жерде || – математикалық конкатенация (біріктіру) операциясының белгісі.

Алмастыру және қысу (*H*-функция) $C_i||D_i$ *56* битын k_i *48* битына өзгертеді, олар раундтық кілттер ретінде қолданылады. $C_i||D_i$ тізбекті алмастыру және қысу 5.5 кестесінде көрсетілген

5.5 кесте

Раундтық кілттерді өңдеудің соңғы функциясы *H*

Кіріс	*14*	*17*	*11*	*24*	*01*	*05*	*03*	*28*	*15*	*06*	*21*	*10*	*23*	*19*	*12*	*04*
Шығыс k_i	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Кіріс	*26*	*08*	*16*	*07*	*27*	*20*	*13*	*02*	*41*	*52*	*31*	*37*	*47*	*55*	*30*	*40*
Шығыс k_i	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
Кіріс	*51*	*45*	*33*	*48*	*44*	*49*	*39*	*56*	*34*	*53*	*46*	*42*	*50*	*36*	*29*	*32*
Шығыс k_i	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

Итерацияның әр қадамында анықталатын раундтық кілттер k_i деген $C_i||D_i$ *56*-биттік тізбегінен нақты биттерді таңдау, алмастыру және қысу нәтижесі. Басқа сөзбен, раундтық кілт

$$k_i = H(C_i||D_i), \qquad (5.1)$$

бұл жерде *H – 48* биттік ұзындығы бар кілтті өңдеуді бітіретін функция, 5.5 кестемен анықталады. *H* функциясын орындау нәтижесі әдебиетте *PC-2* деп аталатын алмастыру болады [12, 29].

5.5 кестесінен кілттің k_i бірінші биты $C_i||D_i$ тізбегінің *14*-ші биті болады, екінші - *17*-ші бит, кілттің k_i *47*-ші биті $C_i||D_i$ тізбегінің *29*-ші биті, ал *48*-ші бит - $C_i||D_i$ *32*-ші биты.

H функциясының кірісінде *56* бит ұзындығы бар $C_i||D_i$ тізбегі болды, ал оны орындау нәтижесінде k_i раундтық кілт ұзындығы *48* бит болды. Қысу *H* функциясы көмегімен алмастыруда *9, 18, 22, 25, 35, 38, 43* и *54* (5.5 кестені қараңыз) нөмірлері бар сегіз бит қатыспау себебімен болды.

5.5 мысал. Ары қарай екінші раундық кілт k_2 қалыптастыру үшін C_2 және D_2 тізбектерді біріктіру операциясының нәтижесін анықтау қажет, егер C_2 және D_2 мәндері тең

$$C_2 = 0f00ceb_{16}; \quad D_2 = cfc33e8_{16}.$$

Шешім. C_2 және D_2 тізбектерін біріктіру келесі болады:

$$C_2||D_2 = 0f00ceb16||cfc33e8_{16} = 0f00cebcfc33e8_{16}.$$

5.6 мысал. Екінші раундтық кілтті k_2 генерациялау қажет, яғни раундтық кілттерді соңғы өңдеу функциясының *H* нәтижесін анықтау қажет. Егер кірісте C_2 және D_2 тізбектерінің біріктірген түрі келесі болса: $C_2||D_2 = 0f00cebcfc33e8_{16}$.

Шешім. $C_2||D_2$ екілік мәндерін қолданып

$$C_2||D_2 = 0f00cebcfc33e8_{16} =$$

$$= 0000\ 1111\ 0000\ 0000\ 1100\ 1110\ 1011$$

$$1100\ 1111\ 1100\ 0011\ 0011\ 1110\ 1000_2$$

және 5.5 кестесі көмегімен ауыстыруды орындап, екінші раундтық кілтті k_2 аламыз

$$k_2 = 0100\ 0101\ 0110\ 1000\ 0101\ 1000\ 0001\ 1010$$

$$1011\ 1100\ 1100\ 1110_2 = 4568581abcce_{16}.$$

5.4.4. Раундтық кілттерді генерациялау алгоритмі

Енді тексеру биттері бар шифр кілтінен раундтық кілттерді құру үшін, алгоритмнің бірнеше процедураларын қолданатын, қарапайым алгоритмді көрсетеміз:

```
Key_Generator (key[64], RoundKeys[16, 48]. SinfTable[16])
{
    permute (64, 56, key, cipherKey, ParityDropTable)
    spilt (56, 28, cipherKey, leftKey, rightKey)
    for (round = 1 to 16)
    {
            shiftLeft (leftKey, ShiftTable[round])
            shiftLeft (rightKey, ShiftTable[round])
            combine (28, 56, leftKey, rightKey, preRoundKey)
            permute (56, 48, preRoundKey, RoundKeys[round],
            KevCompressionTable)
    }
}
shiftLeft (block[28], NumOfShifts)
{
    for (i = 1 to numShifts)
    {
            T ← block[1]
            for (j = 2 to 28) { block [j-1] ← block [j] }
            block[28] ← T
    }
}
```

Бұл жерде *ShiftLeft* сол жақа жылжыту процедурасы. *T* – уақытша блок екеніне назар аударыңыз.

5.5. DES ДЕРЕКТЕРДІ ШИФРЛАУ

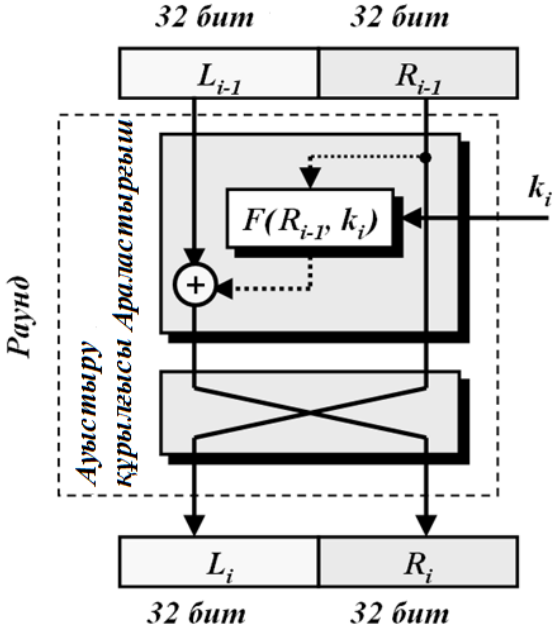
Алдымен *DES* деректерді шифрлауды, содан кейін керішифрлауды қарастырамыз.

Деректердің кіріс блогына *x* алғашқы алмастыруды *IP* орындаудан кейін, алынған *64*-биттік тізбек $x_0 = = IP(x)$ келесі түрде болады:

$$x_0 = L_0 \,||\, R_0,$$

бұл жерде L_0 – сол жақ (*left*) және R_0 – оң жақ (*right*) биттер тізбектері, олардың ұзындықтары бірдей *32* биттен.

DES шифрлау процессі он алты раундтан турады, сондықтан деректер блогы $x_0 = L_0 || R_0$ он алты рет Фейстель сұлбасы деп аталатын сұлба бойынша түрлендіріледі. *DES* әр раунды Фейстель сұлбасын 5.5 суретінде көрсетілгендей қолданады.



5.5 сурет - *DES* шифрлаудың бір раунды

Деректерді шифрлау процессін он алты раундты қолданып математикалық түрде көрсетуге болады:

$$L_i = R_i;$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i), i = 1,2,.,16, \quad (5.2)$$

бұл жерде $F(\bullet)$ – шифрлау функциясы; k_i – раундтық кілттер; $\oplus$ – деректерді биттермен *2* (*xor*) модулі бойынша қосу операциясы.

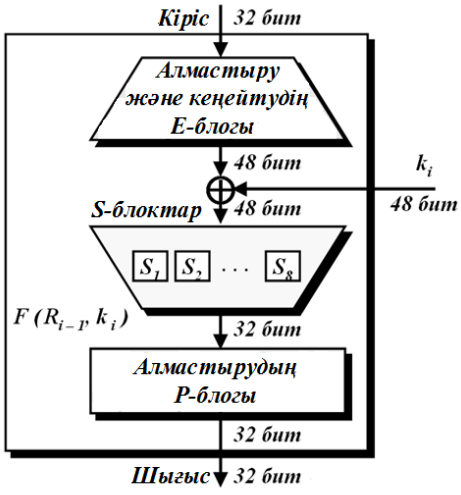
Шифрлаудың он алты раундын орындағаннан кейін $x_{16} = L_{16} \,||\, R_{16}$ деректер блогын IP^{-1} кері алмастырады. Нәтижесінде *64*-биттік шифрланған деректер блогы y шығады

$$y = IP^{-1}(x_{16}) = IP^{-1}(L_{16}||R_{16}). \qquad (5.3)$$

Әр ағымдағы раунд L_{i-1} және R_{i-1} алдынғы раундтан (немесе алмастырудың алғашқы блогынан) алады және келесі раунд үшін L_i және R_i құрады, олар келесі раундқа барады (немесе соңғы алмастыру блогына). Жоғарыда көрсетілгендей, әр раундта екі шифр элементі бар екенін қабылдауға болады: араластырғыш және ауыстыру құрылғысы. Осы элементтің әр қайсысы қайтымды. Ауыстыру құрылғысы – қайтымды, ол деректердің сол жақтағы жартысын оң жақтағы жартысымен орнын ауыстырады. *Xor* операциясы қайтымды болғандықтан араластырғышта қайтымды. Барлық қайтымды емес элементтер *Фейстель функциясында* $F(R_{i-1}, k_i)$ орналасқан.

5.5.1. DES араластырғышының F функциясы

DES Фейстель сұлбасының негізі *F* функциясы. *F* функциясы *48*-биттік кілт көмегімен *32* ең оң жақтағы биттерді (R_{i-1}) *32*-биттік сөзді алу үшін шифрлайды. Бұл функцияда, 5.6 суретте көрсетілгендей, төрт операция бар: алмастырудың және кеңейтудің *E*-блогы; *2* ($\oplus$) модулю бойынша разряд бойынша қосу; ауыстырудың және қысудың *S*-блоктар тобы; түзу ауыстырудың *P*-блогы.



5.6 сурет - *DES F* функциясы

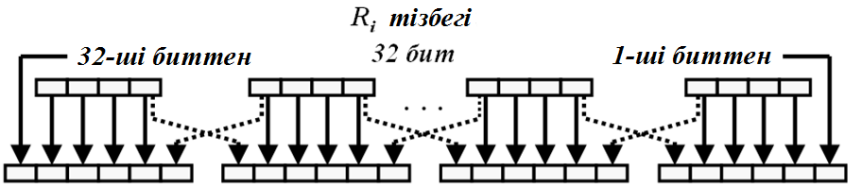
F функцияның алмастыру және кеңейту операциясы

R_{i-1} кірісінде 32 бит, ал кілт k_i ұзындығы *48* бит болғандықтан, алдымен R_{i-1} *48* битке дейін кеңейту қажет. Ол үшін алдымен R_{i-1} *4* биттен сегіз секцияға бөлінеді. Содан кейін *4* биттік әр секция *6* битке дейін кеңейтіледі. Бұл кеңейтуі бар алмастыру алдын ала берілген ереже бойынша орындалады. Секциялар үшін кіріс биттер *1, 2, 3* және *4* шығыста сәйкес *2, 3, 4* және *5* биттеріне меншіктеледі. Шығыс бит *1* алдынғы секциядан кіріс бит *4* негізінде қалыптастырылады; шығыс бит *6* келесі секцияда *1* битінен қалыптастырылады. Егер *1* және *8* секцияларды көрші секция деп қарастырса, онда сол ережелер *1* және *32* биттеріне қолданылады.

Сонымен, алмастыру және кеңейтудің *E*-блогынаң кейін R_{i-1}^{E} тізбегін аламыз

$$R_{i-1}^{E} = E(R_{i-1}). \qquad (5.4)$$

5.7 сурет алмастыру және кеңейтудің *E*-блогының кіріс және шығыстарын көрсетеді.



5.7 сурет – Алмастыру және кеңейту *E*-блок процесстерін түсіндіру

Кіріс және шығыс арасындағы қатынастар математикалық түрде анықталу мүмкін болса да, алмастыру және кеңейтудің *E*-блогын 5.6 кестесін қолданып анықтау оңай.

Шығыс саны *48,* ал кіріс саны *32* екеніне назар аударыңыз. Кейбір кірістер бірден көп шығысқа барады. Мысалы, кіріс бит мәні *5* шығыстың *6* және *8* биттерінің мәні болады.

5.6 кесте

Алмастыру және кеңейту *E*-блок ережелері

Кіріс	*32*	*01*	*02*	*03*	*04*	*05*	*04*	*05*	*06*	*07*	*08*	*09*	*08*	*09*	*10*	*11*
Шығыс R_{i-1}^E	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Кіріс	*12*	*13*	*12*	*13*	*14*	*15*	*16*	*17*	*16*	*17*	*18*	*19*	*20*	*21*	*20*	*21*
Шығыс R_{i-1}^E	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
Кіріс	*22*	*23*	*24*	*25*	*24*	*25*	*26*	*27*	*28*	*29*	*28*	*29*	*31*	*31*	*32*	*01*
Шығыс R_{i-1}^E	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*

5.7 мысал. Алмастыру және кеңейтудің *E*-блогының шығысындағы R_0^E тізбегін анықтау қажет, егер оның кірісіне $R_0 = 18ca18ad_{16}$ тізбегі кірсе.

Шешім. R_0 екілік мәнін қолданып

$$R_0 = 18ca18ad_{16} = 0001\ 1000\ 1100\ 1010\ 0001\ 1000\ 1010\ 1101_2$$

және 5.6 кестесі көмегімен ауыстыруды жасап, келесіні аламыз:

$$R_0^E = 1000\ 1111\ 0001\ 0110\ 0101\ 0100\ 0000\ 1111$$
$$0001\ 0101\ 0101\ 1010_2 = 8f16540f155_{16}.$$

F функцияның разряд бойынша қосу операциясы

Алмастыру және кеңейтудің *E*-блогын қолданумен *32* бит ұзындығы бар R_{i-1} тізбегін *48* бит ұзындығы бар R_{i-1}^E тізбегіне дейін кеңейтуден кейін оң жақ секцияның кеңейтілген бөлігіне R_{i-1}^E және раунд кілтіне k_i *xor* операциясын қолданады. Оң жақ секцияның R_{i-1}^E және кілтінің k_i ұзындығы *48* бит. Раунд кілті тек осы операцияны қолданады.

Сонымен *xor* операциясын орындағаннаң кейін *48* бит ұзындығы бар $R_{i-1}^{\oplus}$ тізбегін аламыз:

$$R_{i-1}^{\oplus} = R_{i-1}^E \oplus k_i. \qquad (5.5)$$

5.8 мысал. 2 модулі бойынша қосқыш шығысында $R_0^{\oplus}$ тізбегін анықтау қажет, егер оның кірісіне $R_0^E = 8f16540f155_{16}$ және раундтық кілт $k_1 = 194cd072de8c_{16}$ келсе.

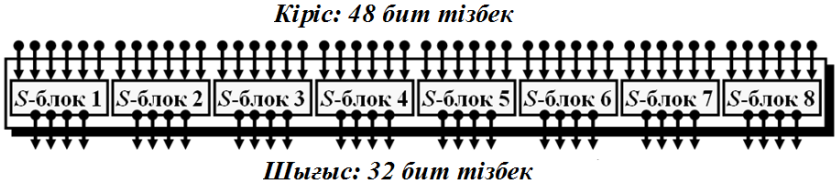
Шешім. R_0^E және k_1 мәндерін екілік кодқа ауыстырып және *2* модулі бойынша деректерді разрядтармен қосуды қолданып келесіні аламыз:

$$\begin{array}{cll} & R_0^E & 100011110001011001010100000011110001010101011010_2 \\ \oplus & k_1 & 000110010100110011010000011100101101111010001100_2 \\ \hline & R_0^{\oplus} & 100101100101101010000100011111011100101111010110_2 \end{array}$$

$R_0^{\oplus}$ мәнін он алтылық түрге түрлендіру түрі $R_0^{\oplus} = 965a847dcbd6_{16}$.

F функцияның ауыстыру және қысу операциясы

Ауыстыру және қысудың *S*-блоктары ақпаратты араластырады (араластыру операциясы). *DES* әр қайсысында *6* кіріс биты және *4* шығыс биты бар *S*-блоктарын қолданады (5.8 сурет).



5.8 сурет – ауыстыру және қысудың *S* - блоктары

48 биттен туратын $R_{i-1}^{\oplus}$ тізбегі *F* функциясының екінші операциясынан кейін *6* биттен сегіз секцияға бөлінеді, және әр секция ауыстыру және қысудың сәйкес *S*-блогына кіреді. Әр ауыстыру және қысудың *S*-блок нәтижесі – *4* бит ұзындығы бар деректер; олар біріккен кезде нәтижесі 32 биттік R_{i-1}^S тізбекпен көрсетіледі:

$$R_{i-1}^S = S(R_{i-1}^{\oplus}). \qquad (5.6)$$

R_{i-1}^{S} мәнін анықтау үшін есептелген қосынды $R_{i-1}^{\oplus}$ сегіз *6*-биттік сөздің конкатенациясы түрінде жазылады:

$$R_{i-1}^{\oplus} = B_1 \,||\, B_2 \,||\, B_3 \,||\, B_4 \,||\, B_5 \,||\, B_6 \,||\, B_7 \,||\, B_8 . \qquad (5.7)$$

Осы кезеңде әр сөз B_j сәйкес ауыстыру және қысудың *S*-блогына кіреді. *S*-блоктардың негізгі мақсаты *48*-биттік векторды *32*-биттіке түрлендіру. *DES* жалпы сегіз *6*-биттік кірісі және *4*-биттік шығысы бар *S*-блок қолданылады. S_j-блок – деген 4×16 көлемі бар матрица, оларға *0* ден *15* дейінгі бүтін элементтер кіреді.

Әр сөз B_j келесі түрде көрсетіледі:

$$B_j = \underbrace{b_1 \,||\, b_6}_{\substack{Жол \\ номірі}} \,||\, \underbrace{b_2 \,||\, b_3 \,||\, b_4 \,||\, b_5}_{\substack{Баған \\ номірі}} . \qquad (5.8)$$

B_j сөздің бірінші және алтыншы биттері, егер оларды саның екілік жазбасы ретінде қарастырсақ, S_j-блок матрицасының жол нөмірін анықтайды, ал қалған төрт бит баған нөмірін анықтайды. Берілген S_j-блоктың жол және баған нөмірлер қыйылысында матрица эедементі болады. Оның екілік жазбасы шығыс болады.

6-биттік блоктар жинағы (5.7) әр блокта *4*-биттік элементті таңдауды қамтамасыз етеді:

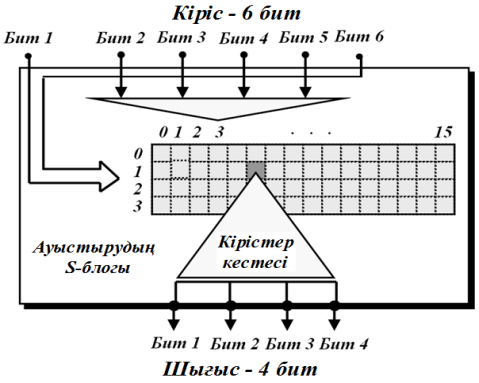
$$S = S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8 . \qquad (5.9)$$

Нәтижесінде аламыз:

$$R_{i-1}^{S} = S_1(B_1)||S_2(B_2)||S_3(B_3)||S_4(B_4)||S_5(B_5)||S_6(B_6)||S_7(B_7)||S_8(B_8)||.$$

Әр S_j-блок үшін өзінің кестесі болғандықтан, сегіз кесте қажет, мысалы, 5.7-5.14 кестелерде көрсетілгендей. Кіріс мәні (жол нөмірі және баған нөмірі) және шығыс мәндері бетте орын тиімдеу үшін ондың нөмірлер ретінде беріледі.

Олар екілік сандармен ауыстырылу мүмкін.



5.9 сурет. Ауыстыру және қысу *S*-блок үшін ережелер

5.7 кесте

Ауыстыру және қысудың S_1-блогы

	Баған нөмірлері																
	00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері *00*	*14*	*04*	*13*	*01*	*02*	*15*	*11*	*08*	*03*	*10*	*06*	*12*	*05*	*09*	*00*	*07*
01	*00*	*15*	*07*	*04*	*14*	*02*	*13*	*01*	*10*	*06*	*12*	*11*	*09*	*05*	*03*	*08*
02	*04*	*01*	*14*	*08*	*13*	*06*	*02*	*11*	*15*	*12*	*09*	*07*	*03*	*10*	*05*	*00*
03	*15*	*12*	*08*	*02*	*04*	*09*	*01*	*07*	*05*	*11*	*03*	*14*	*10*	*00*	*06*	*13*

5.8 кесте

Ауыстыру және қысудың S_2-блогы

	Баған нөмірлері																
	00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері *00*	*15*	*01*	*08*	*14*	*06*	*11*	*03*	*04*	*09*	*07*	*02*	*13*	*12*	*00*	*05*	*10*
01	*03*	*13*	*04*	*07*	*15*	*02*	*08*	*14*	*12*	*00*	*01*	*10*	*06*	*09*	*11*	*05*
02	*00*	*14*	*07*	*11*	*10*	*04*	*13*	*01*	*05*	*08*	*12*	*06*	*09*	*03*	*02*	*15*
03	*13*	*08*	*10*	*01*	*03*	*15*	*04*	*02*	*11*	*06*	*07*	*12*	*00*	*05*	*14*	*09*

5.9 кесте

Ауыстыру және қысудың S_3-блогы

	Баған нөмірлері																
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері	*00*	*10*	*00*	*09*	*14*	*06*	*03*	*15*	*05*	*01*	*13*	*12*	*07*	*11*	*04*	*02*	*08*
	01	*13*	*07*	*00*	*09*	*03*	*04*	*06*	*10*	*02*	*08*	*05*	*14*	*12*	*11*	*15*	*01*
	02	*13*	*06*	*04*	*09*	*08*	*15*	*03*	*00*	*11*	*01*	*02*	*12*	*05*	*10*	*14*	*07*
	03	*01*	*10*	*13*	*00*	*06*	*09*	*08*	*07*	*04*	*15*	*14*	*03*	*11*	*05*	*02*	*12*

5.10 кесте

Ауыстыру және қысудың S_4-блогы

	Баған нөмірлері																
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері	*00*	*07*	*13*	*14*	*03*	*00*	*06*	*09*	*10*	*01*	*02*	*08*	*05*	*11*	*12*	*04*	*15*
	01	*13*	*08*	*11*	*05*	*06*	*15*	*00*	*03*	*04*	*07*	*02*	*12*	*01*	*10*	*14*	*09*
	02	*10*	*06*	*09*	*00*	*12*	*11*	*07*	*13*	*15*	*01*	*03*	*14*	*05*	*02*	*08*	*04*
	03	*03*	*15*	*00*	*06*	*10*	*01*	*13*	*08*	*09*	*04*	*05*	*11*	*12*	*07*	*02*	*14*

5.11 кесте

Ауыстыру және қысудың S_5-блогы

	Баған нөмірлері																
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері	*00*	*02*	*12*	*04*	*01*	*07*	*10*	*11*	*06*	*08*	*05*	*03*	*15*	*13*	*00*	*14*	*09*
	01	*14*	*11*	*02*	*12*	*04*	*07*	*13*	*01*	*05*	*00*	*15*	*10*	*03*	*09*	*08*	*06*
	02	*04*	*02*	*01*	*11*	*10*	*13*	*07*	*08*	*15*	*09*	*12*	*05*	*06*	*03*	*00*	*14*
	03	*11*	*08*	*12*	*07*	*01*	*14*	*02*	*13*	*06*	*15*	*00*	*09*	*10*	*04*	*05*	*03*

5.12 кесте

Ауыстыру және қысудың S_6-блогы

	Баған нөмірлері																
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері	*00*	*12*	*01*	*10*	*15*	*09*	*02*	*06*	*08*	*00*	*13*	*03*	*04*	*14*	*07*	*05*	*11*
	01	*10*	*15*	*04*	*02*	*07*	*12*	*09*	*05*	*06*	*01*	*13*	*14*	*00*	*11*	*03*	*08*
	02	*09*	*14*	*15*	*05*	*02*	*08*	*12*	*03*	*07*	*00*	*04*	*10*	*01*	*13*	*11*	*06*
	03	*04*	*03*	*02*	*12*	*09*	*05*	*15*	*10*	*11*	*14*	*01*	*07*	*06*	*00*	*08*	*13*

5.13 кесте

Ауыстыру және қысудың S_7-блогы

	Баған нөмірлері																
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері	*00*	*04*	*11*	*02*	*14*	*15*	*00*	*08*	*13*	*03*	*12*	*09*	*07*	*05*	*10*	*06*	*01*
	01	*13*	*00*	*11*	*07*	*04*	*09*	*01*	*10*	*14*	*03*	*05*	*12*	*02*	*15*	*08*	*06*
	02	*01*	*04*	*11*	*13*	*12*	*03*	*07*	*14*	*10*	*15*	*06*	*08*	*00*	*05*	*09*	*02*
	03	*06*	*11*	*13*	*08*	*01*	*04*	*10*	*07*	*09*	*05*	*00*	*15*	*14*	*02*	*03*	*12*

5.14 кесте

Ауыстыру және қысудың S_8-блогы

	Баған нөмірлері																
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Жол нөмірлері	*00*	*13*	*02*	*08*	*04*	*06*	*15*	*11*	*01*	*10*	*09*	*03*	*14*	*05*	*00*	*12*	*07*
	01	*01*	*15*	*13*	*08*	*10*	*03*	*07*	*04*	*12*	*05*	*06*	*11*	*00*	*14*	*09*	*02*
	02	*07*	*11*	*04*	*01*	*09*	*12*	*14*	*02*	*00*	*06*	*10*	*13*	*15*	*03*	*05*	*08*
	03	*02*	*01*	*14*	*07*	*04*	*10*	*08*	*13*	*15*	*12*	*09*	*00*	*03*	*05*	*06*	*11*

DES керішифрлаудың аналитикалық қыйындығы ауыстыру *S*-блогының математикалық қасиеттерінен тәуелді, олардың ішінде сызықты емес түрлендірулер орындалатындықтан.

Бұл алгоритмдегі барлық қалған операциялар сызықты. Криптографиялық талдаушы үшін сызықты түрлендіруді аналитикалық есептеу қыйындық тұдырмайды.

DES орындалатын түрлендірулердің сызықты еместігі тек ауыстыру және қысудың *S*-блоктарымен анықталады. Оларды таңдаудың қажетті негіздеуі жоқ. Ауыстыру және қысудың *S*-блоктарында шифрланған жазбаларды бақылау үшін кейбір «тесік» бар деген ойлар айтылды.

Ресми нұсқа келесі: 1976 жылы АҚШ Стандарттардың ұлттық бюросы ауыстыру және қысудың *S*-блоктарын таңдау келесі талаптарымен анықталу қажет екенін айтты [18, 29, 40]:

- әр кестемен берілген ауыстыру және қысудың *S*-блогының әр жолы *{0,1, ... , 15}* көптігінің алмастырыуы болу қажет;

- ауыстыру және қысудың *S*-блоктары өз кірістерінің сызықты немесе аффинды функциялары болмау қажет;

- ауыстыру және қысудың *S*-блогының бір кіріс битын өзгерту шығыстың кемінде екі битын өзгертуге алып келу қажет;

- әр ауыстыру және қысудың *S*-блогы және кез келген кіріс *x* үшін *S(x)* және $S(x \oplus (0,0,1,0,0,0))$ мәндері кемінде екі битпен айырмашылығы болу қажет;

S-блоктар көмегімен ауыстыру процесстерін мысалдар негізінде түсіндіреміз.

5.9 мысал. Ауыстыру және қысудың S_1-блогының кіріс тізбегі $\underline{1}\,0\,1\,1\,1\,\underline{0}$ тең. *S* –блок шығысындағы тізбекті анықтау.

Шешім. Егер бірінші және алтыншы биттерді бірге жазсақ, онда екілік жүйесінде 10_2 аламыз, ол ондық жүйесінде 2_{10} тең болады. Қалған биттер екілік жүйесінде 0111_2 ондық жүйесінде 7_{10} болады.

5.7 кестесінде *2* жол және *7* баған қыйылысында (ауыстыру және қысудың S_1-блогы) нәтиже мәні – 11_{10} ондық жүйеде және 1011_2 екілік жүйеде. Онда кіріс 101110_2 1011_2 шығысты береді.

5.10 мысал. Ауыстыру және қысудың S_8-блогының кіріс тізбегі $\underline{0}\,0\,0\,0\,0\,\underline{0}$ тең. S_8 –блок шығысындағы тізбекті анықтау.

Шешім. Егер бірінші және алтыншы биттерді бірге жазсақ, онда екілік жүйесінде 00_2 аламыз, ол ондық жүйесінде 0_{10} тең болады. Қалған биттер екілік жүйесінде 0000_2 ондық жүйесінде 0_{10} болады.

5.14 кестесінде *0* жол және *0* баған қыйылысында (ауыстыру және қысудың S_8-блогы) нәтиже мәні – 13_{10} ондық жүйеде және 1101_2 екілік жүйеде. Онда кіріс 000000_2 1101_2 шығысты береді.

F функцияның түзу алмастыру операциясы

Түзу алмастырудың *P*-блогы - *F* функциядағы соңғы операция - *32* бит кірісі және *32* бит шығысы бар түзу алмастыру:

$$R_{i-1}^{P} = P(R_{i-1}^{S}). \qquad (5.10)$$

Бұл операция үшін «кіріс-шығыс» қатынастары 5.15 кестесінде көрсетілген. Олар алдынғы алмастыру кестелері сияқты жалпы ережелерге сай. Мысалы, кірістің жетінші биты шығыстың екінші биты болады.

5.11 мысал. Түзу алмастырудың *P*-блогының кіріс тізбегі $R_0^S = ec5965b5_{16}$. Түзу алмастырудың *P*-блогының шығысында қандай тізбек болатынын анықтау.

Шешім. R_0^S екілік мәнін қолданып:

$$R_0^S = ec5965b5_{16} = 1110\ 1100\ 0101\ 1001\ 0110\ 0101\ 1011\ 0101_2$$

және 5.15 кестесі көмегімен ауыстыруды жүргізіп келесіні аламыз:

$$R_0^P = 1000\ 0110\ 1000\ 1101\ 1010\ 1110\ 1111\ 1001_2 = 868daef9_{16}.$$

5.15 кесте

Түзу алмастыру *P*-блок ережелері

Кіріс R_{i-1}^S	*16*	*07*	*20*	*21*	*29*	*12*	*28*	*17*	*01*	*15*	*23*	*26*	*05*	*18*	*31*	*10*
Шығыс R_{i-1}^P	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Кіріс R_{i-1}^S	*02*	*08*	*24*	*14*	*32*	*27*	*03*	*09*	*19*	*13*	*30*	*06*	*22*	*11*	*04*	*25*
Шығыс R_{i-1}^P	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*

5.5.2. DES рауындында разряд бойынша қосу

Түзу алмастырудың *P*-блогы көмегімен түрлендіру нәтижесі *2* модулі (*xor* операцияның нәтижесі) бойынша *32*-биттік L_{i-1} тізбегімен қосылады және *32*-биттік R_{i-1}^L нәтиже алынады (5.6 суретке қараңыз):

$$R_{i-1}^L = R_{i-1}^P \oplus L_{i-1}. \qquad (5.11)$$

5.12 мысал. DES раундының *2* модулі бойынша қосқыш шығысында R_0^L тізбегін анықтау қажет, егер оның кірісіне $R_0^P = 868daef9_{16}$ және $L_0 = acf014a7_{16}$ тізбектері кірсе.

Шешім. R_0^P және L_0 мәндерін екілік түрде көрсетіп және *2* модулі бойынша деректерді қосу ережесін қолданып, аламыз:

$$\begin{array}{cll} & R_0^P & 10000110100011011010111011111001_2 \\ \oplus & L_0 & 10101100111100000001010010100111_2 \\ \hline & R_0^L & 00101010011111101011101001011110_2 \end{array}$$

R_{i-1}^L он алтылық мәнге түрлендіру түрі $R_{i-1}^L = 2a7dba5e_{16}$.

5.5.3. DES раунд секцияларын ауыстыру құралы

Секцияларды ауыстыру құрылғысы *DES* бір раунд операцияларын аяқтау үшін арналған, яғни ол келесі раунд үшін L_i және R_i тізбектерінің мәндерін қалыптастырады. (5.2) және (5.11) сәйкес L_i және R_i тізбектері келесі ретпен анықталады:

$$\begin{aligned} L_i &= R_{i-1}; \\ R_i &= R_{i-1}^L; \quad i = 1,2,\ldots,16. \end{aligned} \tag{5.12}$$

5.13 мысал. L_1 және R_1 тізбектерін деректерді шифрлаудың екінші раунды үшін анықтау қажет, егер ауыстыру құрылғысының кірісіне $R_0 = 305a18ca_{16}$ және $R_0^L = 2a7dba5e_{16}$ тізбектері берілсе.

Шешім. R_0 және R_0^L мәндерін және (5.12) өрнектерін қолдана отырып, аламыз:

$$L_1 = R_0 = 305a18ca_{16}; \quad R_1 = R_0^L = 2a7dba5e_{16}.$$

5.5.4. DES деректерді шифрлау алгоритмі

Араластырғышты және ауыстыру құрылғысын қолданып *16* раундтың әрқайсысы үшін түзу және қайтымды шифрды құруға болады. Түзу шифр шифрлау жағында қолданылады; қайтымды шифр – керішифрлау жағында.

(5.2)…(5.6) және (5.10)…(5.12) өрнектерге сәйкес шифрлау алгоритмді келесі түрде көрсетуге болады[21, 22]:

$$\begin{aligned} C = IP(L_0,R_0) \times F(L_0,R_0 \oplus k_1) \times \\ \times F(L_1,R_1 \oplus k_2) \times \ldots \times F(L_{15},R_{15} \oplus k_{16}) \times IP^{-1}(R_{16},L_{16}). \end{aligned} \tag{5.13}$$

(5.13) шығатындай деректерді шифрлау процессінде тізбекті түрде *IP*, $F(\bullet)$ және IP^{-1} түрлендірулері қолданылады.

Фейстельдің әр түрлендіруі $F(\bullet)$ екі түрлендірудің композициясы болады.

Біріншісі – L_i және R_i транспозициясы (алмастыру) [21, 22]:

$$T(L_i,R_i) = (L_i,R_i),$$

бұл жерде T – деректер блогының сол жақ L_i және оң жақ R_i жартыларын орнымен ауыстыру бойынша операция (транспозиция операциясы).

Екіншісі – нақты түрде *32*-биттік сөздерді *Вернам* түрлендіруі болады, яғни [21, 22]:

$$V(L_i,R_i) = (L_i,R_i \oplus C),$$

бұл жерде V – *Вернам* түрлендіруі; C – раундтық кілт k_i және оң жақ секция L_i тәуелді тізбек;

Сонымен, кез келген раунд үшін

$$F_i(\bullet) = T_i(\bullet) \times V_i(\bullet).$$

Одан басқа [21, 22]:

$$V^2_i(\bullet) \equiv T^2_i(\bullet)$$

теңдік түрлендіру екеніне оңай көз жеткізуге болады. Осындай түрлендірулерді (топ элементтері) *инволюциялар* деп атайды. Олар үшін келесі орындалады:

$$V_i^{-1}(\bullet) = V_i(\bullet), \quad T_i^{-1}(\bullet) = T_i(\bullet).$$

Абельемес топ элементтер көбейтіндісін түрлендіру ережесі бойынша

$$M = C^{-1} = IP^{-1}(L_0,R_0) \times V(L_0 \oplus F(R_0 \oplus k_1)) \times T \times$$
$$\times V(L_1,F(R_1 \oplus k_2)) \times T \times \ldots \times V(L_{15},F(R_{15}) \oplus k_{16})) \times T \times \qquad (5.14)$$
$$\times T \times IP(R_{16},L_{16}).$$

(5.14) түрлендірулерді жүзеге асырып деректерді керішифрлау алгоритмін аламыз

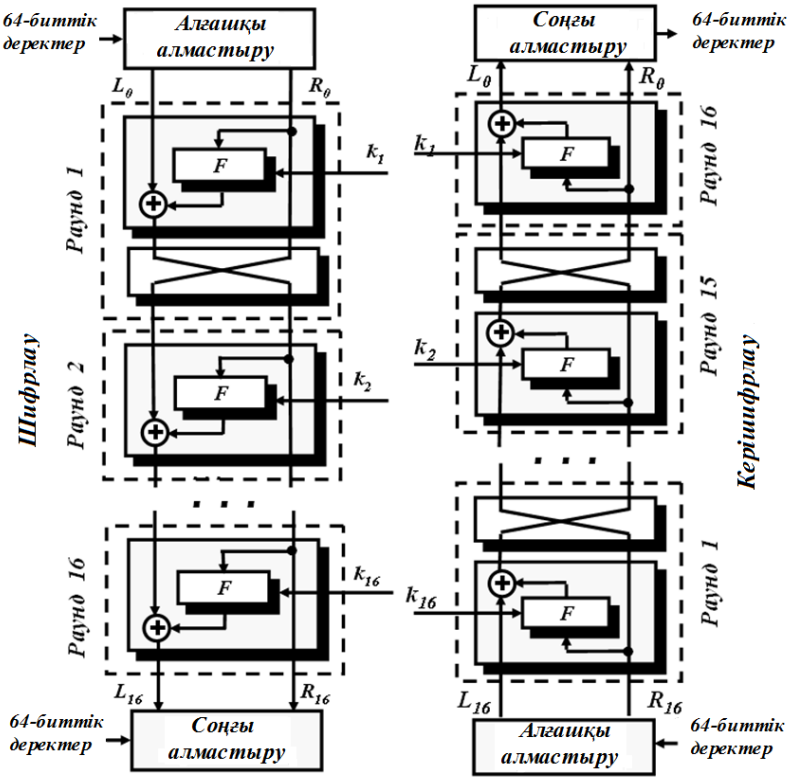
$$M = IP(R_{16},L_{16}) \times F(R_{15} \oplus F(L_{15} \oplus k_{16})) \times$$
$$\times F(R_{14},F(L_{14} \oplus k_{15})) \times \ldots \times F(R_1,F(L_1) \oplus k_1)) \times IP^{-1}(L_0,R_0). \qquad (5.15)$$

Бұл деген керішифрлау сол шифрлау алгоритмімен (5.13) және раунд кілттерімен жүзеге асатынын білдіреді, бірақ раунд кілттер кестесіне (раундтарда қолдану ретіне) кейбір өзгеріс енгізу қажет: раунд кілттерін генерациялау ретін керісіне ауыстыру қажет, яғни керішифрлау кезінде раунд кілттерін таңдау кері болады:

$$k_{16}, k_{15}, k_{14}, k_{13}, k_{12}, k_{11}, k_{10}, k_9, k_8, k_7, k_6, k_5, k_4, k_3, k_2, k_1,$$

егер шифрлау процессінде раунд кілттерін таңдау реті келесі болса

$$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}.$$



5.10 сурет - *DES* бірінші тәсіл үшін түзу және қайтымды шифр

Қойылған мақсатқа (шифрлау және керішифрлау) жету үшін әдістердің бірі келесіде: соңғы раундта басқаларға қарағанда айырмашылық енгізуде; ода 5.10 суретінде көрсетілгендей араластырғыш ғана болады және ауыстыру құрылғысы болмайды.

3 тарауда араластырғыш және ауыстыру құрылғысы өзараинверсты екені дәлелденген. Алғашқы және соңғы алмастырулары да бір біріне инверсты. Алғашқы деректердің сол жақ секциясы шифрлау жағында шифрланады: L_0 - L_{16} сияқты, ал L_{16} керішифрлау жағында L_0 керішифрланады. Тұра сондай әрекеттер R_0 және R_{16} орындалады.

Бірінші әдісте соңғы раундта ауыстыру құрылғысы жоқ.

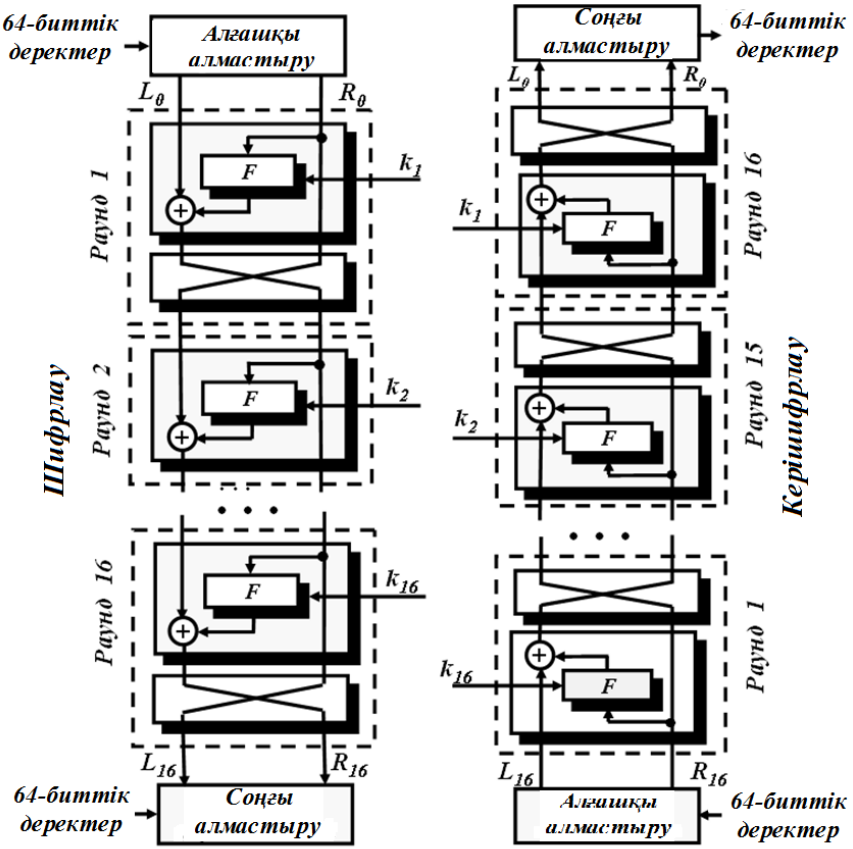
5.14 мысал. DES түзу және қайтымды шифр үшін псевдокодтар. Шифрлау үшін псевдокодта мысал келтірілген және бірінші әдістің төрт қадамына сәйкес келеді. Қолғандары үшін кодтар жаттығу ретінде жасалу мүмкін.

```
Cipher (plainBlock[64], Round Keys[16,48])
{
    pemute (64,64, plainBlock, inBlock, IntialPermutationTable)
    spilt (64, 32, inBlock, leftBlock, right Block)
    for (round = 1 to 16)
    {
            mixer (leftBlock, right Block, RoundKey[round])
            if (round!=16) swapper(leftBlock, right Block)
    }
    combine (32, 64, leftBlock, right Block, outBlock)
    pemute (64,64, outBlock, cipherBlock, FinalPermutationTable)
    mixer (leftBlock[48], right Block[48], RoundKey[48])
    {
            copy (32, rightBlock, T1)
            function (T1, RoundKey,T2)
            exclusive Or (32, leftBlock, T2,T3)
            copy (32, rightBlock, T1)
    }
    swapper (leftBlock[32], right Block[32])
    {
            copy (32, leftBlock, T)
            copy (32, rightBlock, leftBlock)
            copy (32, T, rightBlock)
    }
    Substitute (in block[32], outblock[48], SubstituteTables[8,4,16])
```

```
    {
        for(I = 1 to 8)
        {
            row ← 2 × inBlock[i × 6+1] + inBlock[i × 6+6]
            col ← 8 × inBlock[i × 6+2] + 4 × inBlock[i ×
            × 6+3]+2 × inBlock[i × 6+4] + inBlock[i × 6+5]
            value = SubstituteTables[i][row][col]
            outBlock[i × 4+1] ← value/ 8
            value ← value mod 8
            outBlock[i × 4+2] ← value/ 4
            value ← value mod 4
            outBlock[i × 4+3] ← value/ 2
            value ← value mod 8
            outBlock[i × 4+4] value
        }
    }
}
```

Бірінші тәсілде *16* раундта ауыстыру құрылғысы қолданылмайды, сонымен ол басқа раундтарға ұқсамайды. Бұл шифрда соңғы және бірінші араластырғыштарды бірдей жасау үшін қажет.

Екінші тәсілде барлық *16* раундты бірдей жасауға болады, *16-шы* раундқа қосымша ауыстыру құрылғысын (екі ауыстыру құрылғылары бір бірін бейтараптандырады) қосу арқылы. Берілген түзу және қайтымды шифр сұлбасы 5.11 суретінде көрсетілген.



5.11 сурет - *DES* екінші тәсіл үшін түзу және қайтымды шифр

5.6. DES ДЕРЕКТЕРДІ ШИФРЛАУ МЫСАЛДАРЫ

DES талдау алдында шифрлау және керішифрлау әр раундта биттер мәнін қалай өзгертетінін түсіні үшін бірнеше мысал қарастырамыз.

5.15 мысал. Алғашқы деректердің кездейсоқ блогын және шифрдың кездейсоқ кілтін таңдаймыз және екінші шифрлау тәсілін (барлық сандар он алтылық жүйеде берілген) қолданған кезде шифрланған деректер блогы қандай болатынын анықтаймыз:

- кіріс деректер: $123456abcd132536_{16}$;
- шифр кілті: $abba08192637cddc_{16}$.

5.16 кесте әр раунд нәтижесін және раунд алдында және соңында құрылғын деректерді көрсетеді.

5.16 кесте

5.16 мысалында деректер трассировкасы

Раундтар	*Сол жақ секциялары* L_i	*Оң жақ секциялары* R_i	*Раундтық кілттер* k_i
1	$18ca18ad_{16}$	$5a78e394_{16}$	$194cd072de8c_{16}$
2	$5a78e394_{16}$	$4a1210f6_{16}$	$4568581abcce_{16}$
3	$4a1210f6_{16}$	$b8089591_{16}$	$06eda4acf5b5_{16}$
4	$b8089591_{16}$	$236779c2_{16}$	$da2d032b6ee3_{16}$
5	$236779c2_{16}$	$a15a4b87_{16}$	$69a629fec913_{16}$
6	$a15a4b87_{16}$	$2e8f9c65_{16}$	$c1948e87475e_{16}$
7	$2e8f9c65_{16}$	$a9fc20a3_{16}$	$708ad2ddb3c0_{16}$
8	$a9fc20a3_{16}$	$308bee97_{16}$	$34f822f0c66d_{16}$
9	$308bee97_{16}$	$10af9d37_{16}$	$84bb4473dccc_{16}$
10	$10af9d37_{16}$	$6ca6cb20_{16}$	$02765708b5bf_{16}$
11	$6ca6cb20_{16}$	$ff3c485f_{16}$	$6d5560af7ca5_{16}$
12	$ff3c485f_{16}$	$22a5963b_{16}$	$c2c1e96a4bf3_{16}$
13	$22a5963b_{16}$	$387ccdaa_{16}$	$99c31397c91f_{16}$
14	$387ccdaa_{16}$	$bd2dd2ab_{16}$	$251b8bc717d0_{16}$
15	$bd2dd2ab_{16}$	$cf26b472_{16}$	$3330c5d9a36d_{16}$
16	$cf26b472_{16}$	$19ba9212_{16}$	$181c5d75c66d_{16}$

Кесте *16* раунд нәтижелерін көрсетеді, оларға араластырғыш және ауыстыру (соңғы раундты есептемегенде) кіреді.

Алғашқы (ашық) деректер – әртүрлі *64* битті алу үшін алғашқы алмастырудан өткен (*16* он алтылық сандар): $14a7d67818ca18ad_{16}$.

Оң жақ және сол жақ секцияларға бөлуден кейін: $L_0 = 14a7d678_{16}$ және $R_0 = 18ca18ad_{16}$.

16 раунд және біріктіруден кейін: $L_{16}||R_{16} = cf26b47219ba9212_{16}$.

Сонғы алмастыруды орындау нәтижесінде шифрланған деректерді аламыз:

$$C = c0b7a8d05f3a829c_{16}.$$

Кейбір ережелерді белгілеп кету қажет. Әр раундтың оң жақ секциясы келесі раундтың сол жақ секциясымен тең. Себебі келесіде: оң жақ секция араластырғыштан өзгерусіз өтеді, ал ауыстыру құрылғысы оны сол жақ секциясына ауыстырады.

Мысалы, R_1 екінші раунд араластырғышынан өзгертусіз жіберіледі, бірақ ауыстыру құрылғысын өткенен кейін ол L_2 болып қалады.

5.16 мысал. Тағайындау нүктесінде қабылдаушы жіберушіден алынған шифрланған деректерді сәйкес келетін кілт арқылы қалай керішифрлай алатынын қарастырайық. 5.17 кестесі раунд алдында және соңында құрылған әр раунд нәтижесін және деректерді көрсетеді.

5.17 кесте

5.17 мысалында деректер трассировкасы

Раундтар	*Сол жақ секциялары* L_i	*Оң жақ секциялары* R_i	*Раундтық кілттер* k_i
1	$cf26b472_{16}$	$bd2dd2ab_{16}$	$181c5d75c66d_{16}$
2	$bd2dd2ab_{16}$	$387ccdaa_{16}$	$3330c5d9a36d_{16}$
3	$387ccdaa_{16}$	$22a5063b_{16}$	$251b8bc717d0_{16}$
4	$22a5063b_{16}$	$ff3c485f_{16}$	$99c31397c91f_{16}$
5	$ff3c485f_{16}$	$6ca6cb20_{16}$	$c2c1e96a4bf3_{16}$
6	$6ca6cb20_{16}$	$10af9d37_{16}$	$6d5560af7ca5_{16}$
7	$10af9d37_{16}$	$308bee97_{16}$	$02765708b5bf_{16}$
8	$308bee97_{16}$	$a9fc20a3_{16}$	$84bb4473dccc_{16}$
9	$a9fc20a3_{16}$	$2e8f9c65_{16}$	$34f822f0c66d_{16}$
10	$2e8f9c65_{16}$	$a15a4b87_{16}$	$708ad2ddb3c0_{16}$
11	$a15a4b87_{16}$	$236779c2_{16}$	$c1948e87475e_{16}$
12	$236779c2_{16}$	$b8089591_{16}$	$69a629fec913_{16}$
13	$b8089591_{16}$	$4a1210f6_{16}$	$da2d032b6ee3_{16}$
14	$4a1210f6_{16}$	$5a78e394_{16}$	$06eda4acf5b5_{16}$
15	$5a78e394_{16}$	$18ca18ad_{16}$	$4568581abcce_{16}$
16	$18ca18ad_{16}$	$14a7d678_{16}$	$194cd072de8c_{16}$

Бірінші ереже: раунд кілттері кері ретпен қолдану керек. 5.16 кестесін және 5.17 кестесін салыстырыңыз. *1* раунд кілті *16* раунд кілтімен бірдей. Керішифрлау кезінде L_0 және R_0 мәндері шифрлау кезіндегі R_{16} және L_{16} мәндеріне тең. Осындай сәйкестіктер басқа да раундтарда болады. Бұл түзу және қайтымды шифрлар инверсты екенін ғана емес, сонымен қатар шифрлау кезінде әр раундтың кері ретпен керішифрлау кезіндегі сәйкес раунды болатынын дәлелдейді. Нәтиже алғашқы және сонғы алмастырулар бір біріне инверсты екенін куәландырады.

Шифрланған деректер: $C = c0b7a8d05f3a829c_{16}$.

Алғашқы алмастырудан кейін: $19ba9212cf26b472_{16}$.

Сол және оң секцияларға бөлгеннең кейін: $L_0 = 19ba9212_{16}$ және $R_0 = cf26b472_{16}$.

Біріктіруден кейін: $L_{16}||R_{16} = 18ca18ad14a7d678_{16}$.

Соңғы алмастыруды орындаудан кейін керішифрланған деректерді аламыз:

$$L_{16}||R_{16} = 18ca18ad14a7d678_{16}.$$

5.7. DES АЛГОРИТМІН ТАЛДАУ

DES блокты шифрда кейбір қажетті қасиеттердің қарқындылығын өлшеу үшін қатаң талдалды. *DES* элементтері кейбір критерийлерге сәйкестікке зерттеуден өтті.

5.7.1. DES тасқын эффектісі және толықтық эффектісі

Блокты шифрдың екі қажетті қасиеттері – *тасқын эффектісі* және *аяқталғандық (толықтық эффектісі)*.

Тасқын эффектісі алғашқы деректерде (немесе шифр кілтінде) үлкен емес өзгерістер шифрланған деректерде мәнді өзгерістерге алып келу мүмкін. *DES* осы қасиеттің барлық признактары бар екені дәлелденді [20, 37, 40].

5.17 мысал. DES тасқын эффектісін тексеру үшін бір битке айырмашылығы бар алғашқы деректердің екі блогын бір кілтпен шифрлаймыз және әр раундта қанша битте айырмашылығы бар екенін анықтаймыз.

Бірінші жағдай:

- ашық деректер: 0000000000000000_{16};
- шифр кілті: $23234513987aba23_{16}$;
- шифрланған деректер: $4789fd476e82a5f1_{16}$.

Екінші жағдай:

- ашық деректер: 0000000000000001_{16};
- шифр кілті: $23234513987aba23_{16}$;
- шифрланған деректер: $0a4ed5c15a63fea3_{16}$.

Ашық мәтіннің екі блогында тек ең бірінші битке айырмашылығы болса да, шифрланған мәтіндер блоктардың *29* битке айырмашылықтары бар. Бұл деген ашық деректерде $\approx 1{,}5\,\%$ өзгерістер шифрланған деректерде $\approx 45\,\%$ өзгеруіне алып келетінін білдіреді.

5.18 кесте екі берілген жағдайда L_i және R_i салыстыру бойынша әр раундта өзгерістерді көрсетеді. Мәнді өзгерістер үшінші раундта пайда болатынын көруге болады.

5.18 кесте

5.17 мысалында әртүрлі биттер саны

Раунд	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Биттерде айырмашылықтар	*1*	*6*	*20*	*29*	*30*	*33*	*32*	*29*	*32*	*39*	*33*	*28*	*30*	*31*	*30*	*29*

Толықтық эффектісінде шифрланған деректердің әр биты алғашқы деректердің көптеген биттерінен тәуелді болу керек. *DES* алмастыру және кеңейту *E*-блоктарымен, ауыстыру және қысудың *S*-блоктарымен және түзу алмастырудың *P*-блоктарымен жасалатын араластыру және шашырату өте күшті толықтық эффектісін көрсетеді.

5.7.2. DES құру критерийлері

DES көптеген сынаулары ол берілген критерийлердің кейбіреулеріне сәйкес келетінін көрсетті. Төменде *DES* құрудың кейбір проблемалары қарастырылған.

S-блоктарын құру проблемалары

3 тарауда *S*-блоктарды құрудың жалпы критерийлері қарастырылған болатын. *DES* үшін таңдалған критерийлерді қарастырайық. Блоктардың құрылымы әр раундтан келесігедейін араластыру және шашыратуды қамтамасыз етеді. Осы ережеге және кейбір талдауға сәйкес, ауыстыру және қысу *S*-блоктарының бірнеше қасиеттерін айтып кетуге болады.

1. Әр жолдың шығысы *0* және *15* аралығындағы мәндердің алмасуы.

2. *S*-блоктар сызықты емес. Басқа сөзбен, шығыс – аффинды түрлендіру емес.

3. Егер кірісте бір битты өзгертсе, шығыста екі немесе одан көп бит өзгереді.

4. Егер *S*-блоктың екі кірісінде тек екі орташа биттерде (*3* және *4* биттер) айырмашылық болса, онда шығыс ақпаратта кемінде екі

биттерде айырмашылық болу қажет. Басқа сөзбен, $S(x)$ және $S(x \oplus 001100)$ кемінде екі биттерде айырмашылық болу қажет, бұл жерде x – кіріс және $S(x)$ – шығыс.

5. Егер *S*-блокқа екі кірісте бірінші екі битте (*1* және *2* биттер) және екі соңғы битте (*5* және *6*) айырмашылық болса, онда екі шығыс әртүрлі болу қажет. Басқа сөзбен, бізде келесі қатынас болу қажет: $S(x) \neq S(x \oplus 0011bc00)$, бұл жерде b және c – кез келген биттер.

6. $X_i \oplus x_j \neq 000000_2$ болатын тек *32* алтыбитты «кіріс-шығыс» (x_i және x_j) жұптары бар. Бұл *32* кіріс жұп шығыстың *4* биттен *32* жұп сөздерін құрады. Егер *32* кіріс жұптары арасында кез келген айырмашылық құрса $d = y_i \oplus y_j$, онда осы d ішінде *8* жұптан көп емесі бірдей болу қажет.

7. Алтыншы пункттағыдай критерий үш *S*-блокқа қолданылады.

8. Кез келген *S*-блокта, егер жалғыз кіріс биты тұрақты шама (*0* немесе *1*) ретінде сақталса, онда басқа биттер нөл және бір саны арасындағы айырмашылық минималды болатындай өзгереді.

E-блоктарды және P-блоктарды құру проблемалары

S-блоктардың қатарлар арасында (екі келесі раундта), бір алмастыру және кеңейту (*32*-ні *48*-ге) *E*-блогы және бір түзу алмастыру (*32*-ні *32*-ге) *P*-блогы бар. Бұл екі блок бірге биттердің шашырауын қамтамасыз етеді. Ауыстыру блогын құрудың жалпы принциптері туралы 3 тарауда айтылған. Тек *DES* қолданатын қолданбалы блоктарды қарастырамыз. *DES* блоктарының құрылымында келесі критерийлер жүзеге асырылды:

1. *S*-блоктың әр кірісі басқа *S*-блоктың шығысына жалғанады (алдынғы раундта).

2. Берілген *S*-блоктың бірде бір кірісі сол блоктың шығысына жалғанбайды (алдынғы раундта).

3. Әр *S*-блоктан төрт бит алты әртүрлі *S*-блокқа барады (келесі раундта).

4. *S*-блок шығысының екі биттың біреуіде сол *S*-блокқа бармайды (келесі раундта).

5. Ауыстыру S_{j-2}-блок шығысы ауыстыру S_j-блогының бірінші екі битының біреуіне барады (келесі раундта).

6. Ауытыру S_{j-1}-блогынан шығыс биты ауыстыру S_j-блоктың соңғы екі битының біреуіне барады (келесі раундта).

7. Ауыстырудың S_{j+1}-блок шығысы ауыстыру S_j-блоктың орташа биттерінің біреуіне барады (келесі раундта).

8. Әр ауыстырудың *S*-блогы үшін шығыстың екі биты келесі раундта ауыстыру *S*-блогының бірінші немесе соңғы екі битына барады. Шығыстың басқа екі биты келесі раундта ауыстыру *S*-блогының орташа биттеріне барады.

9. Егер ауыстыру S_j –блок шығысы ауыстыру S_k –блогының орташа биттерінің біреуіне барса (келесі раундта), онда ауыстыру S_k-блогынан шығыс ауыстырудың S_j-блогының орташа битіне бара алмайды. Егер $j = k$ десек, онда ауыстыру *S*-блогының орташа биттері келесі раундта сол ауыстыру *S*-блогының орташа биттеріне бара алмайды.

Раундтар саны

DES Фейстель шифрының он алты раундын қолданады. Алғашқы деректердің әр блогы сегіз раундпен шифрланған соң, шифрланған деректердің әр биты – алғашқы деректер блогының әр битының және әр кілттік биттың функциясы екені дәлелденген [20, 37, 40]. Шифрланған деректер – алғашқы деректер және кілттің толық кездейсоқ функциясы. Осыдан жақсы шифрлау үшін сегіз раунд жеткілікті екені шығады. Бірақ, сынаулар көрсетеді, *DES* он алты раундтан аз нұсқалары алғашқы деректерді білу шабуылдарына беріктілігі «қатқыл күш» шабуылынан төмен екенін көрсетеді. «Қатқыл күш» шабуылы он алты раундты қолдануды талап етеді.

5.7.3. DES әлсіз жерлер

Өткен көп жылдар ішінде криптографиялық талдаушылар *DES* кейбір әлсіз (осал) жерлері табылды. Шифр құрылымында табылған кейбір әлсіз жерлерін қарастырамыз.

Ауыстыру және қысудың S-блоктары. Әдебиетте *S*-блоктардың кем дегенде үш блогы көрсетіледі:

1. S_4–блогында шығыстың үш биты шығыстың бірінші биты алынатын тәсілмен алыну мүмкін: кіріс биттың кейбіреуін толықтырумен.

2. *S*–блок массивіне екі арнайы таңдалған кіріс тұра сол шығысты құру мүмкін.

3. Үш көрші *S*-блоктарда биттерді өзгерте отырып, бір жалғыз раундта сол шығысты алуға болады.

Алғашқы IP және сонғы IP^{-1} *алмастыру блоктары.* Осы блоктар құрылымында бір жұмбақ табылды. Неге *DES* жобалаушылары алғашқы және сонғы алмастыруларын қолданғаны белгісіз. Бұл алмастырулар қауіпсіздік көз қарасынан ешқандай жаңа қасиеттерді енгізбейді.

Алмастыру және кеңейтудің E-блоктары. Алмастыру және кеңейту блоктарында тізбектердің бірінші және төртінші биттері *4* битке қайталанады.

5.7.4. DES шифрының кілттегі әлсіздігі

DES шифрының кілт көлемі

Сыншылар *DES* ең ауыр әлсіздігі – кілт көлемі (*56* бит) деп сендіреді. Шифрланған деректер блогына «қатқыл күш» шабуылын қолдану үшін, қасқой 2^{56} кілттерді тексеру керек:

1. Бүгінгі күні қолжеткізілімді технологияны қолданып, секундына бір миллион кілтті тексеруге болады. Бұл деген бір процессоры бар компьютерді қолданып, *DES* шифрына «қатқыл күш» шабуылын орындау үшін екі мың жылдан аса уақыт керек екенін білдіреді.

2. Егер бір миллион процессор чиптері (қатарлас өңдеу) бар компьютерді жасауға болатын болса, онда барлық кілттер көптігін шамамен *20* сағатта тексеруге болады. *DES* ендірген кезде сондай компьютердің бағасы бірнеше миллион доллар болатын, бірақ ол тез төмендеді. Арнайы компьютер 1998 жылы жасалды, оның көмегімен кілт *112* сағатта табылды.

3. Компьютерлік желілер қатарлас өңдеуді модельдеу мүмкін. 1977 жылы зерттеушілер тобы *3500 Internet-ке* қосылған компьтерді *DES* кілтін *120* күнде табу үшін қолданды. Кілттер көптігі осы компьютерлер арасында бөлінді, және әр компьютер *DES* доменінің бөлігін тексеруге жауапты болды. Егер *3500* желіге байланысқан компьютер кілтті *120* күнде тапса, онда *42000* мүшеден тұратын қоғам кілтті *10* күнде таба алады.

Жоғарыда көрсетілген, *56* бит шифр кілті бар *DES* жеткілікті қауіпсіздікті қамтамасыз етпейді.

DES шифрының әлсіз кілттері

2^{56} мүмкінді кілттер ішінен төртеуі әлсіз деп аталады. *Әлсіз кілттер* – тексеру биттерін (5.1 кестесін қолданып) алып тастау операциядан кейін барлық нөлдерден немесе бірлердің немесе жартысы нөлдерден және жартысы бірлерден тұратын кілттер. Осындай кілттер 5.19 кестесінде көрсетілген.

5.19 кесте

DES шифрының әлсіз кілттері

Тексеру биттерін жою және алмастыру алдындағы кілттер (64 бит)	*Қолданылатын кілттер (56 бит)*
$0101\ 0101\ 0101\ 0101_{16}$	$0000\ 000\ 0000\ 0000_{16}$
$1f1f\ 1f1f\ 1f1f\ 1f1f_{16}$	$0000\ 000\ ffff\ ffff_{16}$
$e0e0\ e0e0\ e0e0\ e0e0_{16}$	$ffff\ ffff\ 0000\ 0000_{16}$
$fefe\ fefe\ fefe\ fefe_{16}$	$ffff\ ffff\ ffff\ ffff_{16}$

Осы әлсіз кілттердің кез келгенінен құрылатын раунд кілттері, - шифр кілті сияқты және сол типті. Мысалы, раундтың сол он алты кілттері бірінші кілтті құрады, ол барлық нөлдерден немесе барлық бірлерден немесе жартылай нөлдерден және бірлерден тұрады.

Бұл кілттерді генерациялау алгоритмі шифр кілтін алдымен екі жартыға бөлу себебімен болады. Блокты араластыру немесе алмастыру блокты өзгертпейді, егер ол барлық нөлдерден, немесе барлық бірлерден, немесе жартылай нөлдер мен бірлерден тұрса.

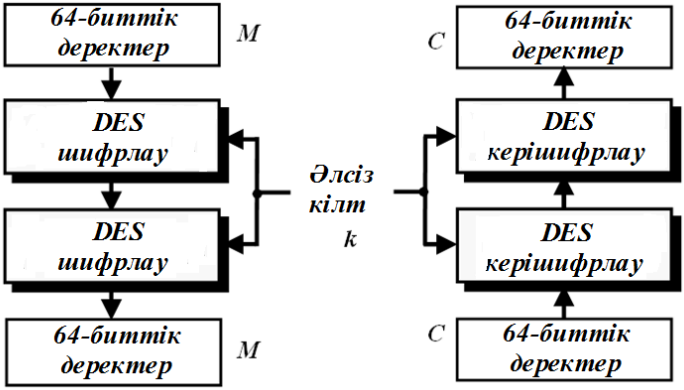
Әлсіз кілттерді қолдану қауіптігі неде? Егер деректер блогын әлсіз кілтпен шифрлап және нәтижесін сол әлсіз кілтпен қайта шифрласақ, онда алғашқы деректер блоғы шығады. Егер екі рет керішифрласақ, онда бір алғашқы деректер блогын құрады. Басқа сөзбен, әр әлсіз кілт өзіне өзі инверсия болады:

$$E_k\,(E_k\,(M)) = M,$$

5.12 суретінде көрсетілгендей.

Әлсіз кілттерден қашу керек, себебі қарсылас оларды жолай ұсталған шифрда тану мүмкін. Егер керішифрлаудың екі кезеңінең кейін нәтиже тұра сондай болса, қарсылас кілтті таптым деп анықтайды.

5.18 мысал. Деректер блогын екі рет шифрлау үшін 5.19 кестеден бірінші әлсіз кілтті қолданып көреміз. Сол кілтпен екі шифрлауды өткізгеннен кейін, нәтижесінде деректер блогын аламыз. Бұл жерде керішифрлау алгоритмі қолданған жоқ, тек екі рет шифрлау орындалды, соған назар аударыңыз.



5.12 сурет - Әлсіз кілтпен екірет деректерді шифрлау және керішифрлау

Кілт: 0101010101010101_{16}.
Алғашқы деректер блогы: 1234567887654321_{16}.
Шифрланған деректер блогы: $814fe938589154f7_{16}$.
Шифр кілті: 0101010101010101_{16}.
Алғашқы деректер блогы: $814fe938589154f7_{16}$.
Шифрланған деректер блогы: 1234567887654321_{16}.

DES шифрының жартылай әлсіз кілттері

Жартылай әлсіз кілттер деп аталатын алты кілттік жұп бар. Осы алты жұп 5.20 кестесінде көрсетілген (тексеру биттерін жою алдында *64* биттік формат).

Жартылай әлсіз кілттер екі түрлі раунд кілттерін құрады және содан кейін оларды сегіз рет қайталайды. Сонымен қатар, әр жұптан құрылған раунд кілттері бірдей, бірақ әртүрлі ретпен келеді.

5.20 кесте

DES жартылай әлсіз кілттері

Жұптағы бірінші кілт	Жұптағы екінші кілт
$01fe\ 01fe\ 01fe\ 01fe_{16}$	$fe01\ fe01\ fe01\ fe01_{16}$
$1fe0\ 1fe0\ 0ef1\ 0ef1_{16}$	$e01f\ e01f\ f10e\ f10e_{16}$
$01e0\ 01e1\ 01f1\ 01f1_{16}$	$e001\ e001\ f101\ f101_{16}$
$1ffe\ 1ffe\ 0efe\ oefe_{16}$	$fe1f\ fe1f\ feoe\ fe0e_{16}$
$011f\ 011f\ 010e\ 010e_{16}$	$1f01\ 1f01\ 0e01\ 0e01_{16}$
$e0fe\ e0fe\ f1fe\ f1fe_{16}$	$fee0\ fee0\ fef1\ fefl_{16}$

Идеяны көрсету үшін, бірінші жұптан раунд кілттерін құрамыз, 5.21 кестесінде көрсетілгендей.

5.21 кесте

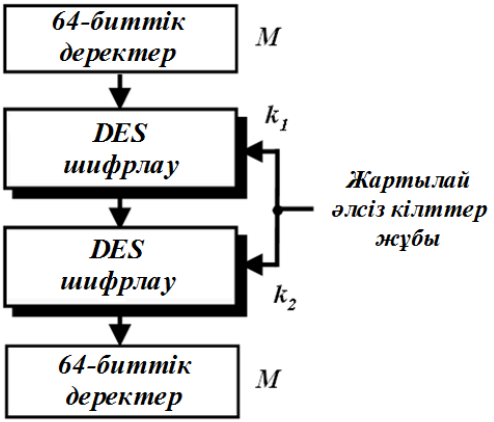
DES шифрының жартылай әлсіз кілттері үшін раундтық кілттер

Раундтар	Жұптағы бірінші кілт үшін раундтық кілттер	Жұптағы екінші кілт үшін раундтық кілттер
1	$9153e54319bd_{16}$	$6eac1abce642_{16}$
2	$6eac1abce642_{16}$	$9153e54319bd_{16}$
3	$6eac1abce642_{16}$	$9153e54319bd_{16}$
4	$6eac1abce642_{16}$	$9153e54319bd_{16}$
5	$6eac1abce642_{16}$	$9153e54319bd_{16}$
6	$6eac1abce642_{16}$	$9153e54319bd_{16}$
7	$6eac1abce642_{16}$	$9153e54319bd_{16}$
8	$6eac1abce642_{16}$	$9153e54319bd_{16}$
9	$9153e54319bd_{16}$	$6eac1abce642_{16}$
10	$9153e54319bd_{16}$	$6eac1abce642_{16}$
11	$9153e54319bd_{16}$	$6eac1abce642_{16}$
12	$9153e54319bd_{16}$	$6eac1abce642_{16}$
13	$9153e54319bd_{16}$	$6eac1abce642_{16}$
14	$9153e54319bd_{16}$	$6eac1abce642_{16}$
15	$9153e54319bd_{16}$	$6eac1abce642_{16}$
16	$6eac1abce642_{16}$	$9153e54319bd_{16}$

5.21 кестені талдау көрсеткендей, әр жартылай әлсіз кілтте сегіз бірдей раунд кілттері бар. Осыдан басқа, бірінші көптікте *1* раунд кілттері екінші көптіктің *16* раунд кілттеріне тең; біріншіде *2* раунд кілттері, екіншіде *15* раунд кілттеріне тең, және ары қарай. Бұл кілттер бір біріне инверсты екенін көрсетеді:

$$E_{k_2}(E_{k_1}(M)) = M,$$

5.13 суретте көрсетілгендей.



5.13 сурет - Жартылай әлсіз кілтпен деректерді екі рет шифрлау

DES шифрының мүмкінді әлсіз кілттері

Сонымен қатар *мүмкінді әлсіз кілттер* деп аталатын *48* кілт бар. *Мүмкінді әлсіз кілт* раундтың тек төрт әртүрлі кілтін құрады; басқа сөзбен, раундтардың он алты кілті төрт топқа бөлінген, және әр топ төрт бірдей раунд кілттерінен тұрады.

5.19 мысал. Әлсіз, жартылай әлсіз немесе мүмкінді әлсіз кілтті кездейсоқ таңдау ықтималдығы қандай?

Шешім. *DES* кілт көптігі 2^{56} тең. Жоғарыда көрсетілген кілттердің жалпы саны – *64: 4 + 12 + 48*. Сондай кілттердің біреуін таңдау ықтималдығы $8{,}8 \cdot 10^{-16}$, яғни аластамалы аз.

DES кілттік толықтырушысы

Кілттер көптігі ішінде (2^{56}) кейбір кілттер кілттегі әр битының инверсиясымен (*0-ді 1-ге* немесе *1-ді 0-ге*) алыну мүмкін. *Кілттік толықтырушы* криптографиялық талдау процессін қарапайымдатады. Қасқой "қатқыл күш" шабуылын орындау үшін мүмкін кілттердің (2^{56}) тек жартысын қолдану мүмкін, себебі:

$$M \oplus M_Д = \text{ffffffffffffffff}_{16}, \quad K \oplus K_Д = \text{ffffffffffffffff}_{16}$$

және

$$C \oplus C_Д = ffffffffffffffff_{16}.$$

Басқа сөзбен, егер алғашқы деректер блогының толықтырушысын кілт толықтырушысымен шифрласа, *шифрланған деректер блогын толықтыру* болады. Қасқой барлық 2^{56} мүмкінді кілттерді тексермеу қажет, ол тек жартысын тексерін нәтижені толықтыра алады.

5.20 мысал. Кілттік толықтыру туралы мәліметтерді тексереміз. Кез келген деректер блогын және кілтті шифрланған деректер блогын табу үшін қолданамыз. Егер кілттік толықтыру және алғашқы деректер блогы бар болса, онда алдынғы шифрланған деректер блогын толықтыруды аламыз (5.22 кестесі).

5.22 кесте

5.20 мысалының нәтижелері

	Кілт түпнұсқасы	Кілттік толықтыру
Кілт	1234123412341234_{16}	$edcbedcbedcbedcb_{16}$
Деректердің алғашқы блогы	$12345678abcdef12_{16}$	$edcba987543210ed_{16}$
Деректердің шифрланған блогы	$e112be1defc7a367_{16}$	$1eed41e210385c98_{16}$

DES шифрының кластерлік кілті

Кластерлік кілт шифрдың екі немесе одан көп әртүрлі кілті бір алғашқы деректер блогынан бірдей шифрланған деректер блогын құратын жағдайларды қарастырады. Әр жартылай әлсіз кілттер жұбы – *кілттік кластер* екені айқын. Бірақ басқа кластерлер табылған жоқ.

5.8. DES КӨПЕСЕЛІ ҚОЛДАНУ

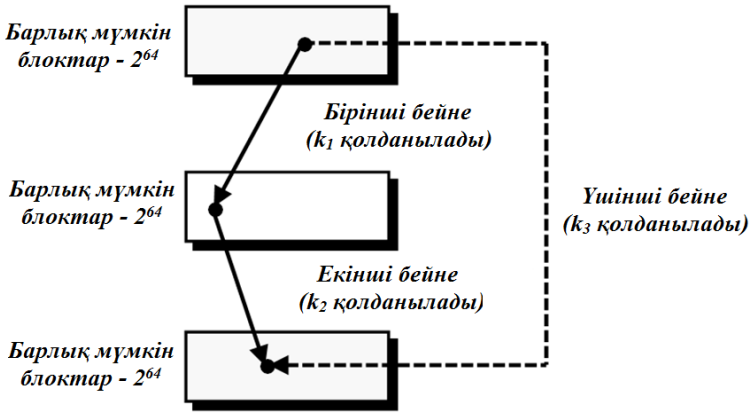
Жоғарыда көрсетілгендей, *DES* негізгі сыны шифр кілтінің ұзындығына бағытталған. Мүмкінді технологиялар және қатарлас процессорлардың мүмкіндіктері «қатқал күш» шабуылын реалды қылады.

Қауіпсіздікті жақсартудың шешімдерінің бірі - *DES* бас тарту және жаңа шифрлау алгоритмін құру. Бірінші шешім - *Advanced*

Encryption Standard (*AES*) шифрлау алгоритмін қолдану. 8 тарауда қарастырылады.

Екінші шешім – *DES* шифрлау алгоритмін әртүрлі кілттермен көпеселі (каскадты) қолдану. Бұл шешімді кейбір уақыт қолданды, үлкен қаржыны және аппараттық құралдарды талап етпеді. Осындай шешімді қарастырайық.

3 тараудан белгілі болғандай, барлық мүмкін кірістерді барлық мүмкін шығыстарға орналастыратын ауыстыру көптік элементтерін бейнелеумен және операциялар жинағымен топ болады. Бұл жағдайда екі тізбекті бейнелеуді қолдану пайдасыз, себебі арқашанда осы екеуінің композициясына (*түйықтық қасиеті*) тең болатын үшіншші бейнелеуді табуға болады. Бұл деген, егер *DES* – топ болса, онда k_3 кілті бар біреселі *DES* соны орындайтынын білдіреді (5.14 суретіне қараңыз).



5.14 сурет – Бейнелеулер композициясы

DES топ емес. Ол келесі екі параметрлерге негізделеді:

1. *DES* мүмкінді кіріс немесе шығыс нөмірі – $N = 2^{64}$. Бұл $N! = (2^{64}!) = 10\ 347\ 380\ 000\ 000\ 000\ 000$ бейнелеулер білдіреді. *DES* топ жасайтын тәсілдерінің бірі – бұл осы барлық бейнелеулерді $log_2 (2^{64}!) \approx 270$ бит кілт көлемімен қолдау. *DES* шифрдың кілт ұзындығы *56* бит болғандықтан, онда бұл қажетті үлкен кілттің кішкене бөлшегі.

2. *DES* топ жасаудың басқа тәсілі – бейнелеулердің көптігі бірінші параметрге тәуелдік мәнінде көптіктің ішкі көптігі болу

керек. Бірінші параметр көмегімен топтан құрылған топтарда кілттік өлшем *56* бит болатыны дәлелденген [20, 21, 22].

Егер *DES* топ болмаса, онда келесі шарт орындалу үшін k_3 кілтін табу азықтималды:

$$E_{K_2}(E_{K_1}(M)) = E_{K_3}(M).$$

Бұл деген кілт ұзындығын үлкейту үшін екіеселі немесе үшеселі *DES* қолдануды мүмкін екенін білдіреді.

5.8.1. Екіеселі DES

Бірінші тәсіл *екіеселі DES* (*Double DES*) қолдануда. Бұл тәсілде шифрлау үшін түзу *DES* шифрдың екі типы және керішифрлау үшін кері шифрдың екі типы қолданылады. Әр тип әртүрлі шифр кілтін қолданады, бұл кілт ұзындығы екіге көбейетінін (*112* бит болатынын) білдіреді. Бірақ екіеселі *DES* ашық деректерді білу шабуылына осал.

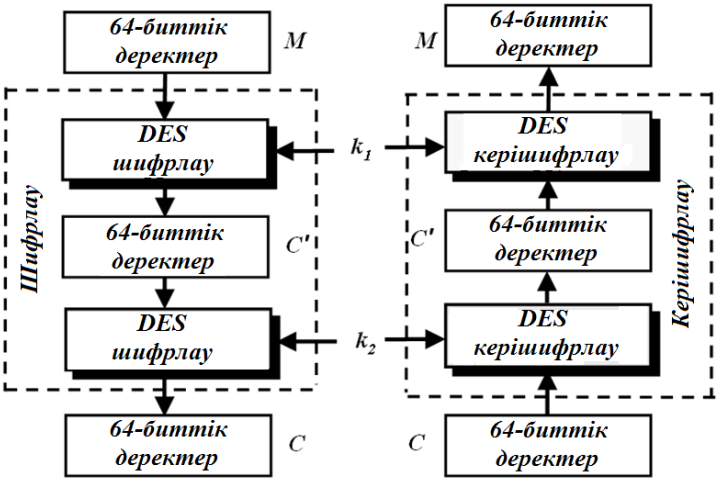
Бірінші көз қарасқа екіеселі *DES* кілтті табу үшін сынау санын екіге көбейтеді 2^{56} (біреселі *DES*) ден 2^{112} (екіеселі *DES*) дейін. Бірақ, *«ортасында кездесу» шабуыл* деп аталатын алғашқы деректерді білу шабуылын қолдану кезінде екіеселі *DES* беріктілікті (сынау бойынша тек 2^{57} дейін) жақсартады, бірақ көпке емес (к 2^{112}) екенін дәлелдеуге болады.

5.15 сурет екіеселі *DES* үшін диаграмманы көрсетеді.

Жіберуші алғашқы деректер блогын *M* шифрланған деректер блогына *C* шифрлау үшін екі шифр кілтін k_1 және k_2 қолданады; қабылдаушы шифрланған деректер блогын *C* және екі шифр кілтін k_1 және k_2 алғашқы деректер блогын *M* қалпына келтіру үшін қолданады.

Орташа нүктеде *C'* – бірінші шифрлау немесе бірінші керішифрлаумен құрылған деректер блогы. Дұрыс жұмысты қамтамасыз ету үшін ол шифрлау және керішифрлау үшін бірдей болу қажет. Басқа сөзбен, екі қатынас бар:

$$C' = E_{K_1}(M) \quad \text{и} \quad C' = E_{K_2}(C).$$

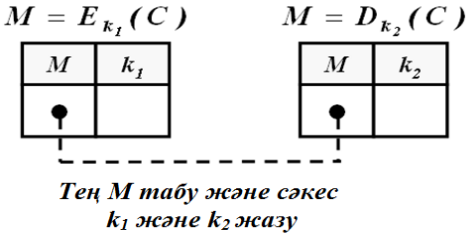


5.15 сурет - Екіеселі *DES* «ортасында кездесу» шабуылы

Қасқой алдынғы *M* және *C* (алғашқы деректерді білу шабуылы) жұбын жолай ұстады дейік. Жоғарыда көрсетілген бірінші қатынасқа негізделіп қасқой k_1 кілттің барлық мүмкінді мәндерін (2^{56}) қолданып *M* шифрлайды және *C'* үшін алынған барлық мәндерді жазып алады. Жоғарыда көрсетілген қатынастарының екіншісіне негізделіп қасқой k_2 кілттің барлық мүмкін мәндерін қолданып *C'* керішифрлайды. *C'* үшін алынған барлық мәндерді жазып алады. Ары қарай қасқой *C'* мәндеріне сәйкес сұрыпталған екі кесте құрады. Ол *C'* мәндерін салыстырып k_1 және k_2 жұптарын табады, олар үшін *C'* мәні екі кестеде бірдей болу керек (5.16 суретте көрсетілгендей).

Кем дегенде бір жұп болу керек екеніне назар аударыңыз, себебі ол екі кілт комбинациясының толық іздеуін өткізеді:

1. Егер тек бір сәйкестік болса. Қасқой екі кілтті (k_1 және k_2) тапты. Егер бірден көп үміткер болса, қасқой келесі қадамға барады.



5.16 сурет - «Ортасында кездесу» шабуылы үшін кестелер

2. Қасқой басқа жолай ұсталған алғашқы деректер блогы және шифрланған деректер блогы жұбын аладыда әр үміткерді кілттер жұбын табу үшін қолданады. Егер кілттер жұбы ретінде ол бірнеше үміткерді тапса, ол бірегей жұпты тапқанша 2 қадамды қайталайды.

Екінші қадамды бірнеше жолай ұсталған *«шифрланған деректер-алғашқы деректер»* жұптарына қолданғаннан кейін кілттер табылғаны дәлелденген [20, 21, 22]. Бұл деген, 2^{112} сынау көмегімен шифр кілттерін іздеудің орнына, қасқой 2^{56} кілтті іздеу сынауларын өткізеді және екі рет тексереді (егер бірінші қадамда бір үміткер табылмаса, көбірек сынаулар қажет болады). Басқа сөзбен, біреселі *DES* екіеселі *DES* дейін жылжу кезінде сынаулар көлемі 2^{56} дан 2^{57}дейін көбейеді (2^{112} дейін емес).

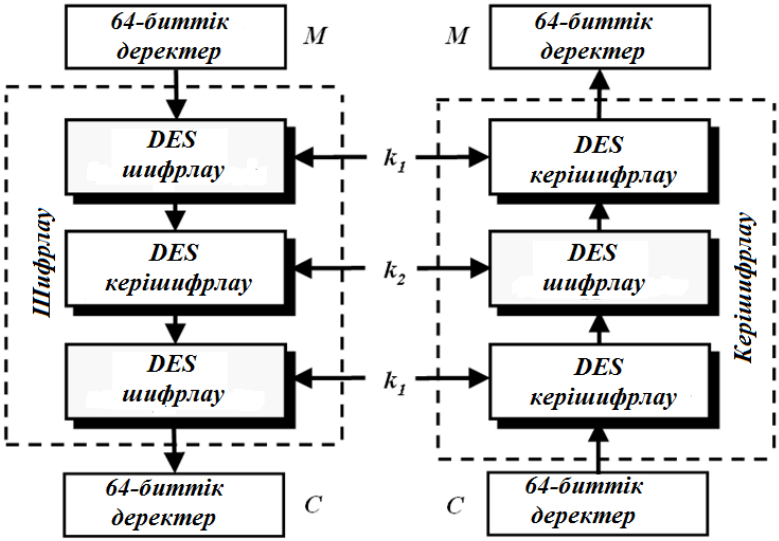
5.8.2. Үшеселі DES

DES қауіпсіздігін жақсарту үшін *үшеселі DES* (*Triple DES*) ұсынылды. Ол шифрлау және керішифрлау үшін *DES* үш каскадын қолданады. Бүгінгі күні үшеселі *DES* екі нұсқасы қолданылады: *шифрдың екі кілті бар үшеселі DES* және *шифрдың үш кілті бар үшеселі DES.*

Шифрдың екі кілті бар үшеселі DES

Шифрдың екі бар *үшеселі DES* шифрдың тек екі кілті бар: k_1 және k_2. Бірінші және үшінші каскадтар k_1 қолданады; екінші каскад k_2 қолданады. Үшеселі *DES-ті DES*-пен үйлесімді жасау үшін, ортанғы каскад шифрлау жағында керішифрлауды (кері шифрды) және керішифрлау жағында шифрлауды (түзу шифр) қолданады. Осы тәсілмен, шифр кілті *k DES* шифрланған хабар, егер $k_1 = k_2 = k$, онда үшеселі *DES* керішифрлану мүмкін. Үшеселі *DES* «алғашқы деректерді білу» шабуылына осал болса да, ол екі еселі *DES* қарағанда беріктірек. Ол өз уақытында банктар үшін қабылданған.

5.17 сурет шифрдың екі кілті бар үшеселі *DES* көрсетеді.



5.17 сурет – Шифрдың екі кілті бар үшеселі *DES*

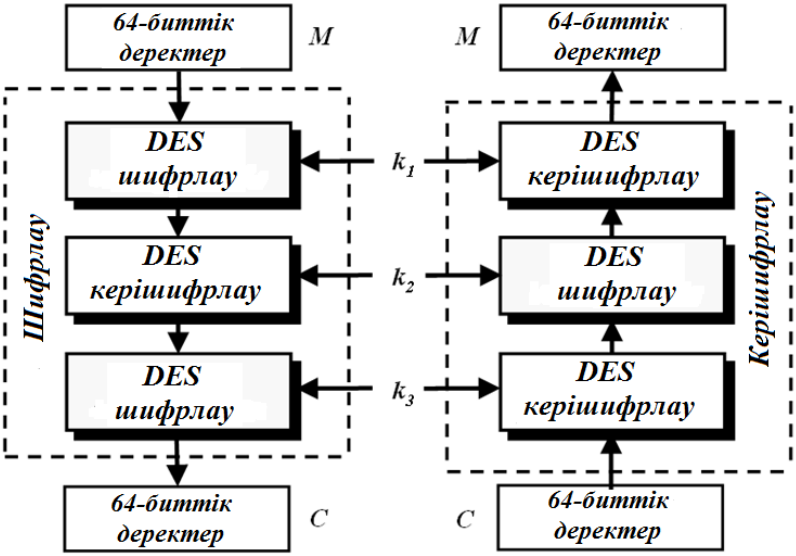
Шифрдың үш кілті бар үшеселі DES

Шифрдың екі кілті бар үшеселі *DES* «алғашқы деректерді білу» шабуылы болу мүмкіндігі кейбір қосымшаларды үш кілті бар үшеселі *DES* қолдануды мәжбүрледі. Алгоритм шифрлау жағында *DES* түзу шифрының үш каскадын және керішифрлау жағында кері шифрдың үш каскадын қолдану мүмкін (5.18 сурет).

Біреселі *DES* үйлесімдік үшін шифрлау жағы *EDE* қолданады, ал керішифрлау жағы – *DED*. *E* (*encryption*) – шифрлау каскады, *D* (*decryption*) – керішифрлау каскады.

Шифрдың кілтінің жалпы ұзындығы бұл әдісте үлкейеді (*112 + + 56=168* бит).

Біреселі *DED* үйлесімдік $k_1 = k$ және қабылдаушы таңдаған бір кез келген кілтке k_2 және k_3 орналастырумен қамтамасыз етіледі. Шифрдың үш кілті бар үшеселі *DES* көптеген қосымшалармен қолданылады, мысалы *PGP* (*Pretty Good Privacy – «жеткілікті» жақсы конфиденциалдық*), себебі ақпараттың қасиетінің жоғары қорғалғандығын – конфиденциалдықты қамтамасыз етеді.



5.18 сурет - Шифрдың үш кілті бар үшеселі *DES*

Үшеселі *DES* қарапайым *DES* жеткілікті кеңінен тараған альтернативасы және ол *ANSI X9.17, ISO 8732* және *ISO 8732* стандарттарында кілттерді басқару кезінде қолданылады. Кейбір криптографиялық талдаушылар оданда сенімді шифрлау үшін үш немесе бес кілті бар *DES* қолдануды ұсынады.

5.9. DES ҚАУІПСІЗДІГІ

DES, үлкен маңызы бар, бірінші блокты шифр ретінде, қауіпсіздікке көп сынаулардан өтті. Қолданған шабуылдар ішінде тек үшеуі қызығушылық тұдырады: *«қатқыл күш»* шабуылы, дифференциалды және сызықты криптографиялық талдаулар.

5.9.1. DES “Қатқыл күш” шабуылы

Шифрдың қысқа кілті бар *DES* әлсіздігі қарастырылды. Шифр кілтінің әлсіздігі басқа қарастырылған кемшіліктермен бірге, *DES* 2^{55} сынаулармен бұзылу мүмкіндігін көрсетті. Бірақ бүгінгі күні көптеген қосымшалар немесе екі шифр кілті бар *Double DES* (кілт

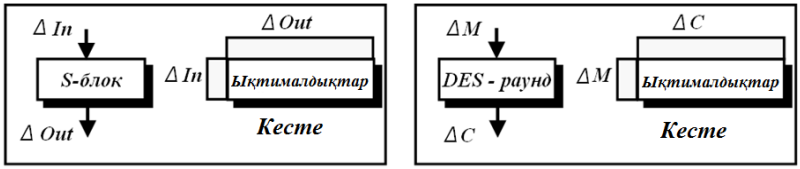
ұзындығы 2^{112}), немесе үш шифр кілті бар *Triple DES* (кілт ұзындығы 2^{168}) қолданады. Осы екі *DES* көпеселі нұсқалары «қатқыл күш» шабуылдарына жақсы беріктілікті көрсетуге мүмкіндік береді.

5.9.2. DES дифференциалды криптографиялық талдау

3 тарауда заманауи блокты шифрлары үшін дифференциалды криптографиялық талдау әдістемесі қарастырылған. Осындай шабуыл түріне *DES* берік емес. Бірақ *DES* құрастырушылары бұл шабуыл түрін білгеніне және *DES* осы шабуыл түріне берік жасау үшін *S*-блоктарды жобалағанына және раундтар санын арнайы таңдағанына көбі көрсетеді. Егер алғашқы деректердің 2^{47} таңдауы немесе 2^{55} белгілі алғашқы деректері бар болса, онда дифференциалды криптографиялық талдауды қолдану арқылы *DES* бұзуға болатыны көрсетілген. «Қатқыл күш» шабуылына қарағанда бұл тиімдірек көрінседе, біреу алғашқы деректердің 2^{47} таңдауын немесе 2^{55} белгілі алғашқы деректерін білу мүмкін емес. Сондықтан, *DES* дифференциалдық криптографиялық талдауға берік деп айтуға болады. Осыған қоса раундтар санын *20* дейін көбейту шабуыл үшін талап етілетін алғашқы деректер санын 2^{64} да көбейетіні көрсетілген. Бұндай көбейу мүмкін емес, себебі *DES* алғашқы деректер блогының саны тек 2^{64}.

DES үшін дифференциалды криптографиялық талдау *Бихаммен* (*Biham*) және *Шамирмен* (*Shamir*) құрылған. Бұл криптографиялық талдауда қасқой алғашқы деректерді таңдаумен шабуылдарға көңіл бөледі. Талдау әртүрлі кіріс сигналдардың шифрлау құрылғысы немесе бағдарламасынан өту айырмашылығын қолданады. «Айырмашылық» түсінігі бұл жерде *xor* операциясы көмегімен екі әртүрлі кіріс хабардың (алғашқы деректердің) бірдей болмауын қарау үшін қолданады. Басқа сөзбен, қасқой $M \oplus M'$ әр раунд өңдеуінде қалай айырылатынын талдайды.

Дифференциалды криптографиялық талдау идеясы кіріс айырмашылығы және шығыс айырмашылығы арасындағы ықтималды қатынастарында негізделеді. Талдауда екі қатынас қызығушылық тұдырады: 5.19 суретінде көрсетілгендей дифференциалды профайл және раунд сипаттамасы.

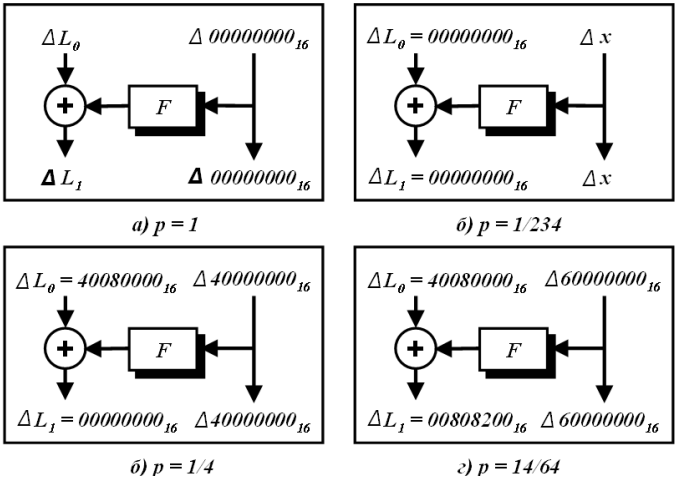


а) б)

5.19 сурет – *DES* диференциалды профайл (а) және раунд сипаттамасы (б)

Дифференциалды профайл (*xor* профайлы) *S*-блоктың кіріс айырмашылығы және шығыс айырмашылығы арасында ықтималды қатынасты көрсетеді. Осындай профайлдар *DES* сегіз *S*-блоктың әрқайсысына құрылу мүмкін.

Раунд сипаттамасы дифференциалды профайлға ұқсас, бірақ тұтас раунд үшін есептеледі. Ол кірістің бір айырмашылығы анықталған шығыс айырмашылығын қандай ықтималдықпен құратынын көрсетеді. Әр раунд үшін сипаттама бірдей екеніне назар аударыңыз, себебі айырмашылық кіретін кез келген қатынас раунд кілтінен тәуелді емес. 5.20 сурет раундтың әртүрлі төрт сипаттамасын көрсетеді.

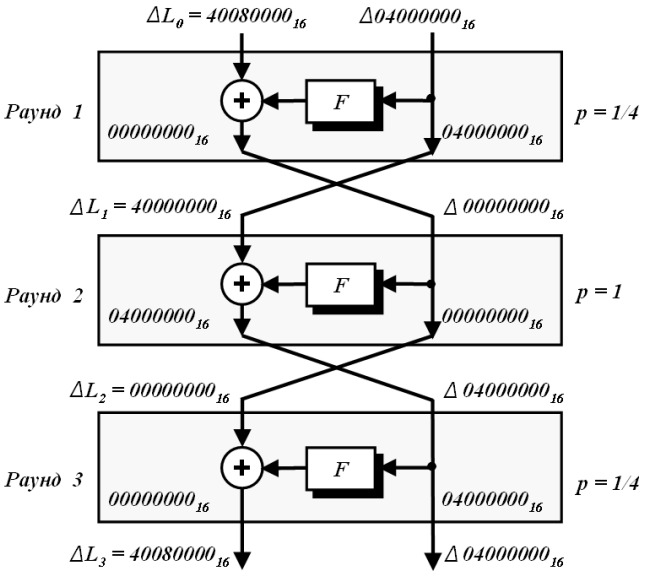


5.20 сурет – Дифференциалды криптографиялық талдау үшін раундтың кейбір сипаттамалары

Раунд үшін көптеген сипаттамалар болса да 5.20 сурет олардың ішінен төртеуін көрсетеді. Әр сипаттамада кіріс айырмашылықтары және шығыс айырмашылықтары оң және сол сакцияларға бөлінген. Әр сол немесе оң айырмашылықтары *32* биттен немесе сегіз он алты саннан тұрады.

Осы барлық сипаттамаларды *DES* раундында «кіріс-шығыс» қатынастыран табатын бағдарламалар таба алады. 5.20 а сурет кіріс айырмашылығы (x, 00000000_{16}) шығыста (x, 00000000_{16}) айырмашылығын *1* ықтималдығымен беретінін көрсетеді. 5.20 б суреті 5.20 а суретінің сипаттамасындай сипаттама көрсетеді, тек сол және оң кіріс пен шығыстары орнымен ауысты; ықтималдық қатты өзгереді. 5.20 в сурет кіріс айырмашылығы (40080000_{16}, 04000000_{16}) шығыс айырмашылығын (00000000_{16}, 0400000_{16}) *1/4* ықтималдығымен беретінін көрсетеді.

Және, 5.20 г суреті кіріс айырмашылығы (00000000_{16}, 60000000_{16}) шығыс айырмашылығын (00808200_{16}, 6000000_{16}) *14/64* ықтималдығымен беретінін көрсетеді.

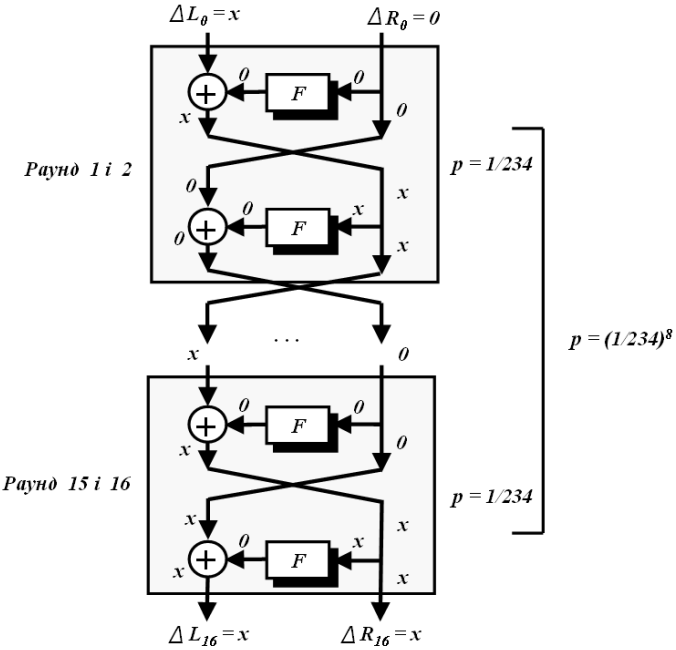


5.21 сурет – Дифференциалды криптографиялық талдау үшін үшраундтық сипаттама

Бірраундтық сипаттамаларды құрған және сақтағаннаң кейін криптографиялық талдаушы раундың көптік сипаттамаларын құру үшін раундтың әртүрлі санын араластыра алады. 5.21 сурет үшеселі *DES* жағдайын көрсетеді.

5.21 суретте үш араластырғыш және тек екі ауыстыру құрылғысы қолданылады, себебі соңғы раунд үшін ешқандай ауыстыру құрылғысы қажет емес. Бірінші және үшінші раунд араластырғыштарында көрсетілген сипаттамалар 5.20 б суретте көрсетілгендермен бірдей. Екінші раундтағы араластырғыш сипаттамасы – 5.20 а суретіндегідей. Осы нақты жағдайда, өте қызықты нәрсе, нүктелер, «кіріс-шығыс» айырмашылықтары – тұра сондай ($\Delta L_3 = \Delta L_0$ және $\Delta R_3 = \Delta R_0$).

Он алты раунды бар шифр үшін көп әртүрлі сипаттамаларды компиляциялауға болады. 5.22 сурет мысалды көрсетеді.



5.22 сурет – Дифференциалды криптографиялық талдау үшін он алтыраундтық сипаттама

5.22 суретінде *DES* екі раунды бар сегіз секциядан тұрады. Әр секция 5.20 а және 5.20 б суретіндегі сипаттамаларды қолданады.

Егер соңғы раундтарда ауыстыру құрылғысы жоқ болса, онда кіріс (*x, 0*) (*1/234*)^{8} ықтималдықпен шығысты (*0, x*) құратыны түсінікті.

Мысал үшін, он алты раунды бар *DES* шабуылды құру үшін қасқой 5.21 суреті бойынша сипаттаманы қолданады деп есептейік. Қасқой көп алғашқы деректерді (*x, 0*) түрде шифрлау үшін, кейбір тәсілмен жіберушіні итермелейді, бұл жерде сол бөлігі – *x* (әртүрлі мәндер) және оң жағы – *0.* Содан кейін қасқой жіберушіден алған барлық шифрланған деректерді (*0, x*) түрде сақтайды. Бұл жерде *0* деген 00000000_{16} екеніне назар аударыңыз.

Дифференциалды криптографиялық талдауда қасқойдың соңғы мақсаты шифр кілтін табуда. Ол үшін басынан соңынадейін әр раундтың кілттерін (k_{16} … k_1)табу қажет.

Егер қасқойда жеткілікті көп алғашқы деректер/шифрланған деректер жұптары (әрқайсысы *x* әртүрлі мәндерімен) бар болса, онда ол соңғы раундта қатынастарды қолдану $0 = F(k_{16},x)$ және k_{16} кейбір биттерді табу мүмкін. Бұны ең ықтималды мәндерді таңдап жасауға болады.

Басқа раундтар үшін кілттерді басқа сипаттамаларды немесе «қатқыл күш» шабуылын қолданып табуға болады.

16 раунды бар *DES* шабуылды жүзеге асыру үшін алғашқы деректер/шифрланған деректер жұптарының саны 2^{47} болу қажет екені белгілі. Реалды өмір жағдайда осындай көп таңдауларды табу өте қыйын. Бұл, *DES* осы шабуыл түріне осал емес екенін білдіреді.

5.9.3. DES сызықты криптографиялық талдау

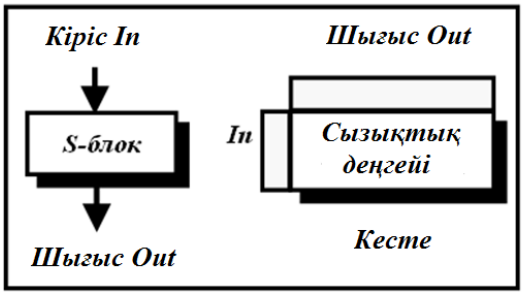
Заманауи блокты шифрлау үшін сызықты криптографиялық талдау әдістемесі 3 тарауда қарастырылғын. Сызықты криптографиялық талдау – дифференциалды криптографиялық талдауға қарағанда жаңа әдістеме. *DES* дифференциалдық криптографиялық талдауға қарағанда сызықты криптографиялық талдауды қолдануға әлсіздеу, себебі келесіде болу ықтимал: бұл шабуыл түрлері *DES* жобалаушыларына белгісіз болды және *S*-блоктар сызықты криптографиялық талдауға соншалықты берік емес. *DES* белгілі алғашқы деректердің 2^{43} жұбын қолданумен бұзылу мүмкін екені көрсетілген. Бірақ тәжірибелік түрде осындай жұптар санын жолай ұстау аз ықтималды.

DES үшін сызықты криптографиялық талдау *Матцуимен* құрылған [7, 20]. Бұл – алғашқы деректерді білу шабуылы. Талдау

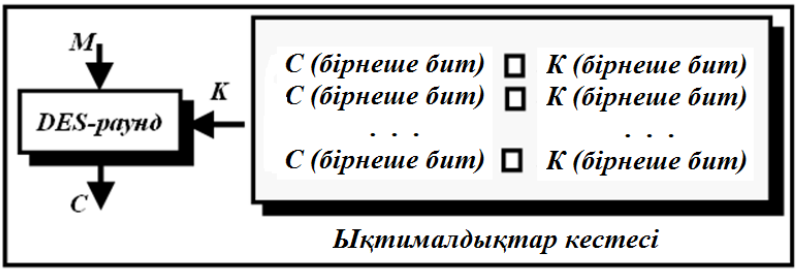
нақты биттер жинағын шифрлау құрылғысы арқылы тарауды қолданады.

Сызықты криптографиялық талдау сызықтық қатынастарына негізделген. Криптографиялық талдаудың бұл типінде екі қатынастар түрі қызығушылықты тұдырады: 5.23 суретінде көрсетілгендей, сызықты профайлдар және раунд сипаттамалары.

Сызықты профайл *S*-блоктың кіріс және шығыс арасында сызықтық деңгейін көрсетеді. *S*-блокта шығыстың әр биты – барлық кіріс биттердің функциясы. *S*-блокта тілекті қасиеті орындалады, егер шығыстың әр биты – барлық кіріс биттерінің сызықты емес функциясы болса.



а) сызықты профайл



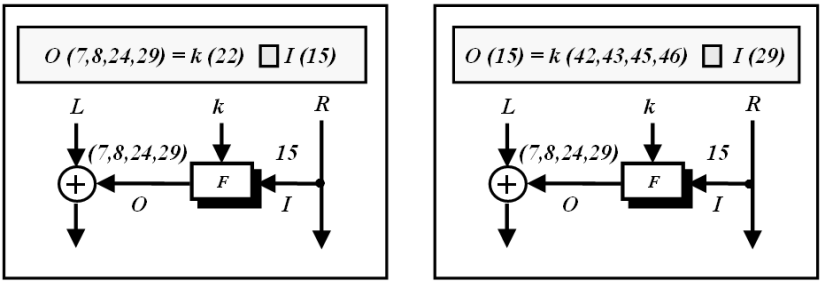
б) раунд сипаттамалары

5.23 сурет - *DES* сызықты профайл және раунд сипаттамасы

Өкінішке орай, *DES* бұл идеалды жағдай жоқ; шығыстың кейбір биттері – кейбір кіріс биттер комбинацияларының сызықты функциясы. Басқа сөзбен, кейбір «кіріс-шығыс» биттер комбинациялары бір бірі арасында сызықты функцияны қолдану арқылы бейнелену мүмкін. Сызықты профайл кіріс және шығыс

арасында сызықтық (немесе сызықтық емес) деңгейін көрсетеді. Криптографиялық талдау *S*-блоктың әрқайсысы үшін кесте құру мүмкін, жалпы сегіз әртүрлі кестені. Оның ішінде бірінші баған алты биттен мүмкінді кіріс комбинацияларын көрсетеді 00_{16} ден $3F_{16}$ дейін. Бірінші жол төрт биттен мүмкінді шығыс комбинацияларын көрсетеді 00_{16} ден F_{16} дейін. Кірістер берілген жобаның сызықтық (немесе сызықтық емес) деңгейін көрсетеді. Сызықтық деңгейін өлшеу қалай орындалатынын көрсетпейміз, бірақ сызықтықтан жоғары деңгейі бар кірістер криптографиялық талдау үшін қызықты.

Сызықты криптографиялық талдауда раунд сипаттамасы кіріс биттерінің, раунд кілттер биттерінің және шығыс биттерінің комбинацияларын сызықтық қатынасты анықтау үшін көрсетеді. 5.24 суреті раундтың екі әртүрлі сипаттамаларын көрсетеді.



а) p = 52/64 *б) p = 42/64*

5.24 сурет – Сызықты криптографиялық талдау үшін раундтың кейбір сипаттамалары

Әр жағдай үшін қолданатын белгілер жүйесі *2* модулі бойынша қосылатын биттерді анықтайды. Мысалы, *O*(*7, 8, 24, 29*) функциядан шығатын *7, 8, 24* және *29* биттерінің *xor* операциясын белгілейді; *k(22)* раунд кілттіндегі *22*-ші битті белгілейді; *I(15)* функцияға кіретін *15*-ші битті белгілейді.

Төменде 5.24 а және 5.24 б суреттерінің бөліктері үшін жеке биттерді қолданатын қатынастары көрсетілген:

- *а* бөлігі: $O(7) \oplus O(8) \oplus O(24) \oplus O(29) = I(15) \oplus k(22)$;
- б бөлігі: $F(15) = I(29) \oplus k(42) \oplus k(43) \oplus k(45) \oplus k(46)$.

Бірраундтық сипаттамаларды құру және сақтағаннаң кейін талдаушы раундтың көптік сипаттамасын құру үшін әртүрлі раундтарды аралатыру мүмкін. 5.25 сурет үшраундтық *DES*

жағдайын көрсетеді, бұл жерде *1* және *3* раундтар 5.24 а суретінде көрсетілгендей бір сипаттаманы қолданады, ал *2* раундта кез елген сипаттама қолданылған.

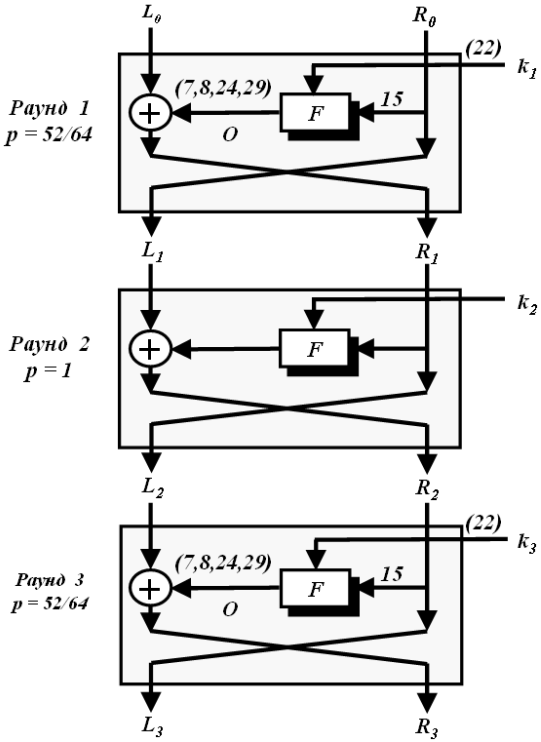
Сызықты криптографиялық талдаудың мақсаты «алғашқы деректер – шифрланған деректер» жұбының кейбір биттер арасындағы сызықтық қатынасты және кілтті табуда. Осындай қатынасты 5.25 суретте көрсетілген үш раунды бар *DES* үшін орнатуға болама екенін көреміз.

Раунд *1*: $R_1(7, 8, 24, 29) = L_0(7, 8, 24, 29) \oplus R_0(15) \oplus k_1(22)$.

Раунд *3*: $L_3(7, 8, 24, 29) = L_2(7, 8,24, 29) \oplus R_2(15) \oplus k_3(22)$.

Бірақ L_2 сәйкес R_1, R_2 сәйкес және R_3. L_2-ні R_1-ге және R_2-ні R_3-ке ауыстырғаннаң кейін екінші қатынаста аламыз:

$$L_3(7, 8, 24, 29) = R_1(7, 8, 24, 29) \oplus R_3(15) \oplus k_3(22).$$



5.25 сурет – Сызықты криптографиялық талдау үшін үшраундтық сипаттамалар

R_1 оның *1* раундтағы эквивалентті мәніне ауыстыруға болады, нәтижесінде келесіні аламыз:

$$L_3(7, 8, 24, 29) = L_0(7, 8, 24, 29) \oplus R_0(15) \oplus$$
$$\oplus k_1(22) \oplus R_3(15) \oplus k_3(22).$$

Бұл қатынастар кіріс және шығыс биттер арасында түрлендіруден кейін үш раундтан тұратын барлық жүйе үшін:

$$L_3(7, 8, 24, 29) \oplus R_3(15) = L_0(7, 8, 24, 29) \oplus$$
$$\oplus R_0(15) \oplus k_1(22) \oplus k_3(22).$$

Басқа сөзбен, келесіні аламыз:

$$C(7, 8, 15, 24, 29) = M(7, 8, 15, 24, 29) \oplus k_1(22) \oplus k_3(22).$$

Бір қызық сұрақ: үшраундты (немесе *n*-раундты) *DES* ықтималдығын қалай табуға болады. *Матцуи* (*Matsui*) осы жағдайдағы ықтималдықты көрсетті [7, 32]

$$P = 0{,}5 + (2n-1)\prod_i (p_i - 0{,}5),$$

бұл жерде n – раундтар саны; p_i – раундтың әр сипаттамасының ықтималдығы; P – толық ықтималдық.

Мысалы, 5.24 суретіндегі үшраундтық талдау үшін толық ықтималдық

$$P = 0{,}5 + (2^3 - 1)\,[(52/64 - 0{,}5) \times (1 - 0{,}5) \times (52/64 - 0{,}5)] = 0{,}695.$$

Он алты раундтық сипаттама да компиляциялану мүмкін, кіріс биттерінің кейбір биттер, шифрланған деректердің кейбір биттер және раунд кілттерінің кейбір биттер арасындағы сызықты қатынастарды қамтамасыз ету үшін:

$$C \text{ (кейбір биттер)} = M \text{ (кейбір биттер)} \oplus$$
$$\oplus k_1 \text{ (кейбір биттер)} \oplus \ldots \oplus k \text{ (кейбір биттер)}.$$

Ашық деректердің кейбір биттер, шифрланған деректер биттері және раунд кілттері биттері арасындағы көптеген қатынастарды табу және сақтау дан кейін қасқой раунд кілттерінде биттерді табу

үшін кейбір «алғашқы деректер-шифрланған деректер» (алғашқы деректерді білу шабуылы) жұптарына жүгіну және сақталған сипаттамалардың сәйкес биттерін қолдану мүмкін.

16-раундтық *DES* шабуыл құру үшін 2^{43} белгілі «алғашқы деректер-шифрланған деректер» жұптары болу қажет екені белгілі. Сызықты криптографиялық талдау дифференциалды криптографиялық талдауға қарағанда ықтималдырақ көрінеді екі себебпен: бірінші – оның қадам саны азырақ; екінші – ол алғашқы деректерді таңдау шабуылына қарағанда алғашқы деректерді білу шабуылына қарапайымдау. Бірақ осындай да шабуыл *DES* жұмыс істейтіндерге қауіп тұдырудан алыс.

Бақылау сұрақтары және тапсырмалары

1. *DES* деректер блогының, шифр кілтінің және раунд кілтінің көлемі қандай?
2. *DES* раундтар саны неше?
3. Шифрлау және керішифрлаудың бірінші тәсілінде қанша араластырғыш және ауыстыру құрылғысы қолданылады? Екінші тәсілінде қаншасы қолданылады?
4. *DES* шифр алгоритмінде қанша алмастыру қолданылады?
5. *DES* қанша *xor* операциясы қолданылады?
6. *DES* кеңейтумен алмастыруды жүзеге асыратын операция неге қажет?
7. Неге *DES* раунд кілттерінің генераторы тексеру биттерін алып тастауды талап етеді?
8. *DES* шифрының әлсіз, жартылай әлсіз және мүмкінді әлсіз кілт арасында қандай айырмашылық бар?
9. Екіеселі *DES* деген не? Екіеселі *DES* қандай шабуылы оны пайдасыз етті?
10. Үшеселі *DES* деген не? Екі шифр кілті бар үшеселі *DES* деген не? Үш шифр кілті бар үшеселі *DES* деген не?
11. *DES* кіріс деректер тізбегінің мәні: $1234567890abcdef_{16}$. *IP*-алмастыру блогы шығысындағы тізбек мәнін анықтаңыз.
12. *DES*-та R_0 тізбегінің мәні: $R_0 = f0aae8a5_{16}$. Алмастыру және кеңейтудің *E*-блок шығысында тізбек мәнін анықтау.
13. *DES*-та R_1 тізбегінің мәні: $R_1 = 116ba133_{16}$. Алмастыру және кеңейтудің *E*-блок шығысында тізбек мәнін анықтау.

14. *DES*-та алмастыру және кеңейтудің *E*-блок шығысында тізбек мәні: $R_1^E = 8a2b57d029a6_{16}$. Раунд кілтінің мәні: $k_1 = 4568581abcce_{16}$. *Фейстелдің* араластырғыш функциясының *xor* операциясы шығысында тізбек мәнін анықтау.

15. *DES*-та алмастыру және кеңейтудің *E*-блок шығысында тізбек мәні: $R_{12}^E = 9f3ca2bffea6_{16}$. Раунд кілтінің мәні: $k_{12} = c2c1e96a4bf3_{16}$. *Фейстелдің* араластырғыш функциясының *xor* операциясы шығысында тізбек мәнін анықтау.

16. *DES*-та араластырғыштың *xor* операциясы шығысында тізбек мәні: $R_{12}^{\oplus} = 5dfd4bd5b555_{16}$. Ауыстыру және қысу блоктар шығысындағы мәндерді анықтау: а) S_2; б) S_3; в) S_4; г) S_5.

17. *DES*-та араластырғыштың *xor* операциясы шығысында тізбек мәні: $R_{13}^{\oplus} = e8848768_{16}$. Араластырғыштың ауыстыру және қысу *S*-блоктар шығысында тізбек мәнін анықтау.

18. *DES*-та араластырғыштың ауыстыру және қысу *S*-блоктар шығысында тізбек мәні: $R_2^S = d5b80915_{16}$. Араластырғыштың түзу алмастырудың *P*-блок шығысындағы тізбек мәнін анықтау.

19. *DES*-та араластырғыштың түзу алмастырудың *P*-блок шығысындағы тізбек мәні: $R_3^P = 07d0a03c_{16}$. Тізбек мәні: $L_3 = 4f73c3b3_{16}$. Араластырғыштың *xor* функциясының шығысында тізбек мәнін анықтау.

20. *DES*-та шифрлаудың бесінші раунд кірісінде тізбек мәндері: $L_4 = 07eb4845_{16}$, $R_4 = 48a3638f_{16}$. Араластырғыштың түзу алмастырудың *P*-блок шығысындағы тізбек мәні: $R_4^P = 46932b6e_{16}$. *DES* шифрлаудың бесінші раунд шығысындағы *L* және *R* тізбектерінің мәндерін анықтау.

21. *DES*-та шифрлаудың алтыншы раунд кірісінде тізбек мәндері: $L_5 = 48a3638f_{16}$, $R_5 = 4178632b_{16}$. Раундтық кілт мәні $k_6 = c1948e87475e_{16}$. *DES* шифрлаудың алтыншы раунд шығысындағы L_6 және R_6 тізбектерінің мәндерін анықтау.

22. *DES* шифрының кілт тізбегінің мәні: $K = abba08192637cddc_{16}$. C_0 және D_0 тізбектерінің мәндерін анықтау.

23. *DES* шифрының C_0 және D_0 тізбектерінің мәндері: $C_0 = c3c033a_{16}$ және $D_0 = 33f0cfa_{16}$. Тізбектер мәндерін анықтау: а) C_2 және D_2; б) C_3 және D_3; в) C_7 және D_7; г) C_{12} және D_{12}.

24. *DES* шифрының C_0 және D_0 тізбектерінің мәндері: $C_0 = c3c033a_{16}$ және $D_0 = 33f0cfa_{16}$. Раундтық кілттер k_2 мәндерін анықтау.

25. *DES* шифрының C_0 және D_0 тізбектерінің мәндері: $C_0 = c3c033a_{16}$ және $D_0 = 33f0cfa_{16}$. Кілттер мәндерін анықтау:

а) төртінші раундтық кілтінің;

б) тоғызыншы раундтық кілтінің;

в) он үшінші раундтық кілтінің;

г) он алтыншы раундтық кілтінің;

26. *DES* шифрлаудың он алтыншы раунд шығысында тізбек мәндері: $L_{16} = c95320e2_{16}$, $R_{16} = 9b7b3602_{16}$. Шифрланған деректер мәндерін анықтау.

6 тарау

ДЕРЕКТЕРДІ БЛОКТЫ СИММЕТРИЯЛЫҚ ШИФРЛАУДЫҢ КРИПТОГРАФИЯЛЫҚ АЛГОРИТМДЕРІН ОРЫНДАУ РЕЖИМДЕРІ

DES аз уақыттан кейін *АҚШ* тағы бір федералды стандарт қабылданды, ол деректерді шифрлау үшін *DES* алгоритмін қолданудың (орындаудың) төрт тәсілін ұсынды. Сол уақыттан бұл режимдер жалпықабылдан болып кез келген блокты шифрлармен қолданылады. Кешірек тағы бір режим қосылды.

Кез келген симметриялық блокты шифрлау алгоритмі үшін оларды орындаудың бес режимі анықталған:

1. *Электронды кодты кітап режимі (Electronic Code Book - ECB)*. Бұл орындау режимінде шифрланбаған деректердің әр блогы шифрлаудың бір кілтін қолданып басқа блоктарға тәуелсіз шифрланады. Типтік қолданулар – бірлік мәндерді қауіпсіз жіберу (мысалы, криптографиялық кілтті).

2. *Шифрланған деректердің блоктарын жалғау режимі (Cipher Block Chaining - CBC)*. Бұл орындау режимінде криптографиялық алгоритмнің кірісі шифрланбаған деректердің келесі блогына және шифрланған деректердің алдынғы блогына *xor* операциясын қолдану нәтижесі. Типтік қолданулар – жалпы блокты-бағытталған жіберу, аутентификация.

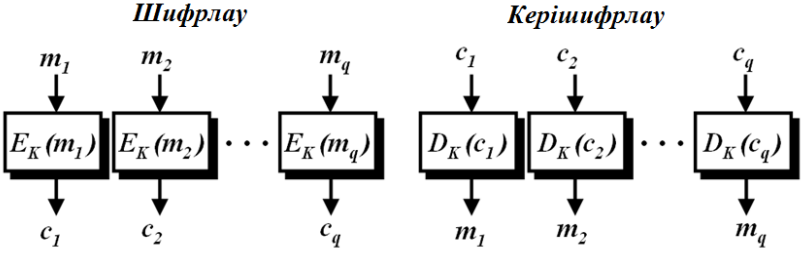
3. *Шифрланған деректер бойынша кері байланыс режимі (Cipher Feedback - CFB)*. Бұл орындау режимінде алгоритмді әр шақырған сайын кіріс мәннің *l* биты өңделеді. Алдынғы шифрланған блок алгоритмге кіріс ретінде қолданылады; алгоритм шығысының *l* битіне және *l* биттен тұратын келесі шифрланбаған блокқа *xor* операциясы қолданылады, оның нәтижесі *l* биттен тұратын келесі шифрланған блок. Типтік қолданулар – ағынғабағытталған жіберу, аутентификация.

4. *Шығыс бойынша кері байланыс режимі (Output Feed back - OFB)*. Бұл орындау режимі *CFB* орындау режиміне ұқсас, келесіден басқа: келесі блокты шифрлау кезінде алдынғы блокты шифрлау нәтижесі алгоритм кірісіне беріледі; тек осыдан кейін шифрланбаған деректердің кезекті *l* биттерімен *xor* операциясы орындалады. Типтік қолданулар – шуы бар арнамен ағынғабағытталған жіберу (мысалы, спутниктік байланыс).

5. *Санағыш режимі* (*Counter Mode - CTR*). Бұл орындау режимі *OFB* режиміне өте ұқсас, бірақ кілттік ағын мәндерін генерациялау үшін инициализация векторының бірегей кездейсоқ мәндерін қолданудың орнына бұл режим санағышты қолданады. Оның мәні шифрланатын әр ашық деректер блогына қосылады. Санағыштың бірегей мәні әр блок кілттік ағынның бірегей мәнімен біріктірілетінін кепілдейді.

6.1. “ЭЛЕКТРОНДЫ КОДТЫ КІТАП” ОРЫНДАУ РЕЖИМІ

Электронды кодты кітап режимі ECB блокты шифрларды қолданудың стандартты тәсілдер арасында ең қарапайым (6.1 сурет).



6.1 сурет – «Электронды кодты кітап» орындау режимінің құрылымы

Шифрланатын деректер M, берілген ұзындығы бар mi блоктарға бөлінеді. Соңғысын қажеттілік болса, толықтырады. Блоктын әркайсысы бір шифрлау кілтін қолданып тәуелсіз шифрланады

$$c_i = E_K(m_i), \qquad (6.1)$$

бұл жерде сі – шифрланған деректер блоктары; Е – шифрлау функциясы; К – шифр кілті.

Керішифрлау процессі – алдыңғы операцияның қайтымды операциясы (6.1)

$$m_i = D_K(c_i), \qquad (6.2)$$

бұл жерде D – керішифрлау функциясы.

Берілген режимнің негізгі артықшылығы – жүзеге асыру қарапайымдылығы. Бірақ ECB режимімен бір қатар проблема байланысқан.

Бірінші проблемада mi = mj болған кезде ci = cj бірдей блоктары болады, яғни кірістегі бірдей блоктар шығыста бірдей блоктарға пайда болуына алып келеді. Бұл шынында проблема, себебі хабардың шаблонды басы мен соңы бірдей болады, бұл криптографиялық талдаушыға хабар мазмұны туралы кейбір ақпаратты береді.

Екінші проблема келесімен байланысты: егер хабардан кейбір блокты алып тастаса, ол ешқандай із қолтырмайды, сонда шабуыл жасайтын адам оны пайдаланып жіберілетін ақпаратты бұрмалау мүмкін.

Үшінші екіншіге ұқсас, бірақ басқа хабарлардан блоктарды ендірумен байланысты.

Бұл проблемаларды көрсету үшін, шифрдың қарапайым үлгісін алайық, оның ішінде блок сөзге сәйкес келеді және ашық мәтін келесі:

"*Плати Алисе сто фунтов, не плати Бобу двести фунтов*"

шифрланған түрі келесі:

"*У кошки четыре ноги, а у человека две ноги*".

Енді қабылдаушыны Алисаға екі жүз фунт төлетуге болады, бір жүздің орнына, оған келесі хабарды жіберіп

"*У кошки две ноги*",

бұл хабарды бірінші хабарда екінші хабардағы бір блогын ауыстырумен алынған. Осыған қоса, Алисаға төлемдерді тоқтатуға болады, ол үшін шифрланған хабардың A блогын бірінші хабардың басына қою қажет. A – ашық хабардың не сөзінің шифрланған нұсқасы. Шифрланған хабарды алушыны Бобқа екі жүз фунт төлетуге болады, егер шифрланған хабардан A блогын алып тастаса.

Бұндай шабуылдарға қарсы тұруға болады, ашық деректердің бірнеше блоктарының бақылау соммаларын косып немесе

шифрланған деректердің әр блогына “контексті идентификатор” қосылатын режимді қолданып.

ECB режимінде шифрлау кезінде хабарды жіберу кезеңінде пайда болған шифрланған деректердің бір битіндегі қате, қате жіберілген бүкіл блокқа тарайды және ол дұрыс керішифрланбайды, бірақ бұл басқа керішифрланған блоктарға әсер етпейді. Бірақ, егер шифрланған деректер биті жоғалып қалса немесе біреуі қосылса, онда ары қарай барлық шифрланған деректер дұрыс емес керішифрланады, егер блок шекараларын кейбір құрылымды қолданып белгіленбесе.

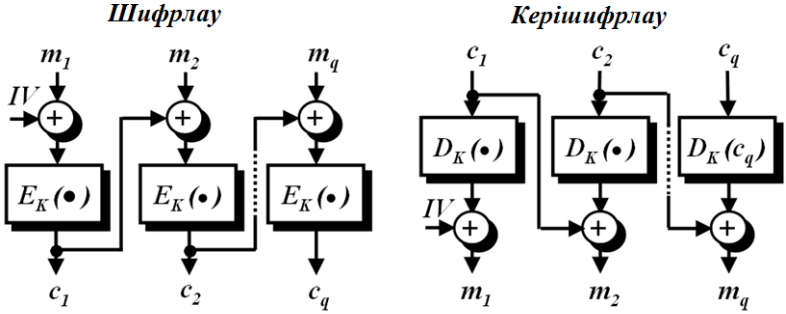
Хабардың көбісі шифрлау үшін 64-биттік (немесе, кез келген басқа ұзындықты, мысалы 128 бит) блоктарға нақты бөлінбейді, соңында әдетте қысқартылған блокты қолданады. ECB режимі 64-биттік блоктарды қолдануды талап етеді. Бұл проблеманы шешу тәсілі деректер блогын берілген ұзындыққа дейін толықтыру.

Соңғы блок кейбір тұрақты үлгімен толық блокты алғанша толықтырылады: нөлдермен, бірлермен, кезектелген нөл мен бірлермен. Керішифрлаудан кейін толықтырған деректерді алып тастау үшін соңғы блоктың соңғы байтына байттар (биттер) саны жазылады. Мысалы, блок ұзындығы – 64 бит болсын және соңғы блок үш байттан тұрсын (24 бит). Блокты 64 битке дейін толықтыру үшін бес байт қажет. Сондықтан төрт байтты толықтыру қажет (мысалы, нөлдермен) және бір байтқа толықтырудың ұзындығы жазылады – 5. Керішифрлаудан кейін соңғы блоктың соңғы байтын алып толықтыру ұзындығын анықтау қажет, содан кейін керішифрланған деректердің соңғы блогынан бес байтты жою қажет. Бұл әдіс дұрыс жұмыс істеу үшін, әр хабар толықтырылу қажет, оның ұзындығы тұтас блоктар санынан тұратын болса да. Бұл жағдайда толықтырудың тұратын бір толық блокты қосу қажет болады, ол блокта жеті байт толықтырудан тұрады және сегізінші байтта толықтыру ұзындығы болады.

6.2. “ШИФРЛАНҒАН ДЕРЕКТЕРДІҢ БЛОКТАРЫН ЖАЛҒАУ” ОРЫНДАУ РЕЖИМІ

ECB режимін қолдану кезінде болатын проблемаларды айналып өту жолының бірі шифрланған деректер блоктарын «жалғауда», яғни деректердің шифрланған блоктарының әр қайсысына контексті идентификаторды қосуда. Оны жасаудың ең қарапайым тәсілі –

«*шифрланған деректердің блоктарын жалғау*» немесе CBC режимін қолдану (6.2 сурет).



6.2 сурет - «Шифрланған деректердің блоктарын жалғау» орындау режимінің құрылымы

Бұл режимде шифрлау үшін алғашқы деректер, әдеттегідей, блоктар сериясына бөлінеді

$$m_1, m_2, m_3, \ldots, m_q,$$

бұл жерде q – шифрлау үшін алғашқы деректердің блоктар саны (толықтырумен бірге).

Алдынғы режимдегідей, алғашқы деректер ұзындығы блок ұзындығына еселі болу үшін соңғы блокқа толықтыру қажет болу мүмкін.

Шифрлау келесі өрнектерге сәйкес орындалады:

$$c_1 = E_K(m_1 \oplus IV),\ c_i = E_K(m_i \oplus c_{i-1}),\ i=2,3,\ldots,q. \qquad (6.3)$$

Бірінші блокты шифрлауға алғашқы шама IV (initialization vector –инициализация векторы) қатысады, оны шифрлау функциясын беруге жатқызуға болады. Шифрлауға IV шаманы алғашқы деректердің бірдей бөліктері шифрланған кезде әртүрлі болу үшін қосады.

IV шамасы керішифрлауда да қолданылады. Бұл процесс келесі түрде көрсетіледі:

$$m_1 = D_K(c_1) \oplus IV,\ m_i = D_K(c_i) \oplus c_{i-1},\ i=2,3,\ldots,q. \qquad (6.4)$$

Шифрланған деректердің соңғы 64-биттік блогы құпия кілт K, алғашқы мән IV және ұзындыққа тәуелсіз алғашқы деректердің әр битының функциясы болатыны айқын. Бұл шифрланған деректер блогы хабарды аутентификациялау коды (message authentication code – MAC) деп аталады.

MAC құпия кілті K және алғашқы мән IV бар жіберушімен деректерді керішифрлағаннаң кейін оңай тексерілу мүмкін. Ол үшін жіберуші орындаған процедураны қайталау қажет. Бөтен адам қабылдаушы шынайы деп қабылдайтын MAC-ты, оны жалған хабарға қосу үшін немесе MAC шынайы хабардан бөліп алып өзгертілген немесе жалған хабармен қолдану үшін, генерация жасай алмайды.

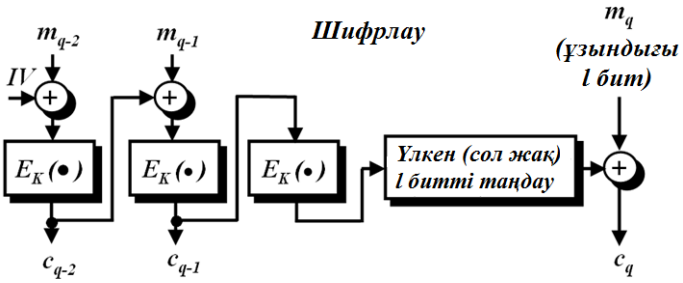
Берілген режимнің артықшылығы, ол жіберу кезінде қателердің жиналуына мүмкіндік бермейді. (6.4) сәйкес mi блогы керішифрлаудан кейін ci-1 және ci функциясы болады. Сондықтан жіберу кезіндегі қате алғашқы деректердің mi-1 және mi екі блогын жоғалтуға алып келеді, алғашқы деректердің бірінші блогы толығымен жоғалса, екінші блокта тек бір бит өзгереді, ол бит өзгерген биттің нөміріне сәйкес келетін бит.

6.2 суретінде көрсетілгендей деректерді шифрлау үшін бір кездейсоқ деректер блогы IV қолданылады. Бұл блоктың ешқандай маңызды мәні жоқ, ол әр хабарды бірегей жасау үшін қолданады. Жақсы IV ретінде уақыт белгесін немесе кездейсоқ биттер тізбегін қолдануға болады.

Бірдей алғашқы деректермен хабарлар IV қолданғанда шифрлау кезінде әртүрлі шифрланған деректермен хабарларға түрлендіріледі. Сондықтан қасқой шифрланған деректер блогын ауыстыруға немесе ендіруге (жоюға) мүмкіндік алмайды.

Әр жіберілетін хабар үшін IV бірегейлік талабы міндетті емес. Сонымен қатар IV құпия сақталмайды, ол шифрланған деректермен ашық жіберілу мүмкін. Жіберілетін хабар бірнеше блоктардан тұрсын m1, m2, … , mq. m1 IV шифрланады. m2 IV рөлінде m1 шифрланған деректерді қолданумен шифрланады. m3 IV рөлінде m2 шифрланған деректерді қолданумен шифрланады, және ары қарай. Сонымен, егер деректер блогының саны q тең болса, онда егер алғашқы IV құпия сақталса да, q - 1 «инициализация векторлары» ашық. Сондықтан IV құпия сақтауға себептер жоқ, IV – бітеуіш блогы, оны нөлдік жалғау блогы m0 деп есептеуге болады.

Толықтыру ECB режиміндегідей қолданылады, бірақ кейбір қосымшаларда шифрланған деректер көлемі алғашқы деректер көлемімен нақты сәйкес келу керек. Мысалы, шифрланған файл жадыда ашық деректер файлымен бірдей көлем алу керек. Бұл жағдайда соңғы қысқа блокты басқаша шифрлауға тұра келеді. Соңғы толық емес блок l биттен тұрсын. Соңғы толық блокты шифрлап алып (6.3 сурет) шифрланған деректердің q-1–ші блогын қайта шифрлау қажет. Содан кейін, алынған шифрланған деректер блогынан үлкен (сол жақтағы) l битты таңдап, солар үшін және қысқа блок үшін xor операциясын орындап, соңғы шифрланған деректерді аламыз.



6.3 сурет - *CBC* режимінде соңғы қысқа блокты шифрлау

6.3. “КЕРІ БАЙЛАНЫС” ОРЫНДАУ РЕЖИМІ

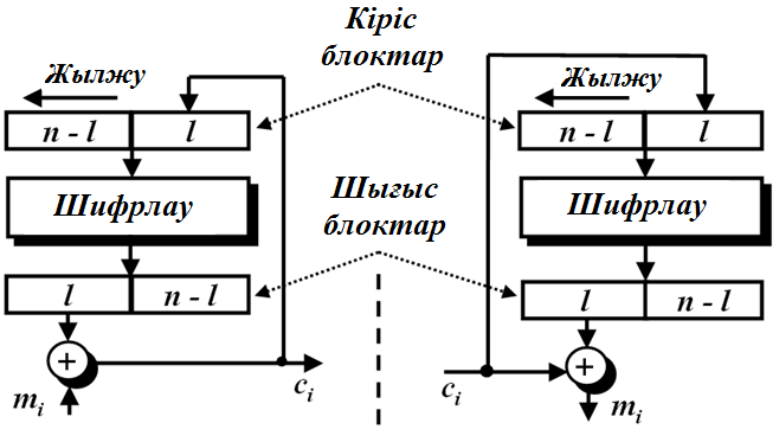
ECB және *CBC* режимдері хабар блоктарын шифрлау және керішифрлау үшін арналған. *ECB* және *CBC* режимдерінде егер деректердің тұтас блогы болмаса, деректерді шифрлау басталмайды. Бұл кейбір желілік қосымшалар үшін проблемаларды тұғызады. Мысалы, қауіпсіз желілік ортада терминал (клиент) басты компьютерге (серверге) символды енгізгеннең кейін үздіксіз жіберу қажет. Егер деректерді байттармен өңдеу қажет болса, онда *ECB* және *CBC* режимдері талаптарға сәйкес келмейді.

6.3.1. “Шифрланған деректер бойынша кері байланыс” орындау режимі

Шешім симметриялық блокты шифрлау алгоритмдерін немесе басқаларын шифрланған деректер бойынша кері байланыс режимінде (*CFB*) қолдануда. Бұл режимде блок ұзындығы *n*, бірақ

алғашқы немесе шифрланған деректер блогының ұзындығы – l, бұл жерде $l < n$.

Идея келесіде: симметриялық блокты шифрлау алгоритмдерін алғашқы деректерді шифрлау немесе шифрланған деректерді керішифрлау үшін қолданбау, оны n ұзындығы бар жылжымалы регистр мазмұнын шифрлау және керішифрлау үшін қолдану. Шифрлау алғашқы деректердің l-биттік блогына және l-биттік жылжымалы регистрге *xor* операциясын қолданумен жасалған (6.4 сурет).



6.4 сурет – «Шифрланған деректер бойынша кері байланыс» орындау режимінің құрылымы

Кіріс блогында (солға жылжыту регистры) шифрлау кезінде алдымен оң жақпен теңестірген инициализация векторы болады.

Алғашқы деректерді блоктарға бөлу нәтижесінде l биттен тұратын q блок алынды дейік. Сонда шифрланған деректердің әр i-ші блогы келесідей анықталады:

$$c_i = m_i \oplus g_{i-1},\ i=1,2,\ldots,q, \qquad (6.5)$$

бұл жерде g_{i-1} – шифрланған деректердің алдынғы шығыс блогының l жоғарғы битын белгілейді.

Жылжымалы регистрді (кіріс блокты) жаңарту оның деректерін сол жаққа l битке жылжыту және l кіші (оң жақтағы) разрядтар орнына c_i жазу арқылы өткізіледі.

Керішифрлау *l*-биттік шифрланған деректер блогына және *l*-биттік жылжымалы регистрге *xor* операциясын қолданумен жасалған. Кіріс блогында (солға жылжыту регистры) керішифрлау кезінде алдымен оң жақпен теңестірген инициализация векторы болады. Бірінші блок үшін жылжымалы регистр мазмұны - *IV* – жылжымайтынына назар аударыңыз.

Сонда керішифрланған (алғашқы) деректердің әр *i*-ші блогы келесі ретпен анықталады:

$$m_i = c_i \oplus g_{i-1},\ i=1,2,\ldots,q, \qquad (6.6)$$

бұл жерде g_{i-1} керішифрланған деректердің алдынғы шығыс блогының жоғарғы *l* битын белгілейді.

Әр блок үшін жылжымалы регистр мазмұнын (алдынғы жылжымалы регистр) *l* битке сол жаққа жылжытуды орындайды, c_{i-1} ішінен ең оң жақтағы *l* битты толтырады. Жылжымалы регистр мазмұны шифрланады және тек ең сол жақтағы *l* бит шифрланған деректермен бірге *xor* көмегімен өңделеді c_i блогынан m_i алып отырып. Бірінші блок үшін жылжымалы регистр мазмұны - *IV* – жылжымайтынына назар аударыңыз.

Бұл режимде блоктарды толықтыру талап етілмейтіні қызықты, себебі блок ұзындығы, *l*, әдетте шифрлау үшін деректер блогының көлеміне (мысалы, символ) сәйкес таңдалады. Басқа да қызықты – шифрлауды бастау үшін жүйе үлкен деректер блогын (*64* бит немесе *128* битті) күту міндетті емес. Шифрлау үрдісі кішкене деректер блогына (символ сияқты) орындалады. Бұл екі артықшылық екі кемшілікке алып келеді. *CFB* режимінің *ECB* немесе *CBC* қарағанда тиімділігі төмен, себебі ол негізгі блокты шифрмен *l* көлемі бар кішкене блокты шифрлауды орындайды.

CBC режимі сияқты CFB режимі, шифрланған деректер барлық алдынғы деректерден тәуелді болу үшін, ашық деректердің символдарын бірге байланыстырады.

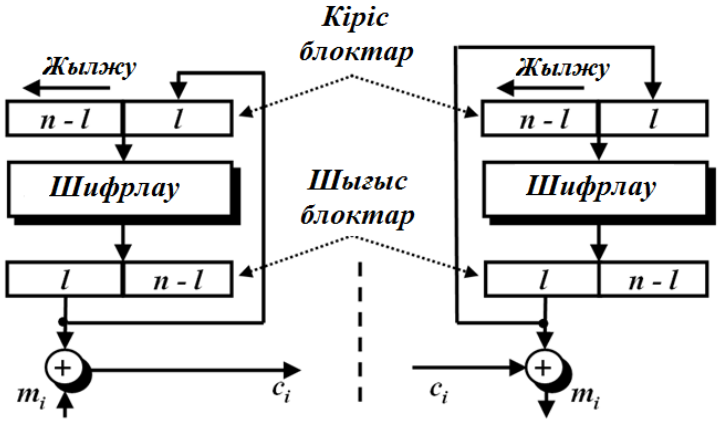
CBC режимінде сияқты IV бірегей болу керек, сонымен қатар, IV әр хабар үшін өзгеру керек.

CFB режимінде ашық деректердегі қате барлық келесі шифрланған деректерге әсер етеді, бірақ керішифрлау кезінде қағажуланады (самоустраняется). Шифрланған деректердегі қате қызықтырақ. Шифрланған деректердегі биттің қатесінің бірінші әсері керішифрланған деректерде бір биттегі қате болады. Содан кейін қате жылжымалы регистрге барады, және қате бит регистрден

шыққанша дұрыс емес керішифрланған деректер қалыптастырылады.

6.3.2. “Шығыс бойынша кері байланыс” орындау режимі

OFB орындау режимі де айнымалы блок көлемін және CFB режимінде сияқты инициализацияланатындай жылжымалы регистрді қолданады, дәлірек – кіріс блогында алдымен оң жақпен теңестірілген IV алғашқы мәні болады (6.5 сурет).



6.5 сурет – «Шығыс бойынша кері байланыс» орындау режимінің құрылымы

Бұл жерде деректерді шифрлаудың әр сеансы үшін регистр күйінің жаңа алғашқы мәнін қолдану қажет, ол арна бойынша ашық деректермен бірге жіберілу керек.

Ашық хабар блоктар түрінде көрсетіледі дейік

$$m_1, m_2, \ldots, m_q.$$

Әр i = 1,2,3,…,q үшін

$$c_i = m_i \oplus g_{i-1}. \qquad (6.7)$$

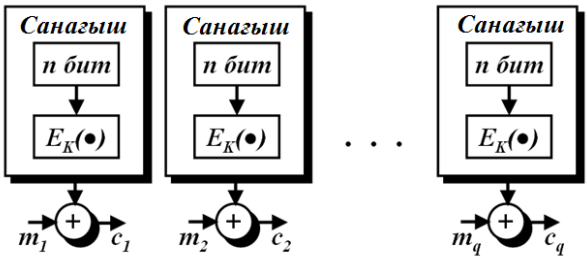
Шифрланған деректер бойынша кері байланыс режимінен айырмашылығы жылжымалы регистрді жаңарту әдісінде. Бұл жоғарғы l битті лақтырып тастап оң жақа gi жазу жолымен жүзеге асады.

OFB режимінің негізгі артықшылығы келесіде: егер жіберу кезінде қате болса, онда ол келесі шифрланған блоктарға әсер етпейді және келесі блоктарды керішифрлауға мүмкіндік сақталады. Мысалы, егер қате бит c_i болса, онда ол тек сол блокты керішифрламау және m_i алмау мүмкіндігіне алып келеді. Келесі блоктар дұрыс керішифрланады.

OFB кемшілігі, *CFB* қарағанда ол хабар ағынын түрлендіру шабуылдарына осалдау.

6.4. "САНАҒЫШ" ОРЫНДАУ РЕЖИМІ

Санағыш режимінде (*CTR*) кері байланыс ақпараты жоқ. Псевдокездейсоқ кілттік ағын санағыш көмегімен жасалады. N битке санағыш алдын ала анықталған мәнге (*IV*) инициализацияланады және негізгі және алдын ала анықталған ереже ($mod\ 2^n$) бойынша ұлғайады. Кездейсоқтықты қамтамасыз ету үшін ұлғаю көлемі блок нөмірінен тәуелді болу мүмкін. Алғашқы және шифрланған деректерде негізгі шифр сияқты бірдей блок көлемі болады. Алғашқы деректердің n көлемі бар блоктары шифрланған деректердің блок көлемі n болатындай шифрланады. 6.6 суретінде санағыш режимінде деректерді шифрлау көрсетілген.



6.6 сурет – «Санағыш» орындау режимінде деректерді шифрлау құрылымы

Алғашқы деректер және шифрланған деректер блоктары арасындағы қатынас төменде көрсетілген.

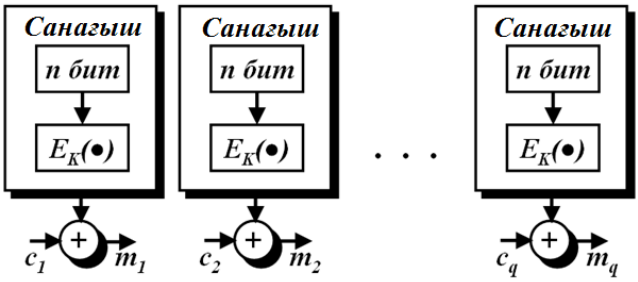
Шифрлау:

$$c_i = m_i \oplus E_{K_i}(n_i),$$

бұл жерде n_i – санағыштың n-биттік мәні.

Керішифрлау (6.7 сурет):

$$m_i = c_i \oplus E_{K_i}(n_i).$$



6.7 сурет – «Санағыш» орындау режимінде деректерді керішифрлау құрылымы

CTR режимі негізгі блокты шифрдың шифрлау функциясын (E_K) шифрлау және керішифрлау үшін қолданады. Алғашқы деректер блогы m_i шифрланған деректерден c_i қалпына келтірілу мүмкін екенін жеңіл дәлелдеуге болады.

CTR режимін *OFB* және *ECB* режимдерімен салыстыруға болады. Толығырақ *OFB, CTR* алдынғы шифрланған деректер блогынан тәуелсіз кілттік ағынды құрады, бірақ *CTR* керібайланыс ақпаратын қолданбайды. *ECB, CTR* сияқты, блоктары бір бірінен тәуелсіз, *n*-биттік шифрланған деректерді құрады – олар тек санағыш мәндерінен тәуелді болады. Бұл қасиеттің кем жері *CTR* режимі *ECB* режимі сияқты реалды уақыт масштабында өңдеу үшін қолданылмайды. Шифрлау алгоритмі шифрлау алдында аяқталған *n*-разрядты деректер блогын күтеді. Бұл қасиеттің артықшылығы *CTR* режимі *ECB* режимі сияқты ерікті қол жеткізу файлдарды шифрлау және керішифрлау үшін қолдану мүмкін, және санағыш мәні файлдағы жазба нөмірімен байланысу мүмкін.

CTR режимі үшін қауіпсіздік проблемалары *OFB* режимдағыдай. Шифрланған деректердегі бір қате керішифрланған деректерде тек сәйкес битке қатысты.

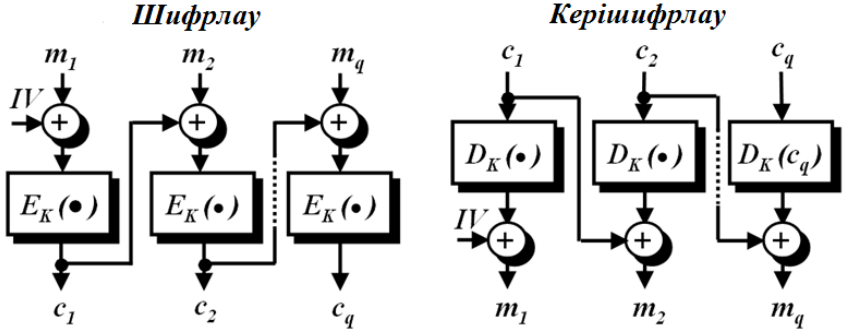
6.5. ШИФРЛАУДЫҢ БАСҚА РЕЖИМДЕРІ

Блокты симметриялық криптографиялық алгоритмдер көмегімен деректерді шифрлаудың бір қатар қосымша режимдері бар [23]:

- *шифрланған деректер блоктарының таратылған тіркелу режимі* (*Propagating Cipher Block Chaining – PCBC*);
- *шифрланған деректер блоктарын бақылау сомамен тіркелу режимі* (*Cipher Block Chaining with Checksum – CBCC*);
- *сызықты емес функциямен шығыс бойынша кері байланыс режимі* (*Output Feedback With a Nonlinear Function – OFBNLF*);
- *ашық деректер бойынша кері байланыс режимі* (*PFB*);
- *ашық деректер блоктарының тіркелу режимі* (*PBC*) және т.б.

Шифрланған деректер блоктарының таратылған тіркелу режимі. *CBC* режимінің потенциалды проблемаларының бірі келесі керішифрланған ашық деректер блогына бақыланатын өзгерістерді енгізу мүмкіндігі. Мысалы, егер қасқой блокта бір битті өзгертсе, онда барлық блок дұрыс емес керішифрланады, бірақ келесі блокта сәйкес позицияда қате пайда болады. Бұл ұнамсыз болатын жағдайлар болады. Бұл қауіпке қарсы тұру үшін ашық деректерде анықталған артықтық болу қажет.

PCBC режимі *CBC* режиміне ұқсас, келесіге қарамағанда: алдынғы ашық деректер блогы сияқты, алдынғы шифрланған деректер блогы шифрлау алдында (6.8, а сурет) немесе керішифрлағаннан кейін (6.8, б сурет) ағымдағы ашық деректер блогымен *xor* операциясын орындайды.



а) б)

6.8 сурет - *PCBC* режимінде деректерді шифрлау құрылымы:
а) шифрлау; б) керішифрлау

Бұл жағдайда деректерді шифрлау және керішифрлау келесі өрнектерді қолданумен орындалады:

$$c_i = E_k (m_i \oplus c_{i-1} \oplus m_{i-1}),\ i=1,2,...,q,\ c_0=IV;$$

$$m_i = c_{i-1} \oplus m_{i-1} \oplus D_k (c_i),\ i=1,2,...,q,\ c_0=IV.$$

PCBC режимі *Kerberos* хаттамасында бір өткенде шифрлауды және деректердің тұтастығын тексеруді орындау үшін қолданады. *PCBC* режимінде шифрланған деректердегі қате келесі барлық блоктарды дұрыс емес керішифрлауға алып келеді.

Бұл, хабардың соңындағы стандартты блокты тексеру хабардың тұтастығын тексеру үшін қолданатынын білдіреді.

Бұл режимнің кемшілігі келесіде: шифрланған деректердің екі блогын алмастыру екі сәйкес келетін ашық деректер блоктарын қате керішифрлауға алып келеді. Бірақ ашық және шифрланған деректерге *xor* операциясының табиғаты арқасында болашақ қателер орнын толтырады. Сондықтан, егер тұтастықты тексеру тек бірнеше соңғы керішифрланған (ашық) деректер блоктарына орындалса, онда бөлшекті қате хабарды алуға болады.

Бұл күмәнді қасиет құрастырушыларды *Kerberos* хаттамасының келесі нұсқасында бұл режимнен бас тартып *CBC* режимін қолдануға күштеді.

Шифрланған деректер блоктарын бақылау сомамен тіркелу режимі CBC режимінен айырмашылығы тек келесіде: шифрлаудың алдында алғашқы деректердің соңғы блогына алғашқы деректердің барлық алдынғы блоктары үшін соммасы *2* модулі бойынша қосылады (6.9 сурет).

Бұл жіберілетін деректердің тұтастығын үлкен емес үстеме шығындармен бақылауға мүмкіндік береді.

Бұл жағдайда деректерді шифрлау және керішифрлау келесі өрнектері қолданумен жүзеге асырылады:

$$s = (m_1 \oplus m_{i+1} \oplus ... \oplus m_{q-1});$$

$$c_i = E_k (m_i \oplus c_{i-1}),\ i=1,2,...,q-1,\ c_0=IV;$$

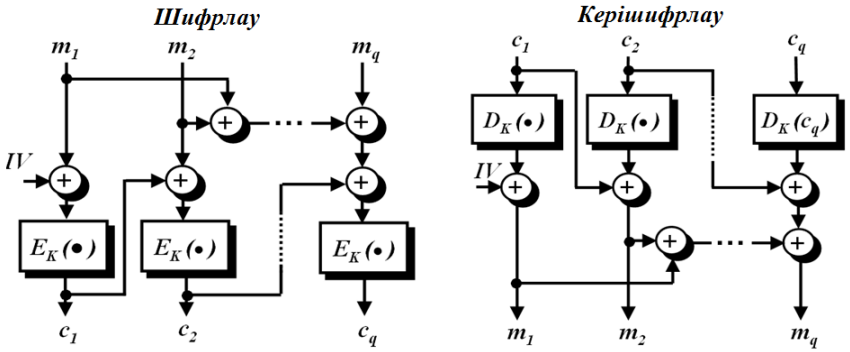
$$c_q = E_k (m_q \oplus s \oplus c_{q-1}),$$

ал керішифрлау – өрнектерді қолданумен:

$$m_i = D_k(c_i) \oplus c_{i-1},\ i=1,2,\ldots,q\text{-}1,\ c_0=IV;$$

$$s = (m_1 \oplus m_{i+1} \oplus \ldots \oplus m_{q-1});$$

$$m_q = E_k(c_q) \oplus c_{q-1} \oplus s).$$



а) б)

6.9 сурет – Шифрланған деректер блоктарын бақылау сомасымен тіркелу режимінде деректерді шифрлау құрылымы:
а) шифрлау; б) керішифрлау

Шифрланған деректердегі бір қате ашық деректердің тек бір блогына таралады, бірақ синхрондауды қолдау қажет.

Ақпаратты өңдеу жылдамдығы базалық алгоритммен шифрлау жылдамдығымен ғана емес, сонымен қатар кілтің ағымдағы мәнін жаңарту жыдамдығымен анықталады.

Ашық деректер блоктарының тіркелу режимі (*Plaintext Block Chaining - PBC*) *CBC* ұқсас, тек *xor* операциясы шифрланған деректер блогына емес, ағымдағы және алдындағы ашық деректер блогына орындалады.

Ашық деректер бойынша кері байланыс режимі (*Plaintext Feedback – PFB*) *CFB* режиміне ұқсас, бірақ кері байланыс үшін шифрланған емес, ашық деректер қолданылады. Екі режимде арнайы таңдалған ашық деректерді қолданумен бұзуға осал. Бұзуға беріктілікті жоғарлату мақсатымен екі режимде ашық мәтіндерді қолдануға рұқсат береді.

Бақылау сұрақтары және тапсырмалар

1. Егер шифрлау үшін заманауи блокты шифрлар қолданса жұмыс істеу режимдері не үшін керек екенін түсіндіріңіз.
2. Симметриялық блокты шифрлардың негізгі жұмыс істеу режимдерін келтіріңіз.
3. *ECB* режимін анықтаңыз және оның артықшылықтарымен кемшіліктерін келтіріңіз.
4. *CBC* режимін анықтаңыз және оның артықшылықтарымен кемшіліктерін келтіріңіз.
5. *CFB* режимін анықтаңыз және оның артықшылықтарымен кемшіліктерін келтіріңіз.
6. *OFB* режимін анықтаңыз және оның артықшылықтарымен кемшіліктерін келтіріңіз.
7. *CTR* режимін анықтаңыз және оның артықшылықтарымен кемшіліктерін келтіріңіз.
8. Негізгі жұмыс істеу режимдерін екі топқа бөліңіз: шифрлау және керішифрлау функцияларын қолданатын – негізгі шифрлар (мысалы, *DES*), және тек шифрлау функцияларын қолданатын.
9. Негізгі жұмыс істеу режимдерін екі топқа бөліңіз: қосымша деректерді талап ететін, және талап етпейтін.
10. Негізгі жұмыс істеу режимдерін екі топқа бөліңіз: сол кілтті барлық блоктарды шифрлау үшін қолданатын, және кілттік ағынды блоктарды шифрлау үшін қолданатын.
11. Қатарлас өңдеумен жылдамдалу мүмкіндігі бар жұмыс істеу режимдерін келтіріңіз.
12. Ерікті қол жеткізу файлдарын шифрлау үшін қолдану мүмкіндігі бар жұмыс істеу режимдерін келтіріңіз.
13. Неге *CFB* режимі синхронды емес ағын шифрын, ал *OFB* – синхрондысын құратынын көрсетіңіз.
14. *CFB* режимінде жіберу кезінде қатенің бір биты қанша блокқа әсер етеді?
15. *ECB* режимінде ($n = 64$) шифрланған деректердің бірінші блогында бір бит (*5-ші*) жіберу кезінде өзгерген. Керішифрланған деректерде мүмкінді биттер өзгерістерін анықтаңыз.
16. *CBC* режимінде ($n = 64$) шифрланған деректердің бірінші блогында бір бит (*5-ші*) жіберу кезінде өзгерген. Керішифрланған деректерде мүмкінді биттер өзгерістерін анықтаңыз.

17. *CFB* режимінде (*l = 8, n = 64*) ашық деректердің бірінші блогының екінші биты өзгерген. Керішифрланған деректерде мүмкінді биттер өзгерістерін анықтаңыз.

18. *CFB* режимінде (*l = 8, n = 64*) ашық деректердің бірінші блогының екінші биты өзгерген. Керішифрланған деректерде мүмкінді биттер өзгерістерін анықтаңыз.

19. *OFB* режимінде (*l = 8, n = 64*) ашық деректердің бірінші блогының екінші биты өзгерген. Керішифрланған деректерде мүмкінді биттер өзгерістерін анықтаңыз.

20. *CTR* режимінде (*n = 64*) шифрланған деректердің екінші блогының екінші биты өзгерген. Керішифрланған деректерде мүмкінді биттер өзгерістерін анықтаңыз.

21. Жіберушімен қолданатын ашық деректер *CFB* режимінде қабылдаушымен қалпына келтірілу мүмкін екенін дәлелдеңіз.

22. Жіберушімен қолданатын ашық деректер *OFB* режимінде қабылдаушымен қалпына келтірілу мүмкін екенін дәлелдеңіз.

23. Жіберушімен қолданатын ашық деректер *CTR* режимінде қабылдаушымен қалпына келтірілу мүмкін екенін дәлелдеңіз.

7 тарау

ГОСТ 28147-89 ДЕРЕКТЕРДІ КРИПТОГРАФИЯЛЫҚ ТҮРЛЕНДІРУ СТАНДАРТЫ

7.1. ГОСТ 28147-89 СТАНДАРТЫН ҚҰРУ ТАРИХЫ

Ақпараттық жүйелерде ақпараттың кепілденген қауіпсіздігін қамтамасыз етудің маңызды мақсаты деректерді шифрлаудың стандарттық алгоритмдерін құру және қолдану. Осындай стандарттардың ішінде бірінші болған, 5 тарауда көрсетілген, ауыстыруларды және алмастыруларды тізбекті қолданатын, американдық *DES* стандарты. Кәзіргі уақытта жиі *DES* үлкен қиындығы және төмен криптографиялық беріктілігі туралы айтады. Тәжірибеде оның түрлендірулерін қолдануға мәжбүр.

Тиімдірек болатын отандық деректерді криптографиялық түрлендіру стандарты *ГОСТ 28147-89* [12, 29 36].

Ол екілік код түрінде берілген кез келген деректерді қорғау үшін ұсынылған, деген мен басқа да шифрлау әдістері аласталмаған. Берілген стандарт әлемдік тәжірибені есепке алып қалыптастырылған, соның ішінде *DES* алгоритмінің кемшіліктері және жүзеге асырылмаған мүмкіндіктері қарастырылған, сондықтан *ГОСТ* стандартын қолдану ұнамды. Алгоритм жеткідікті қиын және төменде көбінесе оның концепциясы қарастырылады.

Бұл стандарт *ГОСТ 28147-89* деп бекітілген, оның атынан көрініп тұрғандай 1989 жылы КСРО-да қабылданған және 1990 жылдың бірінші шілдесінде іске қосылған [36]. Бірақ, бұл шифрдың тарихы оданда терең. Стандарт болжаумен КСРО МҚК сегізінші бас басқаруында, қазіргі уақытта *ФАПСИ*-ге (*РФ Президент жанындағы үкіменттік байланыс және ақпараттың федералды агенттігі*) түрлендірілген, өмірге келген. XX ғасырдың 70-ші жылдары стандартта *"Толық құпия"* грифі тұрған, уақыт өте гриф

"құпия" дегенге өзгерді, содан кейін оны алып тастады. Өкінішке орай, стандарттың құру тарихы және шифрды жобалау критерийлері, стандарттың өзіне қарағанда, кәзірге дейін *"жеті мөр астындағы құпия"* болып тұр [2].

ГОСТ 28147-89 шифрлау стандартының сипаттамасы *"Ақпаратты өңдеу жүйелері. Криптографиялық қорғау. ГОСТ 28147-89 криптографиялық түрлендіру алгоритмі"* деп аталатын өте қызықты құжатта келтірілген [36]. Оның атында "*шифрлау*" терминінің орнында "*криптографиялық түрлендіру*" жалпы түсінігі тұрғаны кездейсоқ емес. Бір бірімен тығыз байланысқан бірнеше шифрлау процедуралардан басқа, құжатта олармен бір принцип негізінде жасалған *имитоендірмені* өндіру алгоритмі сипатталған. Ол криптографиялық бақылау комбинациясы, яғни *имитоқорғау* немесе рұқсатсыз өзгерістерден деректерді қорғау мақсаты үшін құпия кілтті қолданумен алғашқы деректерден өндірілетін код болады.

ГОСТ 28147-89 деректерді криптографиялық түрлендіру стандарты *64* бит ұзындығы бар деректермен жұмыс істейді. Деректерді шифрлау және керішифрлау *32* немесе *16* раунд (цикл) қолдануымен жүзеге асады. Шифр кілті *256* бит, одан *32* биттік сегіз ішкі кілт қалыптастырылады (осы сегіз ішкі кілттен *32* (*16*) раундтық (циклдық) кілттер қалыптастырылады).

Стандартта төрт жұмыс режимі қарастырылған:

- қарапайым ауыстыру режимінде деректерді шифрлау;
- гаммалау режимінде деректерді шифрлау;
- кері байланысы бар гаммалау режимінде деректерді шифрлау;
- имитоендірмені өндіру мақсатында шифрлау.

7.2. ГОСТ 28147-89 МАТЕМАТИКАЛЫҚ АЛҒЫШАРТТАРЫ

ГОСТ 28147-89 алгоритмінің әртүрлі қадамдарында ол жұмыс істейтін деректер әртүрлі қолданылады. Кейбір жағдайларда деректер элементтері тәуелсіз биттер массивтері ретінде өнделеді, басқа жағдайларда - белгісіз бүтін сан ретінде, үшіншіде - бірнеше қарапайым элементтерден тұратын құрылымы бар күрделі элемент ретінде. Сондықтан шатаспас үшін қолданатын белгілер туралы алдын ала келісу қажет.

Деректер элементтері көлбеуі бар үлкен латын әріптерімен (мысалы, *X*) белгіленеді. |*X*| битпен берілген *X* элементінің көлемі

белгіленеді. Сонымен, егер X деректер элементін бүтін теріс емес сан ретінде қарастырса, онда келесі теңсіздікті жазуға болады: $0 \le X \le 2^{|X|}$.

Егер деректер элементі кіші көлемі бар бірнеше элементтерден тұрса, онда бұл факт келесідей белгіленеді:

$$X = (X_0, X_1, ..., X_{n-1}) = X_0 \,||\, X_1 \,||\, ... \,||\, X_{n-1}.$$

Бірнеше деректер элементтерін біреуге біріктіру процедурасы *конкатенация* деп аталады және "||" символымен белгіленеді. Деректер элементтер көлемдері үшін келесі қатынас орындалу керек:

$$|X| = |X_0| + |X_1| + ... + |X_{n-1}|.$$

Деректердің күрделі элементтерін құру және конкатенация операциясы кезінде деректер элементтері өсу ретімен беріледі. Басқа сөзбен, егер күрделі элементті және оның ішіне кіретін деректер элементтерін белгісіз бүтін сандар ретінде қарастырса, онда келесі теңдікті жазуға болады:

$$X = (X_0, X_1, ..., X_{n-1}) = X_0 \,||\, X_1 \,||\, ... \,||\, X_{n-1} =$$

$$= X_0 + 2^{|X_0|} \cdot (X_1 + 2^{|X_1|} \cdot (X_2 + 2^{|X_2|} \cdot (... 2^{|X_{n-2}|} \cdot X_{n-1}) ...)).$$

Алгоритмде деректер элементі бөлек биттер массиві ретінде қарастырылу мүмкін. Бұл жағдайда биттер массив белгіленген әріппен белгіленеді, бірақ жол нұсқасымен, келесі мысалда көрсетілгендей:

$$X = (x_0, x_1, \ ... \ , x_{n-1}) = x_0 \cdot 2^0 + x_1 \cdot 2^1 + ... \ x_{n-1} \cdot 2^{n-1}.$$

Сонымен, *ГОСТ* үшін "*little-endian*" деп аталатын разрядтарды нөмірлеу қабылданған, яғни көпразрядты сөздер деректері ішінде бөлек екілік разрядтар және кіші нөмірлері бар олардың топтарының маңыздылығы кішірек. Бұл туралы тұра айтылған: *"Екілік векторларды жоғары разрядтармен қосу және циклды жылжыту кезінде үлкен нөмірлері бар жинақтаушылардың разрядтары есептеледі"* [36]. *ГОСТ 28147-89* алгоритмі виртуалды шифрлау құрылғысының жинақтаушы-регистрлерін деректермен толықтыруды кіші разрядтардан бастауды ұсынады, яғни

маңыздылығы төмен разрядтардан. Тұра осындай нөмірлеу реті *Intel x86* шығанпроцессорлық сәулетінде қабылданған, сондықтан берілген сәулетте шифрды бағдарламалық жүзеге асыруда деректер сөздері ішінде ешқандай қосымша разрядтарды ауыстыру талап етілмейді.

Егер деректер элементтеріне логикалық мәні бар кейбір операция орындалса, онда осы операция элементтердің сәйкес биттеріне орындалады деп есептеледі. Басқа сөзбен,

$$A \bullet B = (a_0 \bullet b_0,\ a_1 \bullet b_1,\ a_{n-1} \bullet b_{n-1}),$$

бұл жерде $n = |A| = |B|$, ал "$\bullet$" символымен кез келген бинарлық логикалық операция белгіленеді; ереже ретінде, *xor* операциясы, басқаша айтқанда - *2* модулі бойынша қосу операциясы:

$$a \oplus b = (a + b) \bmod 2.$$

Сонымен қоса, алгоритмде екі *32*-разрядты бүтін оң сандарды 2^{32} ($\boxplus$ деп белгіленеді) және 2^{32}-*1* ($\boxplus$) модулі бойынша қосу операциясы қолданады.

Екі сан *a* және *b*, бұл жерде $0 \le a < 2^{32}$, $0 \le b < 2^{32}$:

$$a = (a_{32}, a_{31}, \ldots, a_2, a_1) \quad \text{және} \quad b = (b_{32}, b_{31}, \ldots, b_2, b_1),$$

екілік түрде көрсетілген, яғни

$$a = a_{32} \cdot 2^{31} + a_{31} \cdot 2^{30} + \ldots + a_2 \cdot 2^1 + a_1 \cdot 2^0;$$

$$b = b_{32} \cdot 2^{31} + b_{31} \cdot 2^{30} + \ldots + b_2 \cdot 2^1 + b_1 \cdot 2^0,$$

2^{32} (операция $\boxplus$) модулі бойынша келесі ереже бойынша қосылады:

$$a \boxplus b = a + b, \text{ егер } a + b < 2^{32};$$

$$a \boxplus b = a + b - 2^{32}, \text{ егер } a + b \ge 2^{32}.$$

Екі бүтін сан *a* және *b*, бұл жерде $0 \le a < 2^{32}$-*1*, $0 \le b < 2^{32}$–*1*, сол түрде көрсетілген, келесі ереже бойынша 2^{32} (операция $\boxplus$) модулімен қосылады:

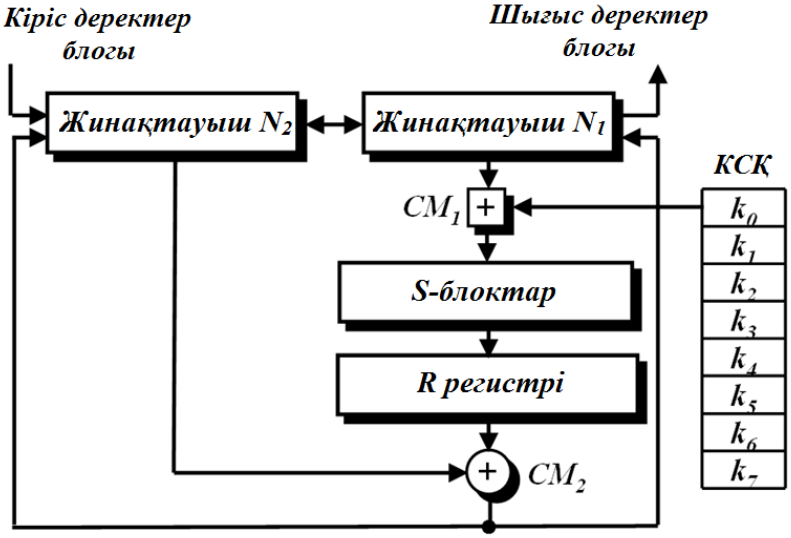
$$a \boxplus b = a + b, \text{ егер } a + b < 2^{32}\text{-}1,$$

$$a \boxplus b = a + b - (2^{32}\text{-}1), \text{ егер } a + b \ge 2^{32}\text{-}1.$$

7.3. ГОСТ 28147-89 ҚАРАПАЙЫМ АУЫСТЫРУ РЕЖИМІНДЕ ДЕРЕКТЕРДІ ШИФРЛАУ

7.3.1. ГОСТ 28147-89 қарапайым ауыстыру режимінде деректерді шифрлаудың криптографиялық жүйесінің құрылымы

Қарапайым ауыстыру (ұқсас режим - *"Электронды кодты кітап"*) режимінде шифрлау алгоритмін жүзеге асыру үшін жалпы криптографиялық жүйенің тек бөліктері қолданылады (7.1 сурет).



7.1 сурет - *ГОСТ 28147-89* стандартының қарапайым ауыстыру режимін жүзеге асыру құрылымы

Сұлба бойынша белгілер:

- N_1 және N_2 - *32*-разрядты жинақтауштар (жылжымалы регистрлер);
- CM_1 - *32*-разрядты 2^{32} ($\boxplus$) модулі бойынша сумматор;
- CM_2 - *32*-разрядты 2 ($\oplus$) модулі бойынша сумматор;
- R – *32*-разрядты деректерді *11* разрядқа солға (жоғары разрядтар жағына) жылжытатын циклдық жылжыту регистрі;
- *КСҚ – 256* биттік кілтті сақтау құрылғысы, сегіз *32*-разрядты жинақтауштардан k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7 тұрады;

- ауыстырудың *S*-блогы, сегіз ауыстыру түйінінен тұрады: S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7.

7.3.2. ГОСТ 28147-89 қарапайым ауыстыру режимінде деректерді шифрлау

Шифрланатын ашық деректер *64*-разрядты блоктарға бөлінеді. T_O *64*-разрядты блокты қарапайым ауыстыру режимінде шифрлау процедурасы *32* циклдан (раундтан) тұрады. *Кілтті сақтау құрылғысына* (КСҚ) шифр кілтінің *K 256* битін сегіз *32*-разрядты ішкі кілттер (сандар) k_i түрінде енгізеді:

$$K = (k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7).$$

Блок битының тізбегін

$$T_O = (a_1(0), a_2(0), \dots , a_{32}(0), b_1(0), b_2(0), \dots , b_{32}(0))$$

32 биттен екі тізбекке бөледі (кері ретпен алып):

$$b(0) = b_{32}(0), b_{31}(0), \dots , b_1(0) \text{ и } a(0) = a_{32}(0), a_{33}(0), \dots , a_1(0),$$

бұл жерде *b(0)* – сол жақ немесе үлкен, ал *a(0)* – оң жақ немесе кіші биттер.

Тізбектерді *b(0)* және *a(0)* белгілеу кезінде төменгі жағындағы индекс бит нөмірін анықтайды.

Шифрлаудың бірінші цикл алдында бұл тізбектерді N_1 және N_2 жинақтауштарға енгізеді. Нәтижесінде N_1 жинақтауштың алғашқы мазмұны

$$N_1 = a(0) = (a_{32}(0), a_{31}(0), \dots , a_1(0)),$$

N_2 жинақтауштың алғашқы мазмұны

$$N_2 = b(0) = (b_{32}(0), b_{31}(0), \dots , b_1(0)).$$

Ашық деректердің *64*-разрядты блогының шифрлау процедурасының бірінші циклын келесі теңдеулермен сипаттауға болады:

$$\begin{cases} a(1) = f(a(0) \boxplus k_0) \oplus b(0); \\ b(1) = a(0), \end{cases}$$

бұл жерде $a(1) - N_1$ бірінші шифрлаудың бірінші циклынаң кейін мазмұны; $b(1) - N_2$ бірінші шифрлаудың бірінші циклынаң кейін мазмұны; $f(\bullet)$ – шифрлау функциясы.

$f(\bullet)$ функциясының аргументі $a(0)$ (N_1 жинақтауштың алғашқы мазмұны) санын және k_0 (КСҚ жинақтауышынан оқылатын ішкі кілт) санын 2^{32} модулі бойынша қосу нәтижесі. Әр сан *32* битке тең.

Мысалы, $a(0) = 1010\ 0100\ 0100\ 1101\ 1100\ 1011\ 0001\ 0101_2$, ал $k_0 = 1001\ 0100\ 1000\ 1001\ 1000\ 0101\ 0010\ 1001_2$.

2^{32} модулі бойынша қосу операциясын орындағаннаң кейін CM_1 сумматор шығысындағы нәтиже

$$a(0) \dot{\boxplus} k_0 = 0011\ 1000\ 1101\ 0111\ 0101\ 0000\ 0011\ 1110_2.$$

$f(\bullet)$ функциясына алынған *32*-разрядты қосындығыға $a(0) \dot{\boxplus} k_0$ екі операциясы бар.

Бірінші операция ауыстыру деп аталады және ауыстырудың *S*-блогымен орындалады. Ауыстырудың *S*-блогы, жоғарыда айтылғандай, сегіз ауыстыру түйінінен тұрады әрқайсысының жады көлемі *64* бит. CM_1 ауыстырудың *S*-блогына баратын *32*-разрядты векторды сегіз ретпен келетін *4*-разрядты векторға бөледі, олардың әрқайсысы сәйкес ауыстыру түйінімен төртразрядты векторға түрлендіріледі.

Ауыстырудың әр түйінін он алты төртразрядты екілік сандарды алмастыру кестесімен көрсетуге болады. Екілік сандардың диапазоны: $0000, 0001, \ldots, 1111_2$. Кіріс вектор кестедегі жол адресін көрсетеді, ал жолдағы сан шығыс векторы болады. Содан кейін төртразрядты векторларды ретпен *32*-разрядты векторға біріктіреді. Ауыстыру түйіндері (ауыстыру кестелері) кілттік элементтері деп есептеледі, олар ЭЕМ желісі үшін жалпы болады және сирек өзгереді. Бұл ауыстыру түйіндері құпия сақталу керек. Ауыстыру түйіндері *RFC 4357* құжатпен анықталған. 7.1 кестеде *S*-блок көмегімен ауыстыру түйін мысалы келтірілген. Бұл ауыстыру түйіні *ГОСТ Р 34.11-94* тестілеу мақсаттары үшін анықталған.

Мысалы, егер 2^{32} модулі бойынша қосу операциясын орындағаннаң кейін CM_1 сумматоры шығысындағы нәтиже

$$a(0) \dot{\boxplus} k_0 = 0000\ 0001\ 0010\ 0011\ 0100\ 0101\ 0110\ 0111_2,$$

онда 7.1 кестені есепке алып келесі мән қалыптастырылады

$$S(a(0) \boxplus k_0) = 0100\ 1011\ 0001\ 0001\ 0101\ 0010\ 0101\ 1110_2.$$

7.1 кесте

S-блоктар көмегімен деректерді ауыстыру

		Төртразрядты сөздер мәндері															
		00	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*
Ауыстырудың S-блоктар нөмірлері	*00*	*04*	*10*	*09*	*02*	*13*	*08*	*00*	*14*	*06*	*11*	*01*	*12*	*07*	*15*	*05*	*03*
	01	*14*	*11*	*04*	*12*	*06*	*13*	*15*	*10*	*02*	*03*	*08*	*01*	*00*	*07*	*05*	*09*
	02	*05*	*08*	*01*	*12*	*10*	*03*	*04*	*02*	*14*	*15*	*12*	*07*	*06*	*00*	*09*	*11*
	03	*07*	*13*	*10*	*01*	*00*	*08*	*09*	*15*	*14*	*04*	*06*	*12*	*11*	*02*	*05*	*03*
	04	*06*	*12*	*07*	*01*	*05*	*15*	*13*	*08*	*04*	*10*	*09*	*14*	*00*	*03*	*11*	*02*
	05	*04*	*11*	*10*	*00*	*07*	*02*	*01*	*13*	*03*	*06*	*08*	*05*	*09*	*12*	*15*	*14*
	06	*13*	*11*	*04*	*01*	*03*	*15*	*05*	*09*	*00*	*10*	*14*	*07*	*06*	*08*	*02*	*12*
	07	*04*	*10*	*09*	*02*	*13*	*08*	*00*	*14*	*06*	*11*	*01*	*12*	*07*	*15*	*05*	*03*

Екінші операция - ауыстырудың *S*-блок шығысынан алынған *32*-разрядты векторды сол жаққа циклды жылжыту (*11* разрядқа). Циклды жылжыту *R* жылжыту регистрімен орындалады.

Ары қарай $f(\bullet)$ шифрлау функциясының жұмыс нәтижесін сумматор CM_2 *2* модулі бойынша *32*-разрядты N_2 жинақтауыштың алғашқы мазмұнымен *b(0)* разряд бойынша қосады. Содан кейін N_1 мәнін N_2 жинақтауышқа жазады (*b(1) = a(0)*), ал CM_2 шығысында алынған нәтижені $(a(1) = f(a(0) \boxplus k_0) \oplus b(0)$ N_1 жинақтауышқа жазады. Осыдан кейін бірінші цикл аяқталады.

Келесі циклдар осы сияқты орындалады, екінші циклда КСҚ-нан k_1 ішкі кілтті оқиды, үшінші циклда k_2, және ары қарай, сегізінші циклда - k_7. *9*-шы циклдан *16*-шы ға дейін, сонымен қатар *17*-ші ден *24*-ші ге дейін КСҚ ішкі кілттер сол ретпен оқылады:

$$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7.$$

Сонғы сегіз циклда (*25*-ші ден *32*-шіге дейін) КСҚ ішкі кілттерді оқу реті керісінше: $k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0$.

Сонымен, *32* циклда шифрлау кезінде КСҚ ішкі кілттерді таңдау реті келесі:

$$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7,$$

$$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0.$$

32-ші циклды аяқтағанаң кейін CM_2 сумматорынан нәтиже N_2 жинақтауышына енгізіледі, ал N_1 жинақтауышында бұрынғы мән сақталады. Шифрлаудың *32*-ші циклынаң кейін алынған N_2 және N_1 жинақтауыштардың мазмұндары T_O ашық деректер блогына сәйкес келетін шифрланған деректер блогы $T_{Ш}$ болады.

Қарапайым ауыстыру режимінде шифрлау теңдіктері:

$$\begin{cases} a(j) = f(a(j-1) \boxplus k_{(j-1) \bmod 8}) \oplus b(j-1), \ j = 1, 2, \ldots, 24; \\ b(j) = a(j-1), \ j = 1, 2, \ldots, 24, \end{cases} \tag{7.1}$$

$$\begin{cases} a(j) = f(a(j-1) \boxplus k_{32-j}) \oplus b(j-1), \ j = 25, 26, \ldots, 31; \\ b(j) = a(j-1), \ j = 25, 26, \ldots, 31, \end{cases} \tag{7.2}$$

$$\begin{cases} b(j) = f(a(j-1) \boxplus k_{32-j}) \oplus b(j-1), \ j = 32; \\ a(j) = a(j-1), \ j = 32, \end{cases} \tag{7.3}$$

бұл жерде $a(j) = (a_{32}(j), a_{31}(j), \ldots, a_1(j))$ – шифрлаудың *j*-ші циклынаң кейін N_1 мазмұны; $b(j) = (b_{32}(j), b_{31}(j), \ldots, b_1(j))$ – шифрлаудың *j*-ші циклынаң кейін N_2 мазмұны, $j = 1,2, \ldots, 32$.

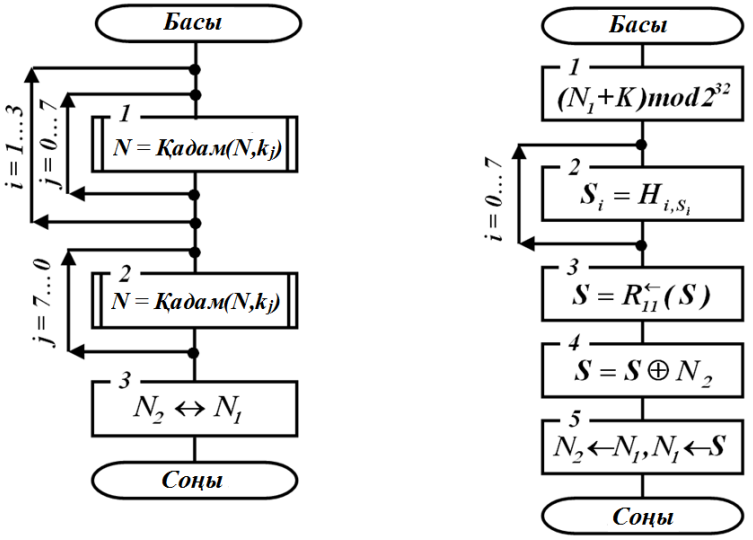
Шифрланған деректер блогы $T_{Ш}$ (*64* разряд) N_2 және N_1 жинақтауыштардан келесі ретпен шығарылады: N_1 жинақтауышынан *1...32* разрядтан, содан кейін N_2 жинақтауышынан *1...32* разрядтан, яғни кіші разрядтардан бастап:

$$T_{Ш} = (a_1(32), a_2(32), \ldots, a_{32}(32), b_1(32), b_2(32), \ldots, b_{32}(32)).$$

Ашық деректердің қалған блоктары қарапайым ауыстыру режимінде берілген алгоритммен шифрланады.

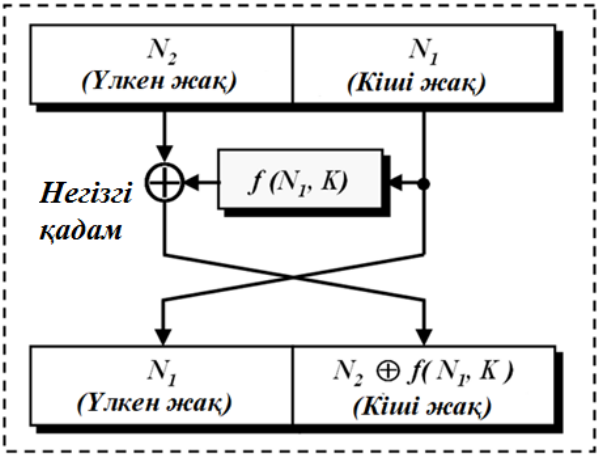
Қарапайым ауыстыру режимінде деректерді шифрлау алгоритмінің құрылымдық сұлбасы 7.2 суретінде келтірілген. 7.2 суретінде он бір разрядқа сол жақа циклды жылжыту операциясы $R_{11}^{\leftarrow}$ белгіленген.

Стандарттың криптографиялық түрлендірудің негізгі қадамының құрылымдың сұлбасы ($N = Қадам(N,k_j)$) 7.3 суретінде көрсетілген.



а) б)

7.2 сурет - Қарапайым ауыстыру режимінде деректерді шифрлау алгоритмінің құрылымдық сұлбасы: а) *32* циклдікі; б) бір циклдікі



7.3 сурет - Стандарттың криптографиялық түрлендірудің негізгі қадамының құрылымдың сұлбасы

7.3.3. ГОСТ 28147-89 қарапайым ауыстыру режимінде деректерді керішифрлау

Қарапайым ауыстыру режимінде деректерді керішифрлау алгоритмін жүзеге асыратын криптографиялық сұлбаның түрі шифрлау кезіндегідей (7.1 суретін қараңыз).

Қарапайым ауыстыру режимінде деректерді керішифрлау шифрлаудың алгоритмі бойынша орындалады, тек келесі өзгеріспен: жинақтауыштың мазмұндары КСҚ-дан керішифрлау кезінде келесі ретпен оқылады (шифрлауға қарсы ретпен)

$$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0,$$

$$k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0, k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0 .$$

Осыған сәйкес қарапайым ауыстыру режимінде деректерді шифрлау үшін (7.1)…(7.3) теңдіктері керішифрлау үшін келесі түрде көрсетілу мүмкін:

$$\begin{cases} a(32-j) = f(a(33-j) \dot{+} k_{(j-1)}) \oplus b(33-j), \ j = 1, 2, \ldots, 8; \\ b(32-j) = a(33-j), \ j = 1, 2, \ldots, 8, \end{cases} \tag{7.4}$$

$$\begin{cases} a(32-j) = f(a(33-j) \dot{+} k_{(32-j) \bmod 8}) \oplus b(33-j), \ j = 9, 10, \ldots, 31; \\ b(32-j) = a(33-j), \ j = 9, 10, \ldots, 31, \end{cases} \tag{7.5}$$

$$\begin{cases} b(32-j) = f(a(33-j) \dot{+} k_{32-j}) \oplus b(33-j), \ j = 32; \\ a(32-j) = a(33-j), \ j = 32. \end{cases} \tag{7.6}$$

Жұмыстың *32* циклдан кейін N_1 және N_2 жинақтауыштардың мазмұндары керішифрланған (ашық) деректер блогын құрады

$$T_O = (a_1(0), a_2(0), \ldots , a_{32}(0), b_1(0), b_2(0), \ldots , b_{32}(0)).$$

Бұл жерде N_1 жинақтауыштың күйі:

$$N_1 = a(0) = (a_{32}(0), a_{31}(0), \ldots , a_1(0)),$$

ал N_2 жинақтауыштың күйі

$$N_2 = b(0) = (b_{32}(0), b_{31}(0), \ldots , b_1(0)).$$

Тұра осылай шифрланған деректердің қалған блоктары керішифрланады.

Егер қарапайым ауыстыру режимінде *64*-биттік T_O блог деректерін шифрлау алгоритмін *A* арқылы белгілесек, онда

$$A(T_O) = A(a(0),b(0)) = (a(32), b(32)) = T_Ш, \qquad (7.7)$$

ал

$$A^{-1}(T_Ш) = A^{-1}(a(32),b(32)) = (a(0), b(0)) = T_O. \qquad (7.8)$$

Қарапайым ауыстыру режимін деректерді шифрлау үшін тек шектелген жағдайларда қолдану қажет екенін есте сақтау керек – шифр кілтін өңдеу кезінде және оны имитоқорғауды қамтамасыз етумен шифрлап байланыс арнамен жіберу немесе ЭЕМ жадысында сақтау үшін.

ГОСТ 28147-89 стандартында деректерді шифрлауды және керішифрлауды "*Цикл 32-З*" ($Ц_{32-З}$) және "*Цикл 32-Р*" ($Ц_{32-Р}$) деп белгілейді.

7.4. ГОСТ 28147-89 ГАММАЛАУ РЕЖИМІНДЕ ДЕРЕКТЕРДІ ШИФРЛАУ

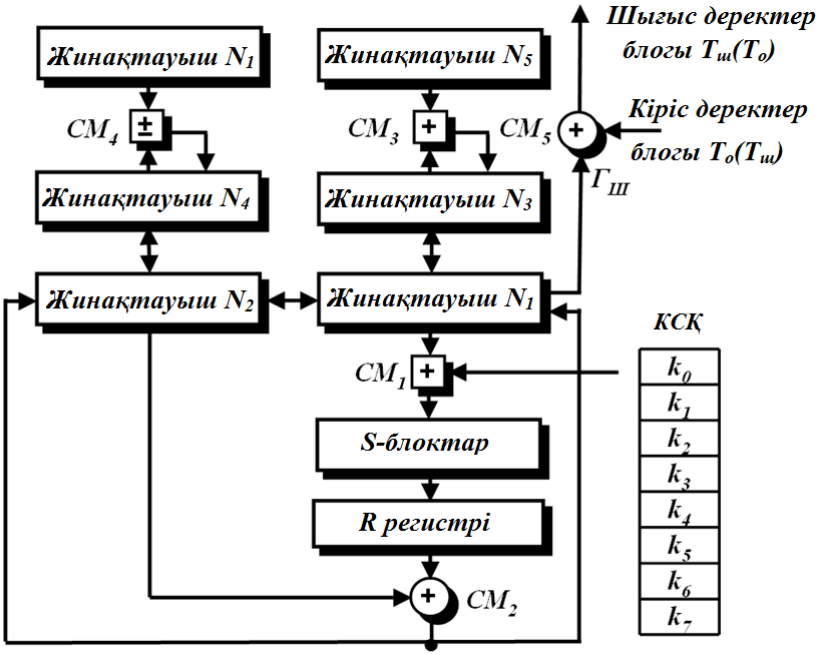
7.4.1. ГОСТ 28147-89 гаммалау режимінде деректерді шифрлаудың криптографиялық жүйесінің құрылымы

Гаммалау режимінде деректерді шифрлау алгоритмін жүзеге асыру үшін жалпы криптографиялық жүйенің барлық блоктарын қолданады (7.4 сурет).

Сұлба бойынша белгілер:

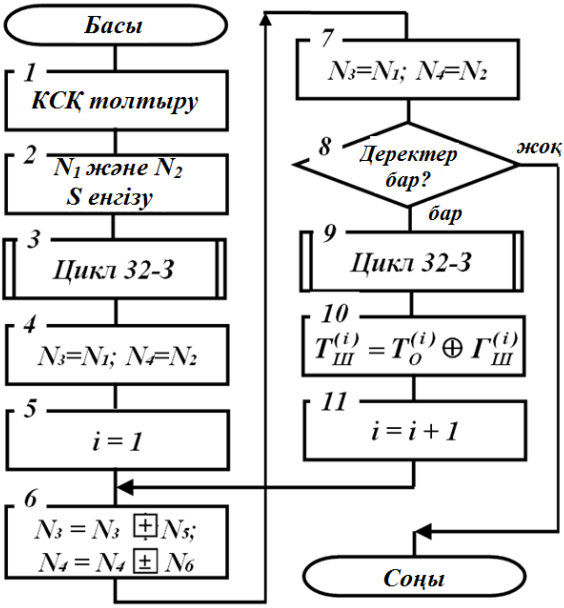
- N_1–N_6 – *32*-разрядты жинақтауыштар;
- $СМ_1$ және $СМ_3$ – 2^{32} ($\boxplus$) модулі бойынша *32*-разрядты сумматорлар;
- $СМ_2$ және $СМ_5$ – *2* ($\oplus$) модулі бойынша *32*-разрядты сумматорлар;
- $СМ_4$ – 2^{32}–*1* ($\boxplus$) модулі бойынша *32*-разрядты сумматор;
- *R* – *32*-разрядты деректерді *11* разрядқа солға (жоғары разрядтар жағына) жылжытатын циклдық жылжыту регистрі;
- *КСҚ* – *256* биттік кілтті сақтау құрылғысы, сегіз *32*-разрядты жинақтауштардан $k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$ тұрады;

- ауыстырудың *S*-блогы, сегіз ауыстыру түйінінен тұрады: S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7.



7.4 сурет – Гаммалау режимін жүзеге асыру құрылымы

Деректерді ширлаудың құрылымдық сұлбасы 7.5 суретінде көрсетілген.



7.5 сурет – Гаммалау режимінде деректерді шифрлау алгоритмі

7.4.2. ГОСТ 28147-89 гаммалау режимінде деректерді шифрлау

Ашық деректер *64*-разрядты блоктарға бөлінеді

$$T_O = (T_O^{(1)}, T_O^{(2)}, ..., T_O^{(q)}),$$

бұл жерде $T_O^{(i)}$ – *i*-ші ашық деректердің *64*-разрядты блогы (*i = 1, 2, ..., q*, ал *q* шифрланатын деректердің көлемімен анықталады).

Бұл блоктар гаммалау режимінде ретпен шифрланады $СМ_5$ сумматорында $Г_{Ш}$ шифр гаммасымен *2* модулі бойынша қосу жолымен. Шифр гаммасы *64* биттік блоктармен өндіріледі, яғни

$$Г_{Ш} = (Г_{Ш}^{(1)}, Г_{Ш}^{(2)}, ..., Г_{Ш}^{(q)}),$$

бұл жерде $Г_{Ш}^{(i)}$ – *i*-ші *64*-разрядты гамманың блогы (*i = 1, 2, ..., q*, ал *q* шифрланатын деректер көлемімен анықталады).

$T_O^{(q)}$ блогындағы екілік разрядтар саны *64* кіші болу мүмкін, сонда шифрлау үшін қолданбаған $\Gamma_{Ш}^{(q)}$ блогынан шифр гамманың бөлігін лақтырып тастайды.

Гаммалау режимінде деректерді шифрлау теңдігінің түрі

$$T_{Ш}^{(i)} = T_O^{(i)} \oplus \Gamma_{Ш}^{(i)},$$

бұл жерде $\Gamma_{Ш}^{(i)} = A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1)$, $i = 1,2,...,q$; $T_{Ш}^{(i)}$ – *i*-ші *64*-разрядтық шифрланған деректер блогы; $A(\bullet)$ – қарапайым ауыстыру режимінде шифрлау функциясы; C_1 және C_2 – *32*-разрядты екілік тұрақтылар; Y_i және Z_i – *32*-разрядты екілік тізбектер, олар $\Gamma_{Ш}^{(i)}$ гамма қалыптастырылу бойынша итерациялық анықталады.

Алдымен Y_i және Z_i алғашқы мәндері келесідей қалыптастырылады:

$$(Y_0, Z_0) = A(\tilde{S}),$$

бұл жерде $\tilde{S}$ – *64*-разрядты екілік тізбек (синхропосылка).

Содан кейін итерация бойынша

$$(Y_i, Z_i) = (Y_{i-1} \boxplus C_2, Z_{i-1} \dot{\boxplus} C_1),\ i = 1, 2, \ldots, q.$$

Гаммалау режимінде шифрлау процедурасын қарастырайық. N_6 және N_5 жинақтауыштарына алдын ала C_1 және C_2 *32*-разрядты екілік тұрақтылар жазылады, олардың мәндері (он алтылық және екілік түрде):

$C_1 = 01010101_{16} = 0000\ 0001\ 0000\ 0001\ 0000\ 0001\ 0000\ 0001_2$;

$C_2 = 01010104_{16} = 0000\ 0001\ 0000\ 0001\ 0000\ 0001\ 0000\ 0100_2$.

КСҚ кілттің *256* биты енгізіледі; N_1 және N_2 жинақтауыштарға - *64*-разрядты екілік тізбек (синхропосылка)

$$\tilde{S} = (S_1, S_2, \ldots, S_{64}),$$

бұл жерде S_i – синхропосылканың *i*-ші екілік разряды.

Синхропосылка $\tilde{S}$ N_1 және N_2 жинақтауыштардың алғашқы мазмұны болады, оны шифр гамманың q блоктарын өндіру үшін

қолданады. N_1 жинақтауыштың алғашқы мазмұны: $N_1 = (S_{32}, S_{31}, ..., S_1)$, ал N_2 жинақтауыштың алғашқы мазмұны: $N_2 = (S_{64}, S_{63}, ..., S_{33})$.

N_4 жинақтауыштың мазмұны CM_4 сумматорында 2^{32}-1 модулі бойынша N_6 жинақтауыштағы 32-разрядты C_1 тұрақтысымен қосылады. Нәтиже N_4 жазылады. N_3 жинақтауыштың мазмұны CM_3 сумматорында 2^{32} модулі бойынша N_5 жинақтауыштағы 32-разрядты C_2 тұрақтысымен қосылады. 2^{32}–1 модулі бойынша қосу қосу нәтижесінің аддитивті инверсты функциясы. Нәтиже N_3 жазылады. N_3 мазмұны N_1 көшіріледі, ал N_4 - N_2, бірақ N_3 және N_4 мазмұндары сақталады. N_1 және N_2 жинақтауыштардың мазмұндары қарапайым ауыстыру режимінде шифрланады.

N_1 және N_2 жинақтауыштардың мазмұнын шифрлау нәтижесі шифр гаммасының бірінші 64-разрядты блогын құрады

$$Г_{Ш}^{(1)} = (\gamma_1^{(1)}, \gamma_2^{(1)}, \gamma_3^{(1)}, ..., \gamma_{63}^{(1)}, \gamma_{64}^{(1)}),$$

бұл жерде γ_i^1, $i = 1, 2, ..., 64$ – гамма тізбегінің екілік разряды.

Бұл гамма блогын ($Г_{Ш}^{(1)}$) разрядтар бойынша CM_5 сумматорында бірінші 64-разрядты ашық деректер блогымен 2 модулі бойынша қосылады

$$T_O^{(1)} = (t_1^{(1)}, t_2^{(1)}, t_3^{(1)}, ..., t_{63}^{(1)}, t_{64}^{(1)}),$$

бұл жерде t_i^1, $i = 1, 2, ..., 64$ – бірінші ашық деректер блогының екілік разряды.

$T_O^{(1)}$ және $Г_{Ш}^{(1)}$ мәндерін 2 модулі бойынша қосу нәтижесінде бірінші 64-разрядты шифрланған деректер блогын алады:

$$T_{Ш}^{(1)} = T_O^{(1)} \oplus Г_{Ш}^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)}, \tau_3^{(1)}, ..., \tau_{63}^{(1)}, \tau_{64}^{(1)})$$

бұл жерде $\tau_i^{(1)} = t_i^{(1)} \oplus \gamma_i^{(1)}$, $i = 1, 2, ..., 64$ – бірінші шифрланған деректер блогының екілік разряды.

Келесі 64-разрядты шифр гамманың блогын $Г_{Ш}^{(2)}$ алу үшін N_4 мазмұны CM_4 сумматорында N_6 жинақтауыштағы C_1 тұрақтысымен 2^{32}-1 модулі бойынша қосылады. Нәтиже N_4 жазылады. N_3 мазмұны CM_3 сумматорында N_5 жинақтауыштағы C_2 тұрақтысымен 2^{32} модулі бойынша қосылады. Нәтиже N_3 жазылады. N_3 жаңа мазмұны N_1

көшіріледі, N_4 жаңа мазмұны - N_2, ал N_3 және N_4 мазмұндары сақталады. N_1 және N_2 жинақтауыштардың нәтижелері қарапайым ауыстыру режимінде шифрланады.

Шифрлау нәтижесінде алынатын N_1 және N_2 жинақтауыштардың мазмұндары шифр гамманың $Г_{Ш}^{(2)}$ екінші *64*-разрядты блогын құрады, ол CM_5 сумматорында ашық деректердің екінші блогымен разрядтармен $T_O^{(2)}$ *2* модулі бойынша қосылады:

$$T_{Ш}^{(2)} = T_O^{(2)} \oplus Г_{Ш}^{(2)} = (\tau_1^{(2)}, \tau_2^{(2)}, \tau_3^{(2)}, ..., \tau_{63}^{(2)}, \tau_{64}^{(2)}).$$

Тұра осылай шифр гамманың блоктары құрылады

$$Г_{Ш}^{(3)}, Г_{Ш}^{(4)}, ..., Г_{Ш}^{(q)}$$

және ашық деректер блоктары $T_O^{(3)}, T_O^{(4)}, ..., T_O^{(q)}$ шифрланады.

Байланыс арнасына немесе *АЕМ* жадысына синхропосылка $\tilde{S}$ және шифрланған деректер блоктары $T_{Ш}^{(1)}, T_{Ш}^{(2)}, ..., T_{Ш}^{(q)}$ жіберіледі.

7.4.3. ГОСТ 28147-89 гаммалау режимінде деректерді керішифрлау

Деректерді керішифрлау кезінде криптографиялық сұлбаның түрі шифрлау кезіндегідей болады (7.4 суретін қараңыз), ал деректерді керішифрлаудың құрылымдық сұлбасы 7.6 суретінде келтірілген.

Керішифрлау теңдігі:

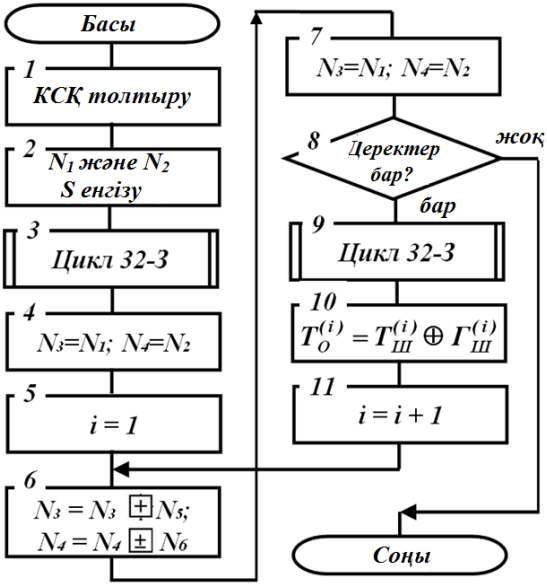
$$T_O^{(i)} = T_{Ш}^{(i)} \oplus Г_{Ш}^{(i)} = T_{Ш}^{(i)} \oplus A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1),\ i = 1, 2, ..., q.$$

Деректерді керішифрлау тек синхропосылка бар кезде мүмкін екенін айта кету қажет, ол шифрдың құпия элементі емес және *АЕМ* жадысында сақталу немесе шифрланған деректермен бірге байланыс арнасымен жиберілу мүмкін.

Керішифрлау процедурасын жүзеге асыруды қарастырамыз. *КСҚ* кілттің *256* битін енгізеді, оның көмегімен деректердің $T_O^{(1)}, T_O^{(2)}, ..., T_O^{(q)}$ шифрлауы орындалады. N_1 және N_2

жинақтауыштарға синхропосылка енгізіледі, және шифр гаммасының q блоктарын өндіру процессі орындалады

$$\Gamma_{Ш}^{(1)},\Gamma_{Ш}^{(2)},...,\Gamma_{Ш}^{(q)}.$$



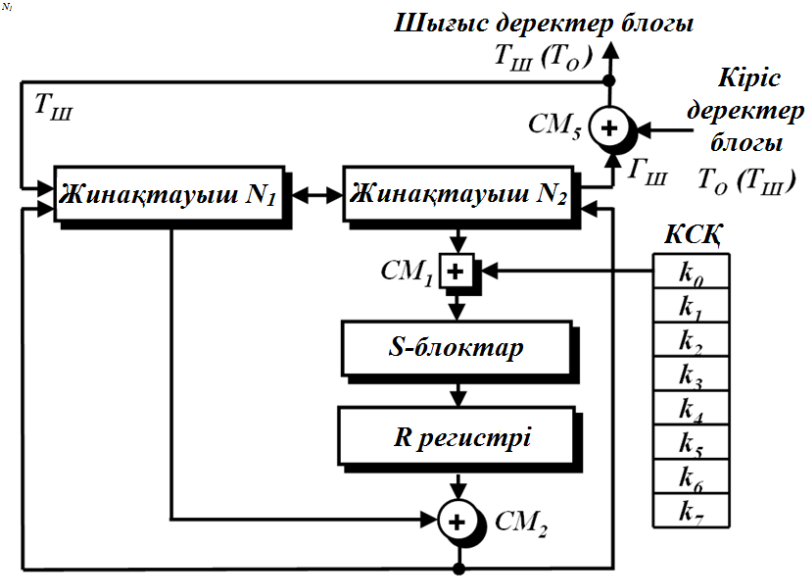
7.6 сурет - Гаммалау режимінде деректерді керішифрлау алгоритмі

Шифрланған деректер блоктары $T_{Ш}^{(1)}, T_{Ш}^{(2)},...,T_{Ш}^{(q)}$ сумматор CM_5 разрядтармен *2* модулі бойынша шифр гаммасымен $\Gamma_{Ш}^{(1)},\Gamma_{Ш}^{(2)},...,\Gamma_{Ш}^{(q)}$ қосылады. Нәтижесінде ашық деректер блоктары $T_O = (T_O^{(1)}, T_O^{(2)},..., T_O^{(q)})$ алынады, бұл жерде $T_O^{(q)}$ мазмұнында *64* разрядтан аз разряд болу мүмкін.

7.5. ГОСТ 28147-89 КЕРІ БАЙЛАНЫСЫ БАР ГАММАЛАУ РЕЖИМІНДЕ ДЕРЕКТЕРДІ ШИФРЛАУ

7.5.1. ГОСТ 28147-89 кері байланысы бар гаммалау режимінде деректерді шифрлаудың криптографиялық жүйесінің құрылымы

Кері байланысы бар гаммалау режимінде деректерді шифрлау алгоритмін жүзеге асыру үшін жалпы криптографиялық жүйенің тек бөлек блоктары қолданылады (7.7 сурет).



7.7 сурет - Кері байланысы бар гаммалау режимін жүзеге асыру сұлбасы

Сұлба бойынша белгілер:

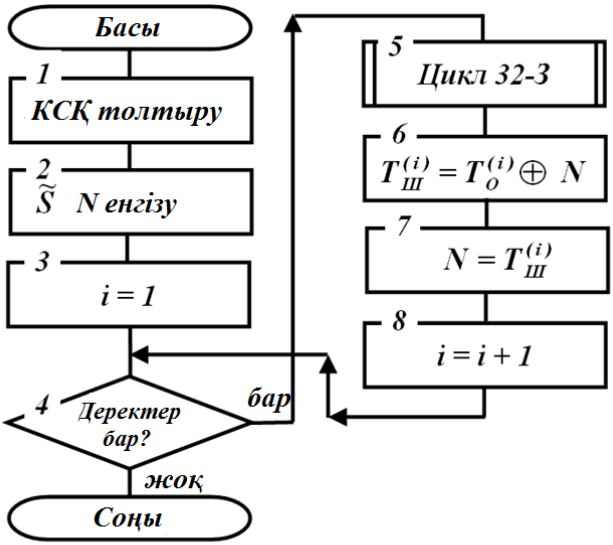
- N_1, N_2 – *32*-разрядты жинақтауыштар;
- CM_1 – 2^{32} ($\dot{+}$) модулі бойынша *32*-разрядты сумматорлар;
- CM_2 және CM_5 – *2* ($\oplus$) модулі бойынша *32*-разрядты сумматорлар;
- CM_4 – $2^{32}-1$ ($\pm$) модулі бойынша *32*-разрядты сумматор;
- *R* – *32*-разрядты деректерді *11* разрядқа солға (жоғары разрядтар жағына) жылжытатын циклдық жылжыту регистрі;

- *КСҚ – 256* биттік кілтті сақтау құрылғысы, сегіз *32*-разрядты жинақтауыштардан k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7 тұрады;

- ауыстырудың *S*-блогы, сегіз ауыстыру түйінінен тұрады: S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7 .

7.5.2. ГОСТ 28147-89 кері байланысы бар гаммалау режимінде деректерді шифрлау

Кері байланысы бар гаммалау режимінде шифрлау алгоритмін жүзеге асыратын криптографиялық сұлба 7.7 суретінде көрсетілген, ал деректерді шифрлау алгоритмінің құрылымдың сұлбасы 7.8 суретінде көрсетілген.



7.8 сурет - Кері байланысы бар гаммалау режимінде деректерді шифрлау алгоритмі

64-разрядты блоктарға бөлінген ашық деректер $T_O = (T_O^{(1)}, T_O^{(2)}, ..., T_O^{(q)})$ кері байланысы бар гаммалау режимінде *2* модулі бойынша разрядтармен шифр гаммасымен $\Gamma_{Ш}$ қосу жолмен шифрланады. Гамма $\Gamma_{Ш}$ *64* биттік блоктармен өндіріледі

$$Г_{Ш} = (Г_{Ш}^{(1)}, Г_{Ш}^{(2)}, ..., Г_{Ш}^{(q)}).$$

$T_O^{(q)}$ блогында екілік разрядтар саны *64* аз болу мүмкін, шифр гаммасы блогындағы $Г_{Ш}^{(q)}$ шифрлау үшін қолданбаған бөлігін лақтырып тастайды.

Кері байланысы бар гаммалау режимінде шифрлау теңдігі келесі:

$$T_{Ш}^{(1)} = Г_{Ш}^{(1)} \oplus T_O^{(1)} = A(\tilde{S}) \oplus T_O^{(1)};$$

$$T_{Ш}^{(i)} = A(T_{Ш}^{(i-1)}) \oplus T_O^{(i)},\ i = 2, 3, 4, ..., q,$$

бұл жерде $T_{Ш}^{(i)}$ – шифрланған деректердің *i*-ші *64*-разрядтық блогы; *A(●)* – қарапайым ауыстыру режимінде шифрлау функциясы.

Интерактивті алгоритмнің бірінші қадамында *A(●)* функциясының аргументі *64*-разрядты синхропосылка $\tilde{S}$, ал келесі қадамдарда - шифрланған деректердің алғашқы блогы $T_{Ш}^{(i-1)}$.

Кері байланысы бар гаммалау режимінде деректерді шифрлау процедурасы келесідей жүзеге асады. *КСҚ* кілттің *256* биті енгізіледі. N_1 және N_2 жинақтауыштарына *64* биттен тұратын синхропосылка $\tilde{S} = (S_1, S_2, ..., S_{64})$ енгізіледі.

N_1 және N_2 алғашқы мазмұны (синхропосылка) қарапайым ауыстыру режимінде шифрланады. N_1 және N_2 жинақтауыштарының мазмұнын шифрлау нәтижесінде алынған *64*-разрядты блок шифр гаммасын құрады

$$Г_{Ш}^{(1)} = A(\tilde{S}),$$

ол CM_5 сумматорында разрядтармен *2* модулі бойынша ашық деректердің бірінші *64*-разрядты блогымен қосылады

$$T_O^{(1)} = (t_1^{(1)}, t_2^{(1)}, \quad t_3^{(1)}, ..., t_{63}^{(1)}, t_{64}^{(1)}).$$

Нәтижесінде бірінші *64*-разрядты шифрланған деректер блогын алады:

$$T_{Ш}^{(1)} = T_O^{(1)} \oplus Г_{Ш}^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)}, \tau_3^{(1)}, ..., \tau_{63}^{(1)}, \tau_{64}^{(1)}).$$

Шифрланған деректер блогы $T_{Ш}^{(1)}$ бір уақытта шифр гамманың $Г_{Ш}^{(2)}$ екінші блогын өндіру үшін N_1 және N_2 жинақтауыштардың алғашқы күйі болып табылады, сондықтан кері байланыс бойынша $T_{Ш}^{(1)}$ көрсетілген N_1 және N_2 жинақтауыштарына жазылады.

N_1 жинақтауыштың мазмұны:

$$N_1 = (\tau_{32}^{(1)}, \tau_{31}^{(1)}, \tau_{30}^{(1)}, ..., \tau_{1}^{(1)}, \tau_{2}^{(1)}),$$

ал N_2 жинақтауыштың мазмұны:

$$N_2 = (\tau_{64}^{(1)}, \tau_{63}^{(1)}, \tau_{62}^{(1)}, ..., \tau_{34}^{(1)}, \tau_{33}^{(1)}).$$

N_1 және N_2 жинақтауыштардың мазмұны қарапайым ауыстыру режимінде шифрланады. N_1 және N_2 жинақтауыштардың мазмұнын шифрлау нәтижесінде алынған *64*-разрядты блок шифр гаммасы $Г_{Ш}^{(2)}$, ол сумматор $СМ_5$ разрядтармен *2* модулі бойынша ашық деректердің екінші блогымен $T_{О}^{(2)}$ қосылады

$$T_{Ш}^{(2)} = T_{О}^{(2)} \oplus Г_{Ш}^{(2)} = A(T_{Ш}^{(1)}) \oplus T_{О}^{(2)}).$$

Шифр гамманың $Г_{Ш}^{(i)}$ келесі блоктарының қалыптасырылуы және ашық деректердің $T_{О}^{(i)}$ сәйкес блоктарын шифрлау алдынғыдай орындалады.

Егер ашық деректердің $T_{О}^{(q)}$ сонғы *q*-ші блогының ұзындағы *64*-разрядтан кіші болса, онда шифр гамманың $Г_{Ш}^{(i)}$ тек сәйкес разрядтар қолданылады, қалғандарын лақтырып тастайды.

Байланыс арнасына немесе *АЕМ* жадысына синхропосылка $\tilde{S}$ және шифрланған деректер блоктары жіберіледі

$$T_{Ш} = (T_{Ш}^{(1)}, T_{Ш}^{(2)}, T_{Ш}^{(3)}, ..., T_{Ш}^{(q-1)}, T_{Ш}^{(q)}).$$

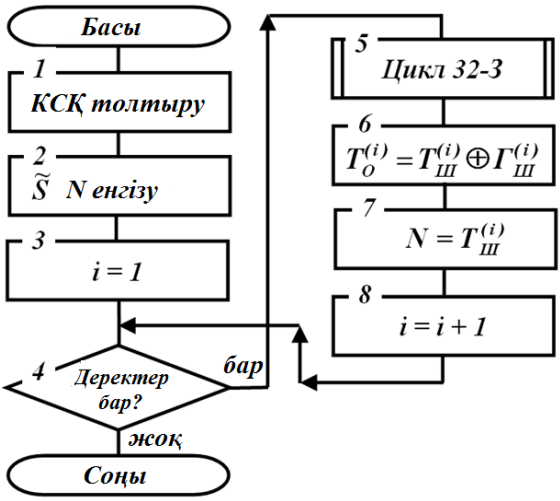
7.5.3. ГОСТ 28147-89 кері байланысы бар гаммалау режимінде деректерді керішифрлау

Кері байланысы бар гаммалау режимінде деректерді керішифрлау кезінде *ГОСТ 28147-89* криптографиялық сұлбасы шифрлау кезіндегіндей болады (7.7.суретіне қараңыз). Деректерді керішифрлау алгоритмінің құрылымдық сұлбасы 7.9 суретінде келтірілген.

Керішифрлау теңдігі:

$$T_O^{(1)} = A(\tilde{S}) \oplus T_Ш^{(1)} = Г_Ш^{(1)} \oplus T_Ш^{(1)};$$

$$T_O^{(i)} = A(T_Ш^{(i-1)}) \oplus T_Ш^{(i)} = Г_Ш^{(i)} \oplus T_Ш^{(i)},\ i = 2, 3, \ldots, q.$$



7.9 сурет - Кері байланысы бар гаммалау режимінде деректерді керішифрлау алгоритмі

Жоғарыда келтірілген қатынастардан деректерді керішифрлау тек синхропосылка бар кезде мүмкін екені көрініп тұр, ол шифрдың құпия элементі емес және *АЕМ* жадысында сақталу немесе шифрланған деректермен бірге байланыс арнасымен жиберілу мүмкін.

Керішифрлау процедурасын жүзеге асыруды қарастырамыз. *КСҚ* кілттің *256* биті енгізіледі, оның көмегімен деректерді

$T_O^{(1)}, T_O^{(2)}, ..., T_O^{(q)}$ шифрлау орындалады. N_1 және N_2 жинақтауыштарға синхропосылка $\tilde{S}$ енгізіледі. N_1 және N_2 жинақтауыштардың алғашқы мазмұны (синхропосылка $\tilde{S}$) қарапайым ауыстыру режимінде шифрланады.

Шифрлау нәтижесінде алынған N_1 және N_2 мазмұны шифр гамманың бірінші блогын құрады

$$Г_Ш^{(1)} = A(\tilde{S}),$$

ол сумматор CM_5 разрядтармен *2* модулі бойынша шифрланған деректер блогымен $T_Ш^{(1)}$ қосылады. Нәтижесінде ашық деректердің бірінші блогын аламыз

$$T_O^{(1)} = Г_Ш^{(1)} \oplus T_Ш^{(1)}.$$

Шифрланған деректер блогы $T_Ш^{(1)}$ шифр гаммасының $Г_Ш^{(2)}$ екінші блогын өндіру үшін N_1 және N_2 жинақтауыштарының алғашқы мазмұны болады

$$Г_Ш^{(2)} = A(T_Ш^{(1)}).$$

N_1 және N_2 жинақтауыштардың алынған мазмұны қарапайым ауыстыру режимінде шифрланады. Шифрлау нәтижесінде алынған блок $Г_Ш^{(2)}$ сумматор CM_5 разрядтармен *2* модулі бойынша шифрланған деректердің екінші блогымен $T_Ш^{(2)}$ қосылады. Нәтижесінде ашық деректердің екінші блогын алады $T_O^{(2)}$. Тұра солай N_1 және N_2 тізбекті шифрланған деректер блоктарын жазады

$$T_Ш^{(2)}, T_Ш^{(3)}, T_Ш^{(4)}, ..., T_Ш^{(q-1)}, T_Ш^{(q)},$$

олардан қарапайым ауыстыру режимінде шифр гамманың блоктары $Г_Ш^{(3)}, Г_Ш^{(4)}, ..., Г_Ш^{(q)}$ өндіріледі.

Шифр гамманың блоктары CM_5 сумматорында разрядтармен *2* модулі бойынша шифрланған деректер блоктарымен қосылады

$$T_Ш^{(3)}, T_Ш^{(4)}, ..., T_Ш^{(q-1)}, T_Ш^{(q)}.$$

Нәтижесінде ашық деректер блоктарын $T_O^{(3)}$, $T_O^{(4)}$,…, $T_O^{(q)}$ алады, және де ашық деректердің соңғы блогында $T_O^{(q)}$ *64* разрядтардан аз болу мүмкін.

7.6. ГОСТ 28147-89 ИМИТОЕНДІРМЕНІ ҚҰРУ РЕЖИМІНДЕ ДЕРЕКТЕРДІ КРИПТОГРАФИЯЛЫҚ ТҮРЛЕНДІРУ

Имитоендірме – *L* биттен тұратын блок, ал ашық деректерден кілтті қолданып арнайы ережемен өндіріледі және содан кейін шифрланған деректерге қосылады олардың *имитоқорғауын* қамтамасыз ету үшін.

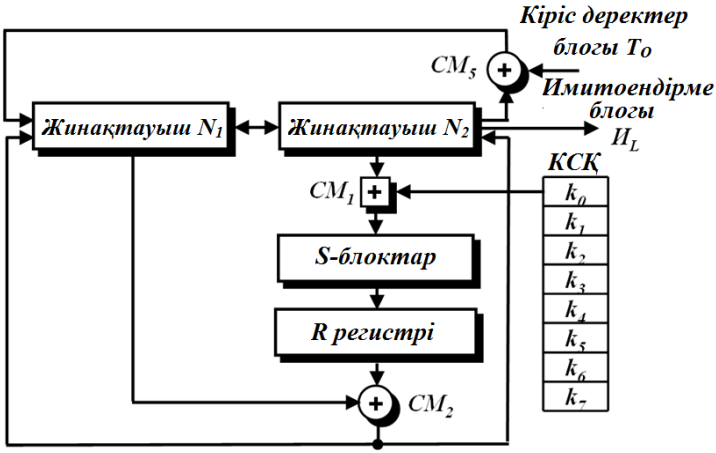
Имитоқорғау – бұл шифрланған байланыс жүйесін жалған деректерді тықпалаудан қорғау.

ГОСТ 28147-89 стандартында имитоендірмені өндіру процессі анықталады, ол деректерді шифрлаудың кез келген режимі үшін бірдей. Имитоендірме $И_L$ ашық деректер блогынан барлық хабарды шифрлаудың алдында немесе блоктармен шифрлаумен қатар өндіріледі. Имитоендірмені өндіруге қатысатын ашық деректердің бірінші блоктарында қызметті ақпарат болу мүмкін (мысалы, мекенжайлық бөлім, уақыт, синхропосылка) және олар шифрланбау мүмкін.

L параметрінің мәні (имитоендірмеде екілік разрядтар саны) жалған кедергілерді тықпалау ықтималдығы $P_{ЛП} = 2^{-L}$ тең екенін есепке алып, криптографиялық талаптарға сәйкес анықталады.

Имитоендірмені өндіру режимінде криптографиялық түрлендіру алгоритмін жүзеге асыру үшін жалпы криптожүйе блоктарының тек бөліктері қолданылады (7.10 сурет).

Бұл режим жалпықабылданған мағынада шифрлау режимі болмайды. Имитоендірмені өндіру режимінде жұмыс істеуде барлық ашық және кілттік деректерден тәуелді кейбір қосымша блок құрылады. Берілген блок шифрланған деректерді кездейсоқ немесе әтейі өзгертілмегенін тексеру үшін қолданылады. Бұл гаммалау режимінде шифрлау үшін ерекше маңызды, себебі қасқой кілтті білмей нақты биттерді өзгерте алады; бірақ егер жіберілетін деректерде артық ақпарат болмаса басқа режимдерде ықтималды өзгерістерді анықтауға болмайды.



7.10 сурет - Имитоендірмені өндіру режимін жүзеге асыру сұлбасы

Имитоендірме тек егер 64 биттен ашық деректер блоктары екіден аз болмаса ғана өндіріледі.

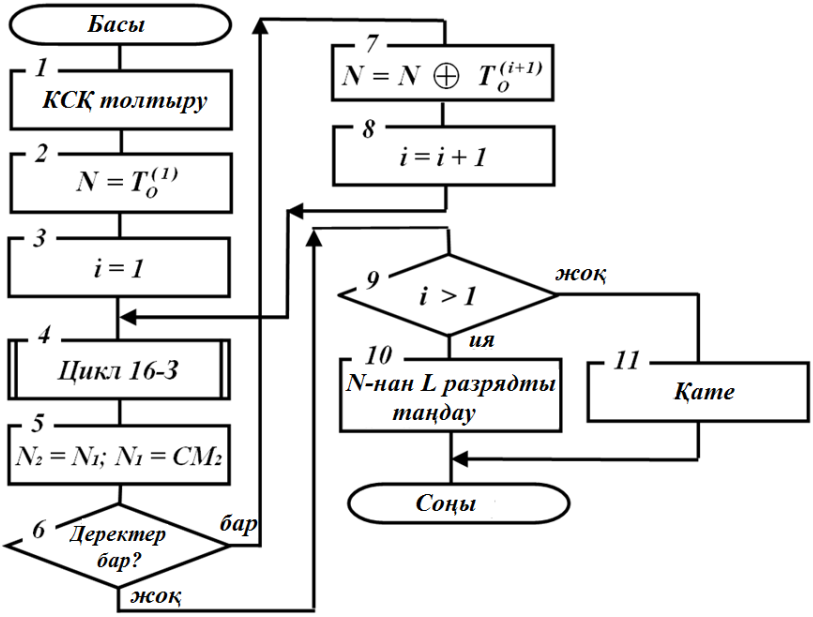
ГОСТ 28147-89 имтоендірмені өндіру режимінде деректерді криптографиялық түрлендіру алгоритмінің құрылымдық сұлбасы 7.11 суретінде келтірілген.

Имитоендірмені өндіру үшін ашық деректер *64*-разрядты блоктар тізбегі түрінде беріледі $T_O^{(i)}$, $i = 1,2,...,q$.

Кілттік сақтау құрылғысына (*КСҚ*) кілттің *К 256* битын енгізеді.

Ашық деректердің бірінші блогы $T_O^{(1)}$ N_1 және N_2 регистрлеріне жазылады, содан кейін қарапайым ауыстыру режимінде шифрлаудың бірінші *16* циклына сәйкес $\tilde{A}(\bullet)$ түрлендіруі орындалады.

16 циклдан кейін алынған *64*-разрядты сан $\tilde{A}(T_O^{(1)})$ *2* модулі бойынша ашық деректердің екінші блогымен $T_O^{(2)}$ қосылады. Қосу нәтижесі ($\tilde{A}(T_O^{(1)}) \oplus T_O^{(2)}$) қайтадан $\tilde{A}(\bullet)$түрлендіріледі.



7.11 сурет - *ГОСТ 28147-89* имтоендірмені өндіру режимінде деректерді криптографиялық түрлендіру алгоритмінің құрылымдық сұлбасы

Алынған *64*-разрядты санды $\tilde{A}(\tilde{A}(T_O^{(1)}) \oplus T_O^{(2)}))$ *2* модулі бойынша ашық деректердің үшінші блогымен $T_O^{(3)}$ қосады және қайтадан $\tilde{A}(\bullet)$ түрлендіріп, *64*-разрядты санды

$$\tilde{A}(\tilde{A}(\tilde{A}(T_O^{(1)}) \oplus T_O^{(2)}) \oplus T_O^{(3)}),$$

алады және ары қарай.

Сонғы блок $T_O^{(q)}$ (қажет болса толық *64*-разрядты блокқа дейін нөлдермен толықтырған) *2* модулі бойынша *q–1* қадамында орындалған есептеу нәтижесімен қосылады, содан кейін қарапайым ауыстыру режимінде $\tilde{A}(\bullet)$ түрлендіруін қолданып шифрланады.

Алынған *64*-разрядты саннан ұзындығы *L* үлкен (сол жақ) бит (бірден аз емес және *64* биттен үлкен емес) үздіксіз $И_L$ (имитоендірмені) кесіндісін таңдайды

$$И_L = (a_{32-L+1}^{(q)}(16), a_{32-L+2}^{(q)}(16), ..., a_{32}^{(q)}(16)),$$

бұл жерде $a_i^{(q)}(16)$ – $\tilde{A}(\bullet)$ түрлендірудің он алтыншы соңғы циклынан кейін алынған *64*- разрядты санның *i*-ші биті $32–L+1 \le i \le 32$.

Имитоендірме $И_P$ байланыс арнасында немесе АЕМ жадысына шифрланған деректердің соңында жіберіледі, яғни

$$T_Ш^{(1)}, T_Ш^{(2)}, T_Ш^{(3)}, ..., T_Ш^{(q-1)}, T_Ш^{(q)}, И_L.$$

Қабылдаушыға келген шифрланған деректер

$$T_Ш^{(1)}, T_Ш^{(2)}, T_Ш^{(3)}, ..., T_Ш^{(q-1)}, T_Ш^{(q)}$$

керішифрланады, және алынған ашық деректер блоктарынан $T_O^{(1)}, T_O^{(2)}, T_O^{(3)}, ..., T_O^{(q-1)}, T_O^{(q)}$ тұра солай имитоендірме $И^*_L$ өндіріледі. Бұл имитоендірме $И^*_L$ байланыс арнасынан немесе АЕМ жадысынан шифрланған деректермен бірге алынған имитоендірмемен $И_L$ салыстырылады. Егер имитоендірмелер сәйкес келмесе, онда керішифрланған кезде алынған ашық деректер блоктары $T_O^{(1)}, T_O^{(2)}, T_O^{(3)}, ..., T_O^{(q-1)}, T_O^{(q)}$ жалған деп есептеледі.

Имитоендірмені өндіру жоғарыда көрсетілген жұмыс істеу режимдерін қолданумен шифрлауға қатар орындалу мүмкін екенін есте сақтау қажет.

7.7. ГОСТ 28147-89 ҚАУІПСІЗДІГІ

7.7.1. ГОСТ 28147-89 криптографиялық беріктілігі

Нақты жүзеге асыруда қолдану үшін криптографиялық алгоритмді таңдаудың анықтау факторының бірі оның *беріктілігі,* яғни қасқойдың оны ашуға қарсы тұру беріктілігі. Шифрдың беріктілігі тұралы сұрақ оны жақын қарастырғанда екі өзара байланысқан сұрақтарға алып келеді:

- берілген шифрды жалпы ашуға болама;
- егер болса, тәжірибелік түрде қаншалықты қиын болады.

Жалпы ашуға мүмкіндігі жоқ шифрлар *абсолютті* немесе *теориялық берік* деп аталады. Бұндай шифрлардың болуы *Шеннон*

теоремасымен дәлелденген [45], бірақ бұл беріктіліктің бағасы әр шифрланатын хабар үшін хабардан кіші емес кілтті қолдану. Барлық жағдайларда, бірқатар ерекше жағдайлардан басқа, бұл баға шамадан тыс, сондықтан тәжірибеде беріктілігі абсолютті емес шифрларды қолданады. Сонымен, жиі қолданатын шифрлау сұлбаларын шегі бар уақыт, нақтырақ, әрқайсысы сандарға орындалатын кейбір операция болатын, шегі бар қадам ішінде ашылу мүмкін. Олар үшін ең маңызды мәні бар түсінік тәжірибелік беріктілік, оларды ашудың тәжрібелік қиындығын көрсететін. Бұл қиындықтың сандық өлшемі шифрды ашу үшін орындалатын қарапайым арифметикалық және логикалық операциялар саны болу мүмкін, яғни берілген шифрланған хабар үшін берілген шамадан кіші емес ықтималдықпен сәйкес ашық мәтінді анықтау. Сонымен бірге криптографиялық талдаушыда керішифрланатын массивке қоса ашық деректер блоктары және оларға сәйкес шифрланған деректер болу мүмкін немесе ол таңдаған кез келген ашық деректер үшін сәйкес келетін шифрланған деректерді алу мүмкіндігі бар болу мүмкін. Жоғарыда келтірілген және басқа көптеген шарттарға тәуелді криптографиялық талдаудың бөлек түрлері бар.

Барлық заманауи криптографиялық жүйелер *Керкгоффс* принципі бойынша құрылған, яғни шифрланған деректердің құпиялығы кілттің құпиялығымен анықталады. Бұл келесіні білдіреді: егер криптографиялық талдаушыға алгоритмнің өзі белгілі болса да, ол хабарды керішифрлай алмайды егер оның сәйкес кілті болмаса. Егер шифрды ашудың барлық кілттердің кеңістігі бойынша толық талдау тәсілінен басқа тиімді тәсілі болмаса, онда шифр жақсы жобаланған деп есептеледі. *ГОСТ 28147-89* бұл принципке сәйкес келуі ықтималды, себебі жылдар бойынша интенсивті зерттеудің нәтижесінде оның криптографиялық талдаудың бір де бір нәтижелі тәсілі ұсынылған жоқ. Беріктілік бойынша ол шифрлаудың американдық стандарт *DES* көптеген дәрежеге жоғары.

ГОСТ 28147-89 256-биттік кілт қолданылады және кілттік кеңістік көлемі 2^{256}. Кәзіргі уақытта бар немесе алыс емес болашақтағы уақытта ешбір электронды құрылғыда көптеген жүз жылдарынан аз уақытта кілтті анықтауға болмайды. Бүгінгі күндері бұл кілт ұзындығы симметриялық криптографиялық алгоритмдер үшін стандарт болып қалды, сондықтан *АҚШ* жаңа шифрлау стандарты да оны қолдайды. Заманауи есептеу құралдарының

мүмкіндіктеріне қарай алдынғы американдық стандарт *DES 56* биттік кілт көлемімен және кілттік кеңістігі 2^{56} жеткілікті берік емес. Бұл XX ғасырдың 90-шы жылдары *DES* толық талдау арқылы бірнеше рет ашумен дәлелденген болатын. Осыған қоса, *DES* арнайы криптографиялық талдау тәсілдеріне осал болыпты, мысалы дифференциалды және сызықты. Осыған байланысты *DES* тәжірибелік емес зерттеу немесе ғылыми қызығушылықты тұдырады. 1998 жылы оның криптографиялық әлсіздігі ресми түрде мойындалды: *АҚШ* Ұлттық стандарттар институты *DES* бойынша үшеселі шифрлауды қолдануды ұсынды. Ал 2001 жылының соңында *АҚШ* жаңа шифрлау стандарты *AES* ресми бекітілді, ол басқа принциптер негізінде құрылған және алдынғы шифрдың кемшіліктерінен бос [38].

7.7.2. ГОСТ 28147-89 сәулеті бойынша ескертулер

ГОСТ 28147-89 шифрлау стандарты бір принциптерге негізделген шифрлар жанұясының мүшесі екені жалпы белгілі. Ең белгілі оның "туысы" американдық шифрлау стандарты *DES.* Осы барлық шифрлар *ГОСТ 28147-89* сияқты үш деңгейлі алгоритмдерге жатады. Негізінде арқашан кейбір *"негізгі қадам"* жатады, оның негізінде *"базалық циклдар"* ұқсас құрылады, және олардың негізінде тәжірибелік шифрлау және имитоендірмені өндіру процедуралары құрылады. Сонымен, бұл жанұя шифрларының спецификасы оның *"негізгі қадамында"*, дәлірек айтқанда оның бөлігінде.

ГОСТ 28147-89 ұқсас шифрлар үшін *"криптотүрлендірудің негізгі қадамдары"* алгоритмдері бірдей құрылған, және бұл сәулет оны бірінші ұсынған адамның атымен *Фейстельдің теңгерімділінген желісі* (*Balanced Feistel Network*) деп аталады [4, 29]. Деректерді бір циклда немесе, оны атауға қабылданған, раундта түрлендіру сұлбасы 7.3 суретінде көрсетілген.

Негізгі қадам кірісіне жұп көлемі бар блок беріледі, оның үлкен және кіші жартылары бір бірінен бөлек өңделеді. Түрлендіру барысында кіші жартысы үлкен жартысының орнына жіберіледі, ал үлкені биттік *xor* операциясы көмегімен кейбір функцияны орындау нәтижесімен құрамдыстырылып кіші жартысының орнына жіберіледі. Аргумент ретінде блоктың кіші жартысын және кілттік ақпарат элементін (X) қабылдайтын бұл функция шифрдың

мазмұндық бөлігі болады және *шифрлау функциясы деп* аталады. Әр түрлі ойлармен шифрланатын блокты екі тең жартыға бөлу тиімді болып шықты: $|N_1| = |N_2|$ – осы факт сәулет атында *"теңгерімділінген"* сөзін бейнелейді. Шифрлайтын теңгерімділінбеген желілерді де қолданады, бірақ теңгерімділінгенге қарағанда өте сирек. Осыған қоса, шифрдың беріктілігі ойымен, кілттік элемент көлемі блоктың жарты көлемінен кіші болмау керектігін талап етеді: $|N_i| \leq |X|$ *ГОСТ 28147-89* барлық үш көлем *32* битке тең.

Егер барлық айтқанды *ГОСТ 28147-89* алгоритмінің негізгі қадам сұлбасына қолданса, онда алгоримнің 1, 2, 3 блоктары (7.2, б суретін қараңыз) оның шифрлау функциясын есептеуді анықтайтыны, ал 4 және 5 блоктары кіріс блогының мазмұнына және шифрлау функциясы мәніне қарап негізгі қадамның шығыс блогын қалыптастыруын анықтайтыны айқын болады.

Алдында *DES* және *ГОСТ 28147-89* беріктілік бойынша салыстыру өткізілді, енді оларды функционалдық мазмұны және жүзеге асыру ыңғайлығы бойынша салыстырамыз. *ГОСТ 28147-89* шифрлау циклдарында негізгі қадам *32* рет қайталанады, *DES 16* рет. Бірақ *ГОСТ 28147-89* шифрлау функциясының өзі *DES* қарағанда қарапайым, *DES* өте көп жүйесіз алмастырулар бар. Бұл операциялар заманауы мамандырылмаған процессорларда өте тиімсіз жүзеге асырылады. *ГОСТ 28147-89* ондай операциялар жоқ, сондықтан ол бағдарламалық жүзеге асыру үшін ыңғайлы.

Intel x86 платформасы үшін *DES* бір де бір жүзеге асыруы *ГОСТ 28147-89* жүзеге асыруының өнімділігінің жартысына да жетпейді, циклы екі есе қысқа болсада. Жоғарыда айтқанның бәрі *ГОСТ 28147-89* құрушылары *DES* кемшіліктері мен артықшылықтарын есепке алғанын, сонымен қатар криптографиялық талдаудың ағымдағы және болашақ мүмкіндіктерін бағалағанын көрсетеді. Шифрлардың жүзеге асыруларының жылдамдығын салыстыру кезінде *DES* негіз ретінде алу актуалды емес. *АҚШ* жаңа шифрлау стандартының тиімділігі бойынша жағдайы жақсы: *AES* шифрында *ГОСТ 28147-89* кілт көлемі *256* бит бірдей болса да, ол шамамен *14%* жылдамырақ жұмыс істейді - егер элементарлық операциялар санымен салыстырса. Осыған қоса, *ГОСТ 28147-89* қатарлауға болмайды, ал *AES* ондай мүмкіндіктер көп. Кейбір сәулеттерде *AES* бұл артықшылығы аз болу мүмкін, басқаларда – көп. Мысалы, *Intel Pentium* процессорінде ол *28%* дейін жетеді. Толығырақ [1, 7, 13, 38, 47] табуға болады.

7.7.3. ГОСТ 28147-89 кілттік ақпарат сапасына қойылатын талаптар және кілт көздері

Шифр кілттері және ауыстыру кестелерінің барлығы емес шифрдың максималды беріктілігін қамтамасыз ететін. Әр шифрлау алгоритмі үшін кілттік ақпаратты бағалаудың өзінің критерийлері болады. Мысалы, *DES* алгоритмінде *әлсіз кілттер* бар екені белгілі, оларды қолдану кезінде ашық және шифрланған деректер арасындағы байланыс жеткілікті қалқаланбайды, және шифрды салыстырмалы оңай ашуға болады.

ГОСТ 28147-89 кілттер және кестелер сапасы туралы критерийлер сұрағына толық жауапты егер алуға болса да, ол тек алгоритм құрушылардан. Сәйкес деректер ашық басылымға шығарылмаған. Бірақ, орнатылған ереже бойынша грифы бар ақпаратты шифрлау үшін, уәкілетті ұйымнан алынған кілттік деректер қолдану қажет. Бұл кілттік деректерді *«әлсіздікке»* тексеретін әдістемелердің бар екендігін білдіреді. Егер *ГОСТ 28147-89* әлсіз кілттердің бар болуы – сұрақ болса, онда әлсіз ауыстыру бұындары бар болғаны күмән тұдырмайды. *«Таптаурын»* ауыстыру кестесі, оны қолданғанда кез келген мән өзіне ауысатын болса, өте әлсіз болатыны айқын. Оны қолданса, кілт қандай болса да шифрды тез бұзуға болады.

Жоғарыда көрсетілгендей, кілттік ақпаратты бағалаудың критерийлері қолжеткізілімді емес, бірақ олар туралы кейбір жалпы ойларды айтуға болады.

Кілт

Кілт статистикалық тәуелсіз, *0* және *1* мәндерін тең ықтималдықпен қабылдайтын биттер массиві болу керек. Бұл кезде кілттің кейбір нақты мәндері *әлсіз* болатынын аластатуға болмайды, яғни оларды қолдану кезінде шифр берілген беріктілік деңгейін қамтамасыз етпеу мүмкін. Бірақ, барлық мүмкін кілттер арасында сондай мәндердің көлемі өте аз. Шифрдың интенсивті зерттеулері белгілі (яғни *ФАПСИ* ұсынған) ауыстыру кестелері үшін бір де бір сондай кілтті тапқан емес. Сондықтан, кейбір шынында кездейсоқ сандар датчигі көмегімен өндірілген кілттер, сапалы болады, олардың ықтималдығы бірден өте аз шамаға айырмашылығы болады. Егер кілттер псевдокездейсоқ сандар генераторымен

өндірілсе, онда қолданатын генератор жоғарыда көрсетілген статистикалық сипаттамаларды қамтамасыз ету керек, және соған қоса, жоғары криптографиялық беріктілігі болу керек - *ГОСТ 28147-89* өзінен төмен емес. Басқа сөзбен, генератор өндіретін тізбек элементтерінің жоқ мүшелерін анықтау есебінің қиындығы шифрды ашудан төмен болмау керек. Осыған қоса, жаман статистикалық сипаттамалары бар кілттерді анықтау үшін әртүрлі статистикалық критерийлер қолдану мүмкін. Тәжірибеде екі критерий жеткілікті: *0* және *1* мәндері арасында кілт биттерінің теңықтималды тарауын тексеру үшін *Пирсон критерийі* (“*хи квадрат*”), ал кілт биттерінің тәуелсіздігін тексеру үшін – *сериялар критерийі* қолданылады.

Кілттерді өндіру үшін ең жақсысы кездейсоқ сандардың аппаратты датчиктерін қолдану болады, бірақ бұл экономикалық ойлар бойынша кейбір кезде қолайсыз. Көлемі үлкен емес кілттік ақпаратты генерациялау кезінде осындай датчик орнына тәжірибеде кең қолданатыны *«электронды рулетка» әдісі* , бұл жерде кездейсоқ биттердің кезекті тізбегін өндіру оператордың компьютер пернетақтасында пернені басу уақытынан тәуелді болады. Бұл сұлбада кездейсоқ деректер көзі компьютер пайдаланушысы, нақтырақ – оның реакциясының уақыттық сипаттамалары. Пернені бір басуда кездейсоқ деректердің бірнеше биттері өндірілу мүмкін, сондықтан кілттік ақпаратты өндірудің жалпы жылдамдығы – секундына бірнеше бит. Бұл әдіс кілттердің үлкен массивтерін алу үшін жарамайды.

Егер көлемі үлкен кілттік ақпарат массивін өндіру қажет болса, онда әртүрлі бағдарламалық псевдокездейсоқ сандар датчиктерін қолдану кең тараған. Осындай датчиктен криптографиялық беріктіліктің жоғары көрсеткіштері талап етілгеннең кейін, оның ретінде шифрдың өзінің гамма генераторын қолдану қарастырылған – шифрмен өндірілген гамманы қажетті көлемі бар *«бөліктерге кесеміз»* - *ГОСТ 28147-89* үшін – *32* байттан. Бұндай тәсіл үшін бізге *мастер-кілт* қажет, оны жоғарыда көрсетілген *«электронды рулетка»* әдісімен алуға болады, ал оның көмегімен, шифрды гамма генераторы режимінде қолданып, қажетті көлемді кілттік ақпарат массивін аламыз. Бұл кілтті өндірудің екі тәсілі – *алгоритмдік* және *қолмен жасалған,* - бір бірін толықтырып жұмыс істейді. Ақпаратты криптографиялық қорғаудың *азбюджеттік* жүйелерінде кілттерді генерациялау сұлбалары жиі осы принцип бойынша құрылады.

Ауыстырулар кестесі

Ауыстырулар кестесі ұзақуақытты кілттік элемент, яғни *бөлек (сеанстық) кілттке* қарағанда ұзақ уақыт жұмыс істейді. Ол бір криптографиялық қорғау жүйесі ішінде барлық шифрлау түйіндері үшін жалпы болып табылады. Ауыстырулар кестесінің конфиденциалдығы бұзылған жағдайда да шифр беріктілігі өте жоғары болады және ұйғарынды шектен төмендемейді. Сондықтан жеке жағжайларда кестені құпия сақтаудың қажеттілігі жоқ, және *ГОСТ 28147-89* көптеген коммерциялық қолдануларда солай істеледі. Басқа жақтан, ауыстырулар кестесі барлық шифрдың беріктілігін қамтамасыз ету үшін сынды маңызды элемент. Лайықсыз кестені таңдау шифрды белгілі криптографиялық талдау әдістерімен ашуға алып келу мүмкін. Ауыстыру түйіндерін өндіру критерийлері - *жеті мөр астындағы құпия* және *ФАПСИ* жақын болашақта онымен бөліспейтіні анық. Сонғы қорытында, берілген нақты ауыстырулар кестесі жақсы немесе жаман екенін айту үшін үлкен жұмыс көлемін жүргізу қажет - көптеген мың адам- және машине- сағат. Бір рет таңдалған және қолданылатын кесте тек, егер оны қолданумен шифр криптографиялық талдаудың кейбір түріне осал болса ғана ауыстырылады. Сондықтан, қарапайым пайдаланушы үшін ең жақсы таңдау белгілі болған кестенің біреуін алу. Мысалы, хэш-функцияға стандарттан, ол *"орталықбанктік"* деп аталады (ауыстыру түйіндері *RFC 4357* құжатымен анықталған). Бұл кестелер туралы мәліметтерді ашық басылымда және интернеттен де алуға болады.

Жеңіл жолдармен жүрмейтіндерге, төменде сапалы кестелерді алудың жалпы сұлбасы келтірілген:

1. Кейбір әдістеме көмегімен кепілденген сызықтық емес сипаттамалармен ауыстырулардың сегіз түйінінен тұратын комплект қалыптастыру. Бұндай әдістеменің бірнешесі бар, олардың бірі - *бент-функцияларды* қолдану [44, 47].

2. Қарапайым сапа критерийлерін орындалуын тексеру, мысалы, *DES* ауыстыру түйіндері үшін жарияланған. Осы туралы тағы бірнеше ойлар: әр түйін төрт логикалық аргументтен төрт логикалық функциямен сипатталу мүмкін. Егер *минималды пішінде* (яғни өрнектің мүмкінді минималды ұзындығымен) жазылған бұл функциялар жеткілікті қиын емес болса, онда бұндай ауыстыру түйінін қолданбайды. Сонымен қатар, ауыстырулар кестесі ішінде

бөлек функциялар бір бірінен жеткілікті дәрежеде айырмашылығы болу қажет. Бұл кезеңде көптеген сапасыз кестелер елеп алып тасталады.

3. Таңдалған кестелермен шифр үшін криптографиялық талдаудың әртүрлі түрлеріне сәйкес келетін, әртүрлі раунд үлгілерін құру қажет, және сәйкес *бейінді* сипаттамаларды өлшеу қажет. Мысалы, сызықты криптографиялық талдау үшін шифрлау рауындының сызықты статистикалық аналогы құрылады және бейінді сипаттама есептеледі - сызықтық емес көрсеткіші. Егер ол жеткілікті емес болса, ауыстырулар кестесін қабылдамайды.

4. Соңында, алдынғы тармақ нәтижелерін қолданып, таңдалған кестемен шифрды мұқият зерттеу қажет (барлық белгілі әдістерімен криптографиялық талдау әрекеттерімен). Осы кезең ең қиын және көлемді болады. Бірақ, егер ол сапалы жасалса, онда ықтималдықтың жоғары деңгейімен таңдалған кестелермен шифр арнайы қазметтермен де ашылмайтынын айтуға болады.

Бірақ, оңайрақ істеуге болады. Шифрда раундтар саны көп болған сайын, бір раунд беріктілі сипаттамаларының барлық шифрдың беріктілігіне әсері азайады. *ГОСТ 28147-89 – 32* раунд, бұл осындай сәулеті бар барлық шифрларға қарағанда көп. Сондықтан тұрмыстық және коммерциялық қолданулардың көбі үшін *0* ден *15-ке* дейін сандардың тәуелсіз кездей соқ сияқты ауыстырулар түйіндерін алу жеткілікті болады. Бұл тәжірибелік түрде жүзеге асырылу мүмкін, мысалы, берілген диапазоннан мәннің біреуі әр картаға бекітілген карталарды араластыру арқылы.

Ауыстырулар кестелері туралы тағы бір қызықты фактіні айта кету қажет. Шифрлау "*32-З*" және керішифрлау "*32-Р*" циклдарының қайтымды болу үшін ауыстырулар түіндері *0* ден *15-ке* дейін сандардың алмастырулары болғандығын талап етпейді. Егер ауыстыру түйінінде қайталанатын элементтер болса және сондай түйінмен анықталатын, ауыстыру қайтымды емес болса да, барлығы жұмыс істейді - бірақ бұл жағдайда шифр беріктілігі төмендейді. Ол үшін, түйіндерінде қайталанатын мәндері бар сондай *"толымсыз"* ауыстырулар кестесін қолданумен, деректер блогын алдымен шифрлау, содан кейін керішифрлау жеткілікті.

ГОСТ 28147-89 қолдану

Өте жиі деректерді криптографиялық қорғау жүйелерінде қолдану үшін *ГОСТ 28147-89* жүзеге асырудың жылдамдығынан үлкен және криптографиялық беріктілігі соншалықты жоғары емес алгоритм талап етіледі. Осындай есептердің типтік мысалы әртүрді электронды биржалық сауда жүйелері, олар реалды уақытта сауда сессияларын басқарады. Бұл жерде қолданылған шифрлау алгоритмдерден жүйенің жедел деректерін сессия барысында (берілген ұсынымдар, қол қойылған келісім-шарттар туралы деректер) керішифрлау мүмкін еместігі талап етіледі, ол аяқталғаннаң кейін қасқой үшін бұл деректер пайдасыз. Басқа сөзбен, кепілденген беріктілік тек бірнеше сағатқа қажет - сауда сессияның типтік уақыты сондай. Толық *ГОСТ 28147-89* бұл жағдайда қолдану жөнсіз екені түсінікті.

Бұл және осыған ұқсас жағдайларда шифрлаудың жылдамдығын жоғарлату үшін не істеу қажет? Жауап үстінде жатыр - базалық циклдарда негізгі қадамдардың (раундтардың) көлемі аз шифрдың түрін қолдану. Шифрлау раундтарын қаншаға азайтамыз, соншаға жылдамдығы жоғарлайды. Көрсетілген өзгерісті екі жолмен жасауға болады: кілт ұзындығын қысқарту және кілттің *"қарау циклдар"* санын қасқартумен. Шифрлаудың базалық циклдарында негізгі қадам саны $N=n\cdot m$ тең, бұл жерде n – кілтте *32*-биттік элементтер саны, m – кілттік элементтерді қолданудың цикл саны, стандартта $n = 8$, $m = 4$. Осы санның кез келгенін қысқартуға болады, бірақ қарапайым нұсқасы - кілттің ұзындығын, оны қолдану сұлбасына тиіспей қысқарту.

Жұмыс жылдамдығын жылдамдату үшін төлем шифрдың беріктілігінің төмендеуі. Негізгі қиындық келесіде: осы төмендеудің шамасын нақты бағалау жеткілікті қиын. Оны жасаудың жалғыз мүмкінді тәсілі - кемітілгін циклдарымен криптографиялық түрлендірудің шифр нұсқаларын *"толық бағдарлама бойынша"* зерттеу. Бұл біріншіден, *ГОСТ 28147-89* құрушылары білетін жабық ақпаратты қолдануды талап етеді, және екіншіден, өте жұмыс көлемді екені түсінікті. Сондықтан бағаны тек жалпы заңдылыққа қарап беріп көрейік.

Шифрдың *"қарқынды"* бұзу әдістеріне беріктілігіне келсек, яғни *"қатқыл күш"* шабуылына, онда бұл жерде бәри түсінікті: *64* биттік кілт бұл шабуыл түріне қол жеткізілімді шегінде тұр, *96* бит және

одан ұзын кілті бар шифр (кілтте *32*-биттік элементтердің тұтас саны болу қажет) оған қарсы тұра алады. Бірнеше жыл бұрын *АҚШ* шифрлау стандарты *DES* кілтті талдау жолымен бірнеше рет бұзылған болатын: алдымен оны *Интернет* жаһандық желісі негізінде ұйымдастырылған есептеу желісі бұзды, содан кейін - мамандырылған, яғни соған арнайы құрылған есептеу машинасы бұзды. *ГОСТ 28147-89* стандарты нұсқасы заманауи процессорда бағдарламалық жүзеге асыруда *DES* төрт есе жылдам жұмыс істейді деп қабылдайық. Онда *8*-раундты *"кемітілген ГОСТ" DES 16* есе жылдам жұмыс істейді. Сонымен қатар, *DES* бұзу уақытынан есептеу техника өнімділігі *Мур заңына* сәйкес төрт есе өскенін қабалдайық. Қорытынында келесіні аламыз: кәзір сегіз циклы бар *"кемітілген ГОСТ"* үшін бір *64*-биттік кілтті тексеру, кезінде *DES* бір кілтін тексеруден *64* есе жылдам орындалады. Сонымен, *"қатқыл күш"* шабуылының жұмыс көлемдігі бойынша *ГОСТ 28147-89* сондай нұсқасының *DES* алдындағы артықшылығы $2^{64-56} = 2^8 = 256$ ден $256/64 = 4$ ретке дейін қысқарады. Бұл өте бұлдыр айырмашылық, жоқ деп айтуға болады.

ГОСТ 28147-89 әлсізденген түрлерінің криптографиялық талдаудың *қарқынды* тәсілдеріне беріктілігін бағалау қиын. Бірақ жалпы заңдылықты бұл жерде табуға болады. Бүгінгі күннің ең күшті криптографиялық талдау түрлерінің *"бейінді"* сипаттамалары шифрлаудың раундтар санынан экспонентті тәуелді. Сызықты криптографиялық талдау үшін бұл сызықтық сипаттамасы *L* болады:

$$L = C \cdot e^{-\alpha \cdot r},$$

бұл жерде *C* және α – тұрақтылар; *r* – раундтар саны.

Осындай тәуелдік дифференциалды криптографиялық талдау үшінде бар. Өзінің *физикалық мәні* бойынша бұндай сипаттамалардың бәрі - ықтималдық. Әдетте криптографиялық талдау үшін қажетті алғашқы деректер көлемі және жұмыс көлемдігі сондай сипаттамаларға қайта пропорционалды. Бұл жерден жұмыс көлемдік көрсеткіштері шифрлаудың негізгі қадам санының өсумен бірге экспонетті өсетіні шығады. Сондықтан раундтар санының бірнеше есе төмендеу кезінде ең белгілі талдау түрлерінің жұмыс көлемділігі алғашқы көлемнен бұл дәреженің түбірі сияқты өзгереді (шамамен). Бұл беріктіліктің өте үлкен төмендеуі.

Басқа жақтан, *ГОСТ 28147-89* төземділіктің үлкен қорымен жобаланған және бүгінгі күнге криптографиялық талдаудың барлық белгілі түріне берік, соның ішінде дифференциалды және сызықты. Сызықты криптографиялық талдауға қолданса бұл оны сәтті өткізу үшін мүмкін болатыннан көп *"ашық блок - шифрланған блок"* жұптарын талап етеді, яғни 2^{64} көп. Жоғарыда айтылғанды есепке алып, *16*-раундтық *ГОСТ 28147-89* сәтті сызықты криптографиялық талдау үшін $\sqrt{2^{64}} = 2^{32}$ блок немесе 2^{35} байт немесе *32* Гбайттан аз емес деректер қажет, ал *8*-раундтық үшін - $\sqrt[4]{2^{64}} = 2^{16}$ блок немесе 2^{19} байт немесе *0,5* Мбайттан аз емес.

Жоғарыда айтылғанның бәрінен, *ГОСТ 28147-89* кемітілген нұсқаларының сипаттамаларын жалпылайтын қортындылар 7.2 кестеде келтірілген.

Сонғы екі нұсқа *12* және *8* раундтармен уақыт бойынша шектелген қорғауды қамтамасыз етеді. Оларды қолдану тек қорғалатын деректердің қысқауақытты (шамамен бірнеше сағат) құпиялығы талап етілетін жерлерде ақталады. Бұл әлсіз шифрлар нұсқаларын қолданудың мүмкінді аймағы - электронды биржалық сауда жүйелерінің *UDP*-трафигын жабу. Бұл жағдайда деректердің әр пакеті (*datagram, UDP* аббревиатурасынан ортанғы *D*) бөлек *64*-биттік кілтте шифрланады, ал кілттің өзі сеанстық кілтте (оның жұмыс аймағы - екі компьютер арасындағы бір байланыс сеансы) шифрланады және деректермен бірге жіберіледі.

Ескерту. Кілттің көлемі, оны қолданудың сұлбасын өзгертпеген кезде, кілтті алдымен *3* рет түзу бағытта, содан кейін қайта бағытта қарауды қарастырады. Жылдамдық индексі стандартты *32*-раундтық нұсқаға қарағанда шифрлаудың жылдамдығы қанша есеге жоғарлайтынын көрсетеді.

Қарастырылатын сұрақтың қайта беті де бар. Егер шифрлау жылдамдығы сынды емес, ал беріктілікке талаптар қатаң болса не болады? *ГОСТ 28147-89* беріктілігін екі жолмен жоғарлатуға болады: оларды *"экстенсивті"* және *"қарқынды"* деп атаймыз.

Біріншісі - шифрлау раундатар санын қарапайым үлкейту. Бірақ *ГОСТ 28147-89* стандарты бұлсызда қажетті беріктілікті қамтамасыз етеді. Көрсетілген жол шифрдың беріктілігін реалды жоғарлатуға мүмкіндік береді: егер бұрын криптографиялық талдау қарапайым мүмкін емес болса, онда енді ол мүлдем мүмкін емес.

7.2 кесте

ГОСТ 28147-89 кемітілген нұсқаларының жалпылама сипаттамалары

Раундтар саны	*Кілт көлемі, бит*	*Жылдамдық индексі*	*Шифрдың ықтималды сипаттамалары(өте өрескел баға)*
24	*192*	*1,33*	*Криптографиялық талдаудың көп түріне берік, немесе беріктік шегінде. Криптографиялық талдаудың тәжірибелік жүзеге асыруы алғашқы деректерге және жұмыс көлеміне жоғары талаптарға байланысты мүмкін емес.*
16	*128*	*2*	*Криптографиялық талдаудың кейбір түріне теориялық түрде берік емес, тәжірибелік жүзеге асыруы көптеген жағдайларда алғашқы деректерге және жұмыс көлеміне жоғары талаптарға байланысты қиындатылған.*
12	*95*	*2,67*	*Криптографиялық талдаудың кейбір түріне түрде берік емес, бірақ қысқа уақытқа үлкен емес деректер көлемдерінің (ондаған-жүздеген Кбайтқа дейін) құпиялығын қамтамасыз етуге жарайды.*
8	*64*	*4*	*Криптографиялық талдаудың кейбір түріне түрде берік емес, бірақ қысқа уақытқа үлкен емес деректер көлемдерінің (ондаған Кбайтқа дейін) құпиялығын қамтамасыз етуге жарайды.*

Шифрлаудың негізгі қадамдар құрылымын және санын өзгертпей шифрдың беріктілігін жоғарлатуға бола ма деген сұрақ өте қызық. Бұл сұрақтың жауабы оңды. *ГОСТ 28147-89* түрлендірудің негізгі қадамында *4* ті *4* битке ауыстыруды орындау қарастырылған, ал тәжірибеде барлық бағдарламалық жүзеге асырулар ауыстыруды байттар бойынша орындайды, яғни *8* ді *8* битке - тиімділік ойымен жасалады. Егер осындай ауыстыруды 8-

биттік сияқтығы ретінде жобаласа, онда бір бір раундтың сипаттамаларын жақсартамыз.

Біріншіден, "*диффуздық*" сипаттама жоғарлайды немесе *"тасқындық"* көрсеткіші - алғашқы деректер және/немесе кілттің бір биты нәтиженің көп битіне әсер етеді.

Екіншіден, көлемі бойынша үлкен ауыстыру түйіндері үшін төменірек дифференциалды және сызықты сипаттамаларды алуға болады, сонымен шифрдың сондай аттары бар криптографиялық талдау түрлерін жүргізу мүмкіндіктерін төмендетіп. Бұл әсіресе *ГОСТ 28147-89* кемітілген циклдары үшін актуалды, ал *8*- және *12*-раундтық нұсқалар үшін қажет. Раундар санын азайтудан беріктілікті төмендетуден есесін қайтарады. Бұл әдісті қолдануды қиындататын - осындай *үлкейтілген* ауыстыру түйіндерін құрастыру. Ірі түйіндерді кішкентайларға қарағанда құрастыру өте қиын.

7.7.4. *ГОСТ 28147-89 стандартты емес қолдану*

ГОСТ 28147-89 криптографиялық алгоритмдерінің негізгі тағайыны - деректерді шифрлау және имитоқорғау екені шартсыз. Бірақ оларға ақпаратты қорғаумен байланысты басқа да қолдануларды табуға болады. Қысқаша олар туралы айтып кетейік:

1. Гаммалау режимінде шифрлау үшін *ГОСТ 28147-89* криптографиялық гамманы өндіруді қарастырады - жақсы статистикалық сипаттамалары, жоғары криптографиялық беріктілігі бар биттер тізбегі. Ары қарай бұл гамма ашық деректерді түрлендіру үшін қолданылады, нәтижесінде шифрланған деректерді алынады. Бірақ, бұл криптографиялық гамманың қолданудың жалғыз мүмкінгі емес. Оны өндіру алгоритмі - өте жақсы сипаттамалары бар *псевдокездейсоқ сандар тізбегінің генераторы*. Әрине, осындай *ПКСТГ* өндірілетін тізбектің тек статистикалық сипаттамаларын алу, ал криптографиялық беріктілік қажет емес жерлерде қолдану дұрыс емес - ондай жағдайларға тиімдірек генераторлары бар. Бірақ, ақпаратты қорғаумен байланысты, әр түрлі қолданулар үшін бұндай көз қажетті болады.

Жоғарыда көрсетілгендей, гамманы кілттерді өндіру үшін *"шикізат"* ретінде қолдануға болады. Ол үшін қажетті ұзындығы бар гамма кесіндісін алу қажет - *32* байт. Бұндай тәсілмен кілттерді қажет болғанда құруға болады және оларды сақтау қажет емес -

егер ондай кілт қайта қажет болса оны оңай қайта өндіруге болады. Ол үшін тек ол қандай кілтте өндірілгенін, қандай синхропосылка қолданылды және өндірілген гамманың қай байтынан басталғанын еске түсіру қажет. Қолданылған кілттен басқа ақпарат құпия емес. Осындай тәсіл қиын және бұтақталған кілттер жүйесін тек бір *мастер-кілтті* қолданып оңай бақылауға мүмкіндік береді.

Алдынғымен бірдей, гамманы алғашқы *"шикізат"* ретінде құпиясөздерді өндіруге қолдануға болады. Бұл жерде сұрақ туындау мүмкін: оларды не үшін генерациялау қажет, қажет кезде оларды жеңіл ойлап табуға болады. Бұндай тәсілдеменің халсіздігі компьютерлік желілерде болған инциденттер сериясымен көрсетілген, ең ірісі 1988 жылдың қарашасында бір тәулікке интернетті істен шығарған *"Моррис құрты"* деп аталды. Компьютерге қаскүнемдік бағдарламаның кірудің тәсілінің бірі құпиясөзді іріктеу болды: бағдарлама жүйеге кіру үшін өзінің ішкі бірнеше жүзден тұратын тізімінен құпиясөздерді тізбекті қолданып әрекеттер жасады, көптеген жағдайларда бұл орын алды. Адамның құпиясөздерді ойлап табу қиялы кедей болып шықты. Сондықтан, қауіпсіздікке тиісті назар аударатын ұйымдарда, құпиясөздерді қауіпсіздік бойынша әкімші генерациялайды және пайдаланушыларға таратады. Кілттерді өндіруге қарағанда, құпиясөздерді өндіру қиынрақ, себебі *"шикі"* екілік гамманы қарапайым бөлшектерге *"кесумен"* бірге символдық түрге түрлендіру қажет. Сонымен қоса, құпиясөзде алфавиттің барлық символдарының тең ықтималдығын қамтамасыз ету үшін, мүмкін, кейбір бөлек мәндерді, алып тастауға қажет болады.

Криптографиялық гамманы қолданудың тағы бір тәсілі - магнитті тасымалдауыштарда деректерді кепілді өшіру. Қайта жазу кезінде магниттік тасымалдауышта алдынғы деректердің іздері қалады, оларды сәйкес сараптама қалпына келтіре алады. Сол іздерді жою үшін қайта жазуды көп рет орындау қажет. Егер ондай процедура кезінде кездейсоқ немесе псевдокездейсоқ деректерді қолданса, онда өшірілген ақпаратты қалпына келтіретін сарапшыларға ол белгісіз болады, сондықтан ақпаратты тасымалдауышқа аз рет қайта жазу қажет болады. Бұл жерде шифр гаммасы орынды болады.

2. Криптографиялық гамма және криптографиялық түрлендіру де шифрлаумен байланысты емес қажеттіліктерге қолдануы мүмкін.

*ГОСТ*ты қолданудың нұсқаларының бірі - деректер массивтері үшін имитоендірмені өндіру екені белгілі. Бірақ кез келген блокты шифр негізінде, соның ішінде *ГОСТ* негізінде, біржақты хэш-функцияны есептеу сұлбасын құру жеткілікті оңай, оны әдебиетте *MDC* деп атайды. Әр түрлі көздерде *MDC өзгерістерді/әрекеттерді табу коды* (*Modification/Manipulation Detection Code*) немесе *хабар дайджесты* (*Message Digest Code*) деп түсінеді.

MDC құпия кілттен тәуелді емес имитоендірменің аналогы ретінде имитоқорғау жүйелерінде қолдану мүмкін. Сонымен қатар, *MDC электронды цифрлық қолтаңба* (*ЭЦП*) сұлбаларында кеңінен қолданады. Ыңғайлық үшін бұндай сұлбаларының көбі тіркелген көлемі бар деректер блогына қол қою үшін құрастырылған. *ГОСТ 28147-89* стандарты негізінде Ресй Федерациясының біржақты хэш-функцияны *ГОСТ Р 34.11-94* есептеу бойынша стандарт құрылған.

Бақылау сұрақтары және тапсырмалар

1. *ГОСТ 28147-89* деректерді криптографиялық түрлендіру алгоритмін қандай мақсаттарға қолдануға болады?
2. *ГОСТ 28147-89* симметриялық шифрлау алгоритмінің негізгі параметрлері қандай?
3. *ГОСТ 28147-89* блокты шифрлау алгоритмінде қандай операциялар қолданылады?
4. Деректерді шифрлаудың және керішифрлаудың сонғы циклының (*32*-ші) басқалардан қандай айырмашылығы бар?
5. *ГОСТ 28147-89* алгоритмінде ауыстырулар кестесі не үшін қолданылады?
6. *ГОСТ 28147-89* шифрлау алгоритмінің *DES* алгоритмінен негізгі айырмашылықтары қандай?
7. *ГОСТ 28147-89* стандартты алгоритмдерін қолдану шартында құпия кілттен басқа хабарды керішифрлау үшін қандай ақпарат керек?
8. *ГОСТ 28147-89* криптографиялық түрлендіру алгоритмі көмегімен деректерді шифрлау қандай режимде орындалу мүмкін?
9. *ГОСТ 28147-89* алгоритмінде гаммалау және кері байланысы бар гаммалау режимдерінің айырмашылықтары қандай?
10. Имитоендірме деген не? Имитоендірме қандай мақсатпен қолдану мүмкін?

11. *ГОСТ 28147-89* алгоритмінің N_1 регистрінде деректер бар, олардың он алтылық санақ жүйесіндеге түрі: N_1 – *191a2ab8*, ал N_2 – *434665b2*. Шифрлау үшін кілттің он алтылық жүйедегі түрі: *eb8a7159 7ce5d63d 4ac1d6e0 bafe4731 a3deb025 8bb389ac 10d3b61a e9ac340f*. Шифрлау циклы – *25*. Циклды аяқтаған соң N_1 және N_2 не болады? Ауыстыру блогы ретінде 7.1 кестесін қолдану қажет.

12. *ГОСТ 28147-89* алгоритмінің N_1 регистрінде деректер бар, олардың он алтылық санақ жүйесіндеге түрі: N_1 – *434665b2*, а N_2 – *191a2ab8*. Шифрлау үшін кілттің он алтылық жүйедегі түрі: *7ce5d63d 4ac1d6e0 bafe4731 a3deb025 8bb389ac 9ea2923b 9a62e045 e9ac340f*. Шифрлау циклы – *9*. Циклды аяқтаған соң N_1 және N_2 не болады? Ауыстыру блогы ретінде 7.1 кестесін қолдану қажет.

13. *ГОСТ 28147-89* алгоритмінің N_1 регистрінде деректер бар, олардың он алтылық санақ жүйесіндеге түрі: N_1 – *10d3b61a*, а N_2 — *eb8a7159*. Шифрлау үшін кілттің он алтылық жүйедегі түрі: *e9ac340f a3deb025 8bb389ac 9ea2923b 9a62e045 bafe4731 7ce5d63d 4ac1d6e0*. Шифрлау циклы – *32*. Циклды аяқтаған соң N_1 және N_2 не болады? Ауыстыру блогы ретінде 7.1 кестесін қолдану қажет.

14. *ГОСТ 28147-89* алгоритмінің N_1 регистрінде деректер бар, олардың он алтылық санақ жүйесіндеге түрі: N_1 – *0d3b61a5*, а N_2 — *eb8a7159*. Шифрлау үшін кілттің он алтылық жүйедегі түрі: *e9ac340f 9a62e045 10d3b61a 8bb389ac 9ea2923b 7ce5d63d eb8a7159 bafe4731*. Шифрлау циклы – *32*. Циклды аяқтаған соң N_1 және N_2 не болады? Ауыстыру блогы ретінде 7.1 кестесін қолдану қажет.

15. *ГОСТ 28147-89* алгоритмінде жалған кедергілерді тықпалау ықтималдығы $P_{ЛП} = 2{,}4 \cdot 10^{-10}$ төмен болмау керек. Имитоендірменің ұзындығын биттермен анықтаңыз.

16. *ГОСТ 28147-89* алгоритмінде *24* бит ұзындығы бар имитоендірме қалыптастырылады. Жалған кедергілерді тықпалау ықтималдығын анықтаңыз.

8 тарау

RIJNDAEL БЛОКТЫ СИММЕТРИЯЛЫҚ КРИПТОГРАФИЯЛЫҚ АЛГОРИТМ ЖӘНЕ AES СТАНДАРТЫ

8.1. AES СТАНДАРТЫН ҚҰРУ ТАРИХЫ

1997 жылы АҚШ-тың стандарттар мен технологиялар Ұлттық институты (*NIST*) 1974 жылдан бастап келе жатқан әлемдегі ең көп тараған криптографиялық алгоритм *DES* алгоритмін ауыстыруға үкімет деңгейінде маңызды ақпаратты жабу үшін криптографиялық қорғаудың жаңа стандартын қабылдау бойынша бағдарламаның басталуы жөнінде жариялады [38]. *DES* көптеген параметрлері бойынша: ең бастысы – беріктілігін санамағанда, кілттің ұзындығы, қазіргі заманғы процессорларда жүзеге асыру ыңғайлылығы, жылдамдығы және басқа параметрлері бойынша ескірген болып саналады. 23 жыл бойы қарқынды криптографиялық талдау кезінде бұл шифрды ашудың кілттік кеңістігі бойынша толық іріктеудің тиімділігімен ерекшеленетін әдістері табылмады.

Кандидаттарға талап төмендегідей болды:

- криптографиялық алгоритм ашық жарияланған болу керек;
- криптографиялық алгоритм *128, 192* және *256* биттік кілттер өлшемдеріне рұқсат ететін симметриялық блоктық шифр болу қажет;
- криптографиялық алгоритм аппараттық та, бағдарламалық жүзеге асырылу үшін де арналуы қажет;
- криптографиялық алгоритм кез келген өнімде ашық қолдану үшін қолжетімді болу қажет, демек патенттелген болмау қажет, басқа жағдайда патенттік құқықтар күшін жою қажет;

- криптографиялық алгоритм келесі параметрлер бойынша зерттелу қажет: беріктік, құны, иілгіштігі, *smart*-карталарда жүзеге асырылуы.

Беріктік. Бұл алгоритмді бағалаудағы ең маңызды критерий. Бағаланды: шифрдың криптографиялық талдаудың түрлі әдістеріне қарсы тұру қабілеттілігі; басқа кандидаттармен салыстырғанда статистикалық қауіпсіздік және салыстырмалы қорғалғандық.

Құны. NIST негізгі мақсаттарының бірін ескере отырып, маңызды критерий – *AES*-тің жетімділігі және кең қолдану аймағы. Құны түрлі платформалардағы (ең алдымен жылдамдыққа) есептеу тиімділігіне, бағдарламалық және аппараттық жүзеге асырудың ыңғайлығына, жадының төмен талаптарына, қарапайымдылыққа (қарапайым алгоритмдерді жүзеге асыру оңай, олар талдау үшін айқынырақ) тәуелді болады.

Иілгіштік. Иілгіштік алгоритмнің келісілген минимумнан (*128* бит) үлкен кілттерді өңдеу қабілеттілігін, түрлі ортада орындау тиімділігі мен сенімділігін, басқа криптографиялық функцияларды: аралас шифрлау, хештеу және басқаларды жүзеге асыру мүмкіндігін қамтиды.

Басқаша айтқанда, *AES* тәжірибелік жүзеге асыру (ең алдымен кілттерді қалыптастыру және шифрлау жылдамдығы) тұрғысынан қарағанда тиімдірек болу қажет, *TripleDES*-ке қарағанда төзімділіктің үлкен қоры болуы қажет, сонымен бірге оған беріктілікте жол бермеу керек.

Smart-карталарда жүзеге асырылуы. Болашақта *AES* қолданылуының маңызды саласы – s*mart*-карталар, бұл жерде басты проблема болып қолжетімді жадының шағын көлемі саналады. *NIST* арзан карталар 256 байт *RAM* (есептеленетін деректер үшін) және *2000 ROM* (алгоритмдер мен тұрақтыларды сақтау үшін) болады деген жорамалдан шықты. Раундық кілттерді қалыптастырудың негізгі екі әдісі бар:

- криптографиялық алгоритмнің алғашқы жұмыс кезеңінде есептеу және жадыда сақтау;
- "*орындау кезінде*" раундық кілттерді есептеу.

Екінші нұсқа *RAM* шығындарын азайтады және сондықтан криптографиялық алгоритмде бұл жердей мүмкіндіктің болуы оның күмәнсіз артықшылығы болып табылады.

Ашық байқауға *Австралия, Бельгия, Ұлыбритания, Германия, Израиль, Канада, Коста-Рика, Норвегия, АҚШ, Франция, Оңтүстік*

Корея және *Жапония* сияқты әлемнің 12 елінің криптографтары құрастырған 15 алгоритм қабылданды.

Байкаудың финалына келесі алгоритмдер шықты: *Mars*, *Twofish* және *RC6* (*США*), *Rijndael* (*Бельгия*), *Serpent* (*Ұлыбритания, Израиль, Норвегия*). *Twofish* өз құрылымы бойынша *Фейстелдің* классикалық шифры болып табылады; *MARS* және *RC6*-ны *Фейстелдің* өзгертілген шифрларына жатқызуға болады, оларда шифрланатын деректер мен құпия кілтке тәуелді өзгеретін позиция санына сөздің биттерін циклдық *«бұраудың»* жаңа аз зерттелген операциясы қолданылады; *Rijndael* және *Serpent* классикалық *SP*-желілері болып табылады. *Mars* және *Twofish* құрылымы ең күрделі, *Rijndael* және *RC6* – ең қарапайым.

Финалисттер *NIST* құжатында берілген бір сызба бойынша сипатталады. Алдымен алгоритмнің анықталған *«әлсіздіктері»* (егер олар болса), содан артықшылықтары, соңында кемшіліктері сипатталады.

Mars байқауға *IBM* фирмасынан қатысты, шифрдың авторларының бірі *Д. Копперсмит* (ағыл. *Don Coppersmith*), ол *DES* өңдеуінің қатысушысы. Алгоритмде қорғауда әлсіздік табылмады.

Жетістіктері:

- қорғалғандығының жоғары деңгейі;
- көбейту және циклдық жылжыту операцияларын ерекше қолдайтын *32*-разрядты платформаларда жоғары тиімділік;
- *256* биттен артық кілт өлшемін потенциалды қолдайды.

Кемшіліктер:

- алгоритмнің күрделілігі оның сенімділігін талдауды қиындатады;
- қажетті операцияларысыз платформаларда тиімділіктің төмендеуі;
- уақытша талдау мен қуатты талдаудан қорғау күрделілігі.

RC6-ны *RSA Lab* фирмасы ұсынған,оның авторларының бірі *Р. Ривест*. Алгоритмде қорғауда әлсіздік табылмады.

Артықшылықтар:

- көбейту және циклдық жылжыту операцияларын ерекше қолдайтын *32*-разрядты платформаларда жоғары тиімділік;
- алгоритмнің қарапайым құрылымы оның сенімділігін талдауды жайдақтатады;
- жақсы зерттелген негізін қалаушының болуы – *RC5*;
- кілтті қалыптастырудың тез процедурасы;

- *256* биттен артық кілт өлшемін потенциалды қолдайды;
- кілт ұзындығы мен раундтар саны айнымалы болу мүмкін.

Кемшіліктер:

- қорғалғанның салыстырмалы төмен деңгейі;
- қажетті операциялары жоқ платформаларда тиімділіктің төмендеуі;
- уақытша талдау мен қуатты талдаудан қорғау күрделілігі;
- "*орындау кезінде*" раундтық кілттерді генерациялаудың мүмкін болмауы.

Егер олардың өз шифры қабылданылмаған болса *Rijndael* байқауға көптеген қатысушылар арасынан ең үздік таңдау болып табылды. *Square* шифрында сол авторлармен негізі қаланған. Алгоритмде қорғауда әлсіздік табылмады.

Артықшылықтар:

- кез келген платформаларда жоғары тиімділік;
- қорғалғанның жоғары деңгейі;
- жадыға төмен талабының арқасында *smart-карталарда* жүзеге асыруда жақсы сәйкес келеді;
- кілтті қалыптастырудың тез процедурасы;
- нұсқаулық деңгейінде қатарластықты жақсы қолдау;
- 32 бит қадамымен кілттің түрлі ұзындығын қолдау.

Кемшіліктер:

- қуатты талдауына осал.

Serpent – Р. Андерсон, Э. Бихам және Л. Кнудсен кәсіби криптографиялық аналитиктердің өңдеуі. Қазіргі барлық белгілі шабуылдарға қарсы тұра алатын шифр құрып, өңдеушілер одан кейін оның раундтарының санын екі есе көбейтті. Алгоритмде қорғауда әлсіздік табылмады.

Артықшылықтар:

- қорғалғандықтың жоғары деңгейі;
- жадыға төмен талабының арқасында *smart-карталарда* жүзеге асыруға жақсы сәйкес келеді.

Кемшіліктер:

- финалистер арасында ең баяу алгоритм;
- қуатты талдауға осал.

Twofish кеңінен тараған *Blowfish* шифрында негізделген; өңдеудің авторларының бірі – *Б. Шнайер*. Шифрдың басты ерекшелігі – құпия кілтінен тәуелді ауыстыру кестелерінің бар болуы. Алгоритмде қорғауда әлсіздіктер табылмады.

Артықшылықтар:

- қорғалғандықтың жоғары деңгейі;
- жадыға төмен талабының арқасында *smart-карталарда* жүзеге асыруға жақсы сәйкес келеді;
- кез келген платформада жоғары тиімділік, соның ішінде *Intel* және *Motorola* фирмаларының болашақ 64-разрядты сәулеттерінде;
- "*орындау кезінде*" раундық кілттерін есептеуді қолдайды;
- нұсқаулық деңгейінде қатарластыруды қолдайды;
- кілт ұзындығын 256 битке дейін қолдайды.

Кемшіліктер:

- алгоритмнің ерекшеліктері оның талдауын қиындатады;
- алгоритмнің жоғары қиындығы;
- қосу операциясын қолдану алгоритмді қуатты талдауға және уақыттық талдауға осал қылады.

2000 жылдың қазан айында байқау аяқталды. Жеңімпаз деп жүзеге асырылуының тиімділігінің, өнімділігінің, беріктілігінің және иілгіштігінің жақсы үйлесімі бар бельгия шифры *Rijndael* танылды. Оның жады көлеміне деген төмен талаптары оны ендірілген жүйелер үшін сөзсіз лайық етеді. Шифрдың авторлары *Йон Дэмен* (*Joan Daemen*) және *Винсент Раймен* (*Vincent Rijmen*), олардың тектерінің бастапқы әріптері *Rijndael* алгоритмінің атауын береді.

Осыдан кейін *NIST* Ақпаратты Өңдеудің Федералды Стандартының (*Federal Information Processing Standart – FIPS*) алдынғы нұсқасын дайындауды бастады және 2001 жылдың ақпанында оны *http://csrc.nist.gov/encryption/aes/* сайтына жариялады. *FIPS*-тың алдын ала нұсқауын ашық талдаудың 90-күндік кезеңі аралығында сынауларды ескере отырып қайта қаралды, осыдан кейін жөндеу мен бекіту үрдістері басталды. 2001 жылдың 26 қарашасында *FIPS-197* стандартының соңғы нұсқасы шифрлаудың жаңа американдық стандартын сипаттайтын *AES* шифры жарияланды. Осы құжатқа сәйкес стандарт күшіне 2002 жылдың 26 мамырынан бастап енгізілді [38].

8.2. AES МАТЕМАТИКАЛЫҚ АЛҒЫШАРТТАРЫ

Бұл бөлім материалдары *Rijndael* [38] алгоритмінің авторлық сипатталуына мен *FIPS-19* құжатының негізделген. *AES* стандарты *Rijndael* алгоритмімен сәйкес келген жерлерде соңғысы ғана

көрсетіледі. Қабылданған стандарттың *Rijndael* криптографиялық алгоритмінен барлық айырмашылықтары ерекше ескертіледі.

Бұл криптографиялық жүйе блоктық алгоритмдерге жатқызылатындықтан, *Фейстель* шифрының тікелей жалпылауы болмаса да, *DES*-пен көп ортаққа ие. Криптографиялық беріктілікті қамтамасыз ету үшін *Rijndael* алгоритмі өзіне қайталанатын раундтарды қосады, оның әрқайсысы ауыстыру, орын ауыстыру және кілтпен қосудан тұрады. Бұдан басқа, *Rijndael* күшті математикалық құрылыды қолданады: оның көп операциялары $GF(2^8)$ өрісінің арифметикасына негізделген. Дегенмен, *DES*-пен салыстырғанда, бұл алгоритмдегі шифрлау мен керішифрлау – әртүрлі процедуралар.

Rijndael алгоритмі $GF(2^8)$ шегі бар өрісінің элементі ретінде қарастырылатын байттармен жұмыс істейді. Өрістегі арифметикалық операциялар келтірілмеген полином модулі бойынша *GF(2)*-ден екілік көпмүшелерге орындайтын операцияларға сәйкес келеді.

Өріс элементтері өзінің коэффициенттерінің жолдары болып берілетін, 7-ден көп емес дәреженің көпмүшелері болып табылады. Егер байтты төмендегі түрде көрсетсе:

$$\{ a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0 \},\ a_i \in \{ 0,1 \},\ i = 0,1,\ldots,7,$$

онда өріс элементтері көпмүшемен сипатталады

$$a_7 \cdot x^7 + a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0 .$$

Мысалы, *{ 11001011 }* байтына (немесе он алтылық формада *{ cb }*) $x^7 + x^6 + x^3 + x + 1$ көпмүшесі сәйкес келеді.

Соңғы өріс элементтері үшін аддитивті және мультипликативті операциялар анықталған.

8.2.1 Аддитивті операциялар

Шегі бар өріс элементтерін қосу - операцияның мәні разряд бойынша *xor* және сондықтан $\oplus$ ретінде белгіленеді. Қосу операциясын орындау мысалы:

- көпмүше түрінде:

$$(x^6 + x^4 + x^2 + x + 1) \oplus (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2;$$

- екілік сан түрінде:

$$\{ 01010111 \} \oplus \{ 10000011 \} = \{ 11010100 \};$$

- он алтылық түрде:

$$\{ 57 \} \oplus \{ 83 \} = \{ d4 \}.$$

Шегі бар өрісте кез келген нөлдік емес элемент α үшін – α кері элементі бар, сонымен қатар $\alpha + (-\alpha) = 0$, бұл жерде нөлдік элемент – бұл $\{ 00 \}$. $GF(2^8)$-те $\alpha + \alpha = 0$ әділ, яғни әрбір нөлдік емес элемент өзінің жеке аддитивті инверсиясы болып табылады.

$GF(2^8)$-тен коэффициенттері бар екі көпмүшелерді қосу - мағынасы $GF(2^8)$ өрісінде келтірілген ұқсас мүшелермен көпмүшелердің қосу орерациясы, яғни

$$a(x) + b(x) = (a_7 \oplus b_7)\cdot x^7 + (a_6 \oplus b_6)\cdot x^6 + (a_5 \oplus b_5)\cdot x^5 +$$
$$+ (a_4 \oplus b_4)\, x^4 + (a_3 \oplus b_3)\cdot x^3 + (a_2 \oplus b_2)\cdot x^2 +$$
$$+ (a_1 \oplus b_1)\cdot x + (a_0 \oplus b_0).$$

Осылайша, екі көпмүшені қосу – мағынасы осы полиномдар коэффициенттердің $GF(2^8)$ өрісінде келтірілген ұқсас мүшелермен разрядты *xor* операциясы.

8.2.2 Мультипликативті операциялар

Шегі бар өріс элементтерін көбейту, ары қарай • ретінде белгіленеді, күрделі операция. $GF(2^8)$ өрісіндегі көбейту – бұл нәтижені сегіз дәрежедегі $p(x)$ келтірілмеген көпмүшенің модулі бойынша алумен және ұқсас мүшелерді келтірген кезде *xor* операциясы қолданумен көпмүшелерді көбейту операциясы. *Rijndael*-де $p(x) = x^8 + x^4 + x^3 + x + 1$ таңдалған, немесе он алтылық формада $1\{ 1b \}$. $1\{ 1b \}$ жазбасы *«артық»* тоғызыншы бит бар екендігін білдіреді.

$GF(2^8)$өрісін құру үшін қолданылатын $p(x) = x^8 + x^4 + x^3 + x + 1$ көпмүшесі көптеген анықтамалықтарда ескерілетін сегіз дәрежелі бірінші келтірілмеген көпмүше болып табылады. Яғни оны таңдау барынша ерікті.

Көбейту операциясына мысал:

$$\{ 57 \} \bullet \{ 83 \} = \{ c1 \},$$

себебі

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod p(x) = x^7 + x^6 + 1,$$

бұл жерде

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 =$$

$$= (x^5 + x^3) \cdot (x^8 + x^4 + x^3 + x + 1) \oplus (x7 + x^6 + 1),$$

ал $x^7 + x^6 + 1$ полиномының он алтылық көрінісі *{ c1 }* мәніне сәйкес.

Кез келген α нөлдік емес элементіне $\alpha \cdot 1 = \alpha$ әділ. $GF(2^8)$-те мультипликативті бірлік болып *{ 01 }* элементі табылады.

Көбейту операциясы басқаша сипатталуы және берілуі мүмкін. Жеті дәрежелі еркін көпмүшені

$$a_7 \cdot x^7 + a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0,$$

x-ке көбейтіп,

$$a_7 \cdot x^8 + a_6 \cdot x^7 + a_5 \cdot x^6 + a_4 \cdot x^5 + a_3 \cdot x^4 + a_2 \cdot x^3 + a_1 \cdot x^2 + a_0 \cdot$$ аламыз.

Алынған көпмүшені $p(x) = 1\,\{1b\}$ модулі бойынша келтіріп, $GF(2^8)$ шегі бар өрісінде нәтижені аламыз. Ол үшін $a_7 = 1$ кезінде жоғарыда алынған көпмүшені *p(x)* алып тастаса болды (разряд бойынша *xor* операциясын қолдану). Егер $a_7 = 0$ болса, онда нәтиже келтірілген болады. Онда *x*-ке көбейту (яғни *{ 00000010 }* екілік формада, *{ 02 }* он алтылық формада) солға жылжу болып және алдағы уақытта *1{ 1b }* көпмүшемен *xor*-ды қолдануы мүмкін.

Xtime() функциясы жоғарыда сипатталған әдіспен *x*-ке көбейту операциясын жүзеге асырсын. *Xtime()* функциясын *n* рет қолданып, x^n-ге көбейту нәтижесін алуға болады, ал *x*-тың әртүрлі дәрежелерін қосып, өрістің кез келген элементін алуға болады. Мысалы:

$$\{57\} \bullet \{13\} = \{fe\},$$

себебі

$$\{57\} \bullet \{02\} = xtime(\{57\}) = \{ae\};$$

$$\{57\} \bullet \{04\} = xtime(\{ae\}) = \{47\};$$

$$\{57\} \bullet \{08\} = xtime(\{47\}) = \{8e\};$$

$$\{57\} \bullet \{10\} = xtime(\{8e\}) = \{07\}.$$

Осыдан

$$\{57\} \bullet \{13\} = \{57\} \bullet (\{01\} \oplus \{02\} \oplus \{10\}) =$$
$$= (\{57\} \bullet \{01\}) \oplus (\{57\} \bullet \{02\}) \oplus (\{57\} \bullet \{10\}) =$$

$$= \{57\} \oplus \{ae\} \oplus \{07\} = \{fe\}.$$

Демек, $GF(2^8)$ өрісіндегі көбейту – бұл кейбір келтірілмеген көпмүшелердің модулі бойынша нәтижені алумен және ұқсас мүшелерді келтіру үшін *xor* операциясын қолданумен көбейтудің қарапайым операциясы [1].

Rijndael-дың раундтық түрлендірулері 32-разрядты сөзбен жұмыс істейді. Алгоритмде *32*-биттік сөздер $GF(2^8)$ өрісінен *3* дәрежелі көпмүшелермен теңестіріледі [38]. Теңдестірілу *«төнкергіш»* пішімында беріледі, яғни үлкен (мәндірек) бит көпмүшенің кіші коэффициентіне сәйкес келеді. Сонымен, мысалы,

$$a_0||a_1||a_2||a_3$$

сөзі

$$a(x) = a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$$

көпмүшесіне сәйкес келеді.

Rijndael алгоритмінде арифметика $m(x) = x^4 + 1$ көпмүшесінің модулі бойынша $GF(2^8)$ көпмүшелер сақинасында арифметикалық әрекеттермен сәйкес келеді. $m(x) = x^4 + 1 = (x + 1)^4$ көпмүшесі келтірілген, және сондықтан, алгоритмдегі арифметикалық іс-әрекет $GF(2^8)$ өрісінің операцияларынан ерекшеленген, сонымен қатар көбейтіндісі *0*-ге тең нөлдік емес элементтердің жұбы болады [38].

$GF(2^8)$ өрісінен коэффициенттері бар екі көпмүшені көбейту – тым күрделі амал. Екі көпмүше көбейтіледі деп алайық

$$a(x) = a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$$

және

$$b(x) = b_3 \cdot x^3 + b_2 \cdot x^2 + b_1 \cdot x + b_0.$$

Көбейту нәтижесі

$$c(x) = a(x) \otimes b(x)$$

$$c(x) = c_6 \cdot x^6 + c_5 \cdot x^5 + c_4 \cdot x^4 + c_3 \cdot x^3 + c_2 \cdot x^2 + c_1 \cdot x + c_0$$

көпмүшесі болады, бұл жерде

$$c_0 = a_0 \cdot b_0;$$

$$c_1 = a_1 \cdot b_0 \oplus a_0 \cdot b_1;$$
$$c_2 = a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2;$$

$$c_3 = a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3;$$

$$c_4 = a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3;$$

$$c_5 = a_3 \cdot b_2 \oplus a_2 \cdot b_3;$$

$$c_6 = a_3 \cdot b_3.$$

Көбейту нәтижесі 4-байттық сөзбен берілу үшін *4*-тен көп емес дәрежедегі көпмүше модулі бойынша нәтиже алу қажет. Шифр авторлары

$$m(x) = x^4 + 1$$

көпмүшесін таңдады. Ол

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4}.$$

көпмүшелісіне әділ [30].

Осылайша, екі көпмүшенің көбейту $\otimes$ нәтижесі $d(x)$

$$d(x) = a(x) \otimes b(x)$$

$m(x) = x^4 + 1$ модулі бойынша

$$d(x) = d_3 \cdot x^3 + d_2 \cdot x^2 + d_1 \cdot x + d_0,$$

көпмүшесі болады, бұл жерде

$$d_0 = a_0 \cdot b_0 \oplus a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3;$$

$$d_1 = a_1 \cdot b_0 \oplus a_0 \cdot b_1 \oplus a_3 \cdot b_2 \oplus a_2 \cdot b_3;$$

$$d_2 = a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 \oplus a_0 \cdot b_3;$$

$$d_3 = a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3.$$

Матрицалық формада төмендегідей түрде жазылу мүмкін:

$$\begin{vmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{vmatrix} = \begin{vmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{vmatrix} \cdot \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{vmatrix}.$$

$b(x) = b_3 \cdot x^3 + b_2 \cdot x^2 + b_1 \cdot x + b_0$ болсын дейік.

$GF(2^4)$ өрісінен коэффициенттері бар $x^4 + 1$ модулі бойынша $b(x)$ көпмүшесін x-ке көбейтуге, соңғы қасиеттерді ескере отырып, сөз шегінде үлкен байт жағына қарай байттардың циклдық жылжуына сәйкес келеді, өйткені

$$x \otimes b(x) = b_2 \cdot x^3 + b_1 \cdot x^2 + b_0 \cdot x + b_3.$$

8.2.3. Мультипликативті және аддитивті кері шамаларды табу операциялары

Шегі бар өрісте $a(x)$ кез келген нөлдік емес элементі үшін кері (инверстік) аддитивті элемент – $a(x)$ бар, сонымен бірге

$$a(x)+(-a(x)) \bmod p(x) = 0.$$

$GF(2^8)$ өрісінде $a(x) + a(x) = 0$ әділ, бұл жерде нөлдік элемент – бұл $\{00\}$, яғни кез келген нөлдік емес элемент өзінің аддитивті инверсиясы болып табылады.

Rijndael шифрында сонымен бірге *берілген элементке мультипликативті кері (инверстік)* элементті табу процедурасы қолданылады, яғни төмендегі теңсіздік орындалатын элементті:

$$a(x) \bullet a^{-1}(x) \bmod p(x) = 1, \qquad (8.1)$$

бұл жерде $a(x) – GF(2^8)$ өрісінің кейбір элементі, $a^{-1}(x)$ – оның мультипликативті кері элементі.

Егер ω кез келген примитивті элемент дәрежесі өрісінің нөлдік емес элементтерін қарастырса, (8.1) көбейтіндісін $GF(2^8)$ өрісінде орындау оңай. Ол үшін құрылғының құрылымын (өріс элементтерінің генераторы) анықтау қажет, ол өрістің әр нөлдік емес элементіне сәйкес примитивті элементтің дәрежесін қоюға мүмкіндік береді.

$$p(x + 1) = (x + 1)^8 + (x + 1)^4 + (x + 1)^3 + (x + 1) + 1 = \qquad (8.2)$$

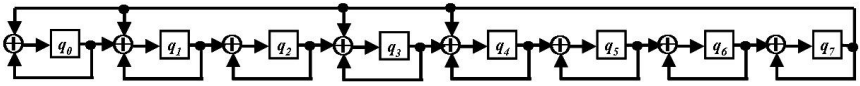
$$= (x^8 + 1) + (x^4 + 1) + (x^3 + x^2 + x + 1) + (x + 1) + 1 =$$

$$= x^8 + x^4 + x^3 + x^2 + 1 = \tilde{p}(x)$$

бар.

$\tilde{p}(x)$ полиномы примитивті, сондықтан $GF(2^8)$ өрісіндегі элементтерді көрсетудің әртүрлі формаларының арасындағы сәйкестікті егер 8.1-суретте көрсетілген құрылғы жұмысын өзгерткенде алуға болады.

00000001	*- { 01 }*	($\omega^0 = 1$)
00000011	*- { 03 }*	(ω^1)
00000101	*- { 05 }*	(ω^2)
00001111	*- { 0f }*	(ω^3)
00010001	*- { 11 }*	(ω^4)
00110011	*- { 33 }*	(ω^5)
01010101	*- { 55 }*	(ω^6)
11111111	*- { ff }*	(ω^7)
00011010	*- { 1a }*	(ω^8)
00101110	*- { 2e }*	(ω^9)
. . .	. . .	. . .
11110111	*- { f7 }*	(ω^{25})
00000010	*- { 02 }*	(ω^{26})
00000110	*- { 06 }*	(ω^{27})
. . .	. . .	. . .
11000111	*- { b4 }*	(ω^{251})
11000111	*- { c7 }*	(ω^{252})
01010010	*- { 52 }*	(ω^{253})
11110110	*- { f6 }*	(ω^{254})
00000001	*- { 01 }*	(ω^{255})



8.1 сурет - $\tilde{p}(x) = x^8 + x^4 + x^3 + x + 1$ примитивті көпмүшемен $GF(2^8)$ өрісінің элементтерінің генераторы

Мысалы,

$$(x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) \bmod p(x) =$$

$$= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod p(x) =$$

$$= x^7 + x^6 + 1$$

немесе

$$\omega^{98} \cdot \omega^{80} = \omega^{178},$$

және сондықтан

$$\{57\} \bullet \{83\} = \{c1\}.$$

$GF(2^8)$ өрісінде $\omega^{255} = 1$ әділ, сондықтан, ω^i және ω^j ($i \neq j$) нөлдік емес элементтері мультипликативті кері болады, тек қана егер $i + j = 255$ [13, 38].

Мысалы, $\omega^4 \cdot \omega^{251} = 1$, яғни $\{11\} \bullet \{b4\} = 1$ немесе

$$(x^4 + 1) \cdot (x^7 + x^5 + x^4 + x^2) \bmod p(x) = 1.$$

Бұл жерде i және j – ω^i мен ω^j мәндері бар күйге $GF(2^8)$ өрісі элементтері генераторын аударуы үшін керекті тактер саны.

$GF(2^8)$ өрісінде мультипликативті кері шамаларды анықтау төмендегі мысалды қолданумен мүмкін.

8.1-мысал. Функцияның кірісінде матрицаның кейбір байты $\{2d\}$-ге тең болсын. Полиномиалды көрініс те бұл $a(x) = x^5 + x^3 + x^2 + 1$. Келтірілмеген $p(x) = x^8 + x^4 + x^3 + x + 1$ көпмүшесімен $GF(2^8)$ өрісінде $a(x)$-тің мультипликативті кері элементін $a^{-1}(x)$ табу үшін бөлудің қалдығының максималды дәрежесі бөлгіштің максималды дәрежесінен кіші болғанша $p(x)$ көпмүшені бөлуді жүргіземіз. Егер бөлу қалдығының максималды дәрежесі 1-ден үлкен болса, бөлу жалғасады, бірақ бөлінгіш ретінде бөлгіш, ал бөлгіш ретінде алдыңғы бөлудің қалдығы алынады.

1.

	x^8	$+x^4$	$+x^3$	$+x$	$+1$	x^5	$+x^3$	$+x^2$	$+1$
$\oplus$	x^8	$+x^6$	$+x^5$	$+x^3$		x^3	$+x$	$+1$	
	x^6	$+x^5$	$+x^4$	$+x$	$+1$				
$\oplus$	x^6	$+x^4$	$+x^3$	$+x$					
	x^5	$+x^3$	$+1$						
$\oplus$	x^5	$+x^3$	$+x^2$	$+1$					
				x^2					

Жүргізілген бөлудің нәтижесін келесі түрде көрсетуге болады:

$$p(x) = x^8 + x^4 + x^3 + x + 1 = (x^5 + x^3 + x^2 + 1) \times$$
$$\times (x^3 + x + 1) + x^2 = a(x) \cdot (x^3 + x + 1) + x^2. \qquad (8.3)$$

Бөлуден қалған қалдықтың максималды дәрежесі (x^2) бөлгіштің максималды дәрежесінен (x^5) кем болуына байланысты бөлу процесі аяқталады. Бөлудің қалдығы x^2 1-ден үлкен болғандықтан, бөлу жалғасады, бөлінгіш ретінде ($x^5 + x^3 + x^2 + 1$) бөлгіші алынады, ал бөлгіш ретінде алдыңғы бөлуден қалған (x^2) қалдық алынады.

2.

$$
\begin{array}{lll}
 & x^5 + x^3 + x^2 + 1 & \mid\, x^2 \\
\oplus & x^5 & \hline \\
 & x^3 + x^2 + 1 & \mid\, x^3 + x + 1 \\
\oplus & x^3 & \\
 & x^2 + 1 & \\
\oplus & x^2 & \\
 & 1 &
\end{array}
$$

Осы бөлінді нәтижесінде 1 қалдық қалды, сондықтан бөлу процессі тоқтатылады.

Жүргізілген бөлу нәтижесін келесі түрде көрсетуге болады:

$$a(x) = x^5 + x^3 + x^2 + 1 = (x^3 + x + 1)\cdot x^2 + 1. \qquad (8.4)$$

(8.4) өрнегін келесі түрде көрсетеміз

$$a(x) + (x^3 + x + 1)\cdot x^2 = 1, \qquad (8.5)$$

ал (8.3) өрнегі

$$p(x) + a(x)\cdot(x^3 + x + 1) = x^2. \qquad (8.6)$$

(8.5)-ті (8.6)-ға қойып және түрлендіру жүргізіп келесіні аламыз

$$\begin{aligned} & a(x) + (x^3 + x + 1)\cdot(p(x) + a(x)\cdot(x^3 + x + 1)) = \\ & = a(x) + (x^3 + x + 1)\cdot p(x) + s(x)\cdot(x^3 + x + 1)^2 = \\ & = a(x)\cdot(1 + (x^3 + x + 1)^2) + (x^3 + x + 1)\cdot p(x) = 1. \end{aligned} \qquad (8.7)$$

$p(x) = x^8 + x^4 + x^3 + x + 1$ келтірілмеген полиномды $GF(2^8)$-те (8.7)-ның сол және оң жағын модульді мәнін есептеп келесіні аламыз:

$$(a(x)\cdot(1 + (x^3 + x + 1)^2) + (x^3 + x + 1)\cdot p(x)) \bmod p(x) = \qquad (8.8)$$

$$= a(x)\cdot(1 + (x^3 + x + 1)^2) \bmod p(x) +$$

$$+ ((x^3 + x + 1)\cdot p(x)) \bmod p(x) =$$

$$= a(x)\cdot(1 + (x^3 + x + 1)^2 \bmod p(x) = 1,$$

$((x^3 + x + 1)\cdot p(x)) \bmod p(x) = 0$ болғандықтан.

(8.8) өрнегінен келесі шығады:

$$a(x)\cdot(1 + (x^3 + x + 1)^2) \bmod p(x) = 1. \quad (8.9)$$

(8.9) бен (8.1) салыстыра отырып, төмендегіні аламыз:

$$a^{-1}(x) = (1 + (x^3 + x + 1)^2). \quad (8.10)$$

(8.10)-ды түрлендірулерді өткізіп, соңында аламыз:

$$a^{-1}(x) = x^6 + x^2. \quad (8.11)$$

Он алтылық санақ жүйесінде $a^{-1}(x)$ байт мәні *{ 44 }*–ке тең болады.

$a(x) = x^5 + x^3 + x^2 + 1$ элементіне $p(x) = x^8 + x^4 + x^3 + x + 1$ келтірілмеген көпмүшесі бар $GF(2^8)$ өрісінде $a^{-1}(x)$ мультипликативті кері элементті табудың дұрыстығын тексереміз. Ол үшін (8.1) өрнегіне $a(x)$ және $a^{-1}(x)$ мәнін қоя отырып, төмендегіні аламыз:

$$a(x)\cdot a^{-1}(x) \bmod p(x) = (x^5 + x^3 + x^2 + 1)\cdot(x^6 + x^2) =$$
$$= (x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^2) \bmod p(x) = 1. \quad (8.12)$$

(8.12) шығатындай $a(x)$ және $a^{-1}(x)$ элементтері $p(x)$ келтірілмеген полиномды $GF(2^8)$ өрісіндегі мультипликативті кері болып табылады.

8.3. AES ДЕРЕКТЕР ФОРМАТЫ

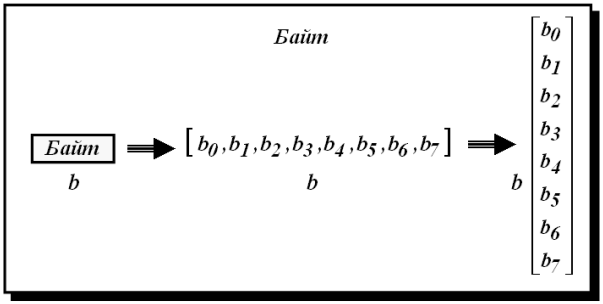
Rijndael – бұл "*Төртбұрыш*" сәулеті бар итерациялық блоктық шифр. Шифрда блоктардың айнымалы ұзындығы мен кілттердің әртүрлі ұзындықтары бар. Кілттің ұзындығы мен блоктардың ұзындығы бір-бірінен тәуелсіз *128*, *192* немесе *256* биттерге тең бо-

лу мүмкін. *AES* стандартында деректер блогының ұзындығы анықталған, ол *128* битке тең.

AES деректерді көрсету үшін бес бірлікті қолданады: *биттер*, *байттар*, *сөздер*, *блоктар* және *күй массивтері*. *Бит* – кіші және бастауыш бірлік; басқа бірліктер кіші бірліктер терминдерінде өрнектелуі мүмкін.

AES-те бит – *0* немесе *1* мәндері бар екілік сан. Биттерді белгілеу үшін жолдық әріптерді қолданамыз.

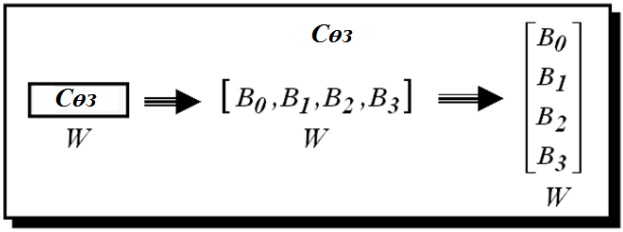
8.2 сурет деректердің бірлігін (пішім) – байтты көрсетеді.



8.2 сурет - Деректер бірлігі (пішім) – байт

Байт – бірегей объект ретінде өңделуі мүмкін сегіз биттен тұратын топ: сегіз биттің бір қатарынан (*1* × *8*) тұратын матрица немесе сегіз биттен тұратын матрицаның бағаны (*8* × *1*). Байттар ақпараты жолдың матрицасы ретінде өңделсе, биттер матрицаға солдан оңға қарай қойылады. Байттар бағандар матрицасы ретінде өңделсе, биттер матрицаға жоғарыдан төмен қарай қойылады. Байттарды белгілеу үшін жолдық әрпін (*b*) қолданамыз.

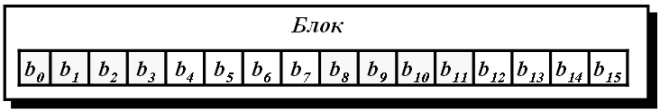
8.3 сурет деректер бірлігін (пішім) – сөзді көрсетеді.



8.3 сурет - Деректер бірлігі (пішім) – сөз

Сөз – бірегей объект ретінде өңделуі мүмкін 32 биттен тұратын топ. Бұл төрт байтты жолдан тұратын матрица немесе төрт байттан тұратын матрица бағаны. Сөз матрица-жол ретінде өңделсе, байттар солдан оңға қарай қойылады. Сөз матрица-баған ретінде берілсе, байттар жоғарыдан төмен қарай қойылады. Сөзді белгілеу үшін *W* үлкен әрпін қолданамыз.

8.4 сурет деректер бірлігін (пішім) – блокты көрсетеді.

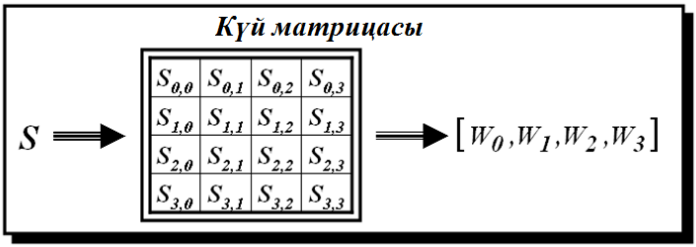


8.4 сурет - Деректер бірлігі (пішім) – блок

AES деректер блоктарын шифрлайды және керішифрлайды. *AES блок – 128* биттерден тұратын топ. Дегенмен блок 16 байттардан тұратын матрица-жол ретінде көрсетілу мүмкін.

AES бірнеше раундтардан тұрады, әр раунд бірнеше каскадтардан тұрады. Деректер блогы бір каскадтан басқасына түрленеді. *AES* шифрының басы мен аяғында *деректер блогы* термині қолданылады; әр каскадқа дейін және кейін деректер блогы *күй матрицасы* деп аталады. Бұл матрицаны белгілеу үшін *S* үлкен әрпін қолданамыз. Әртүрлі каскадтарда күй матрицасы *S*-пен белгіленсе де, кейде күйдің уақытша матрицасын белгілеу үшін *T* үлкен әрпін қолданамыз.

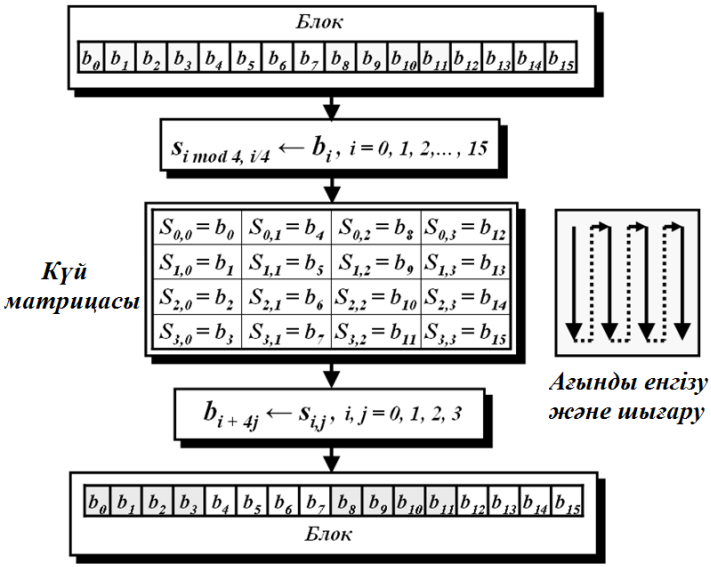
8.5 сурет деректер бірлігін (пішім) – күй матрицасын көрсетеді.



8.5 сурет - Деректер бірлігі (пішім) – күй матрицасы

Күй матрицасы блоктарға ұқсас, *16* байттан тұрады, бірақ әдетте 4×4 байттар матрицасы ретінде өңделеді. Бұл жағдайда күй матрицасының әр элементі $S_{r,c}$ деп белгіленеді, бұл жерде *r* (*0*-ден *3*-ке дейін) жолды анықтайды және *c* (*0*-ден *3*-ке дейін) бағанды

анықтайды. Кей кезде күй матрицасы сөздердің матрица-жолы (1×4) ретінде өңделеді. Бұл егер де сөзді матрица-баған ретінде көрсеткен кезде мағыналы болады. Шифр басында байттар деректер блогында бағаннан бағанға күй матрицасына қойылады, әр бағанда – жоғарыдан төмен қарай. Шифр соңында байттар күй матрицасында 8.6 суретте көрсетілгендей шығарылады:



8.6 сурет - Блоктың күй матрицасына және күй матрицасының блокқа түрленуі

8.2 мысал. 16 символы бар блокты 4×4 матрицасы түрінде қалай сипаттауға болатынын қарастырайық. Мәтіндік блок – "*AES uses a matrix*" болсын дейік. Соңына екі жалған символды қосамыз да, "*AESUSESAMATRIXZZ*"-ны аламыз. Енді әр символды *00* және *25* арасындағы бүтін санмен алмастырамыз. (2.1 суретті қараңыз). Әр байтты екі он алтылық сан бар бүтін сан ретінде көрсетейік. Мысалы, *S* символын алдымен *18*-ге, ал содан соң он алтылық бейнеде *12* ретінде жазамыз. Онда күй матрицасы 8.7 суретінде көрсетілгендей бағаннан бағанға толтырылады.

Мәтін	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Ондық мән	00	04	18	20	18	04	18	00	12	00	19	17	08	23	25	25
Он алтылық мән	00	04	12	14	12	04	12	00	0c	00	13	11	08	17	19	19

Күй матрицасы

00	12	0c	08
04	04	00	23
12	12	13	19
14	00	11	19

8.7 сурет - Алғашқы мәтіннің күй матрицасына өтуі

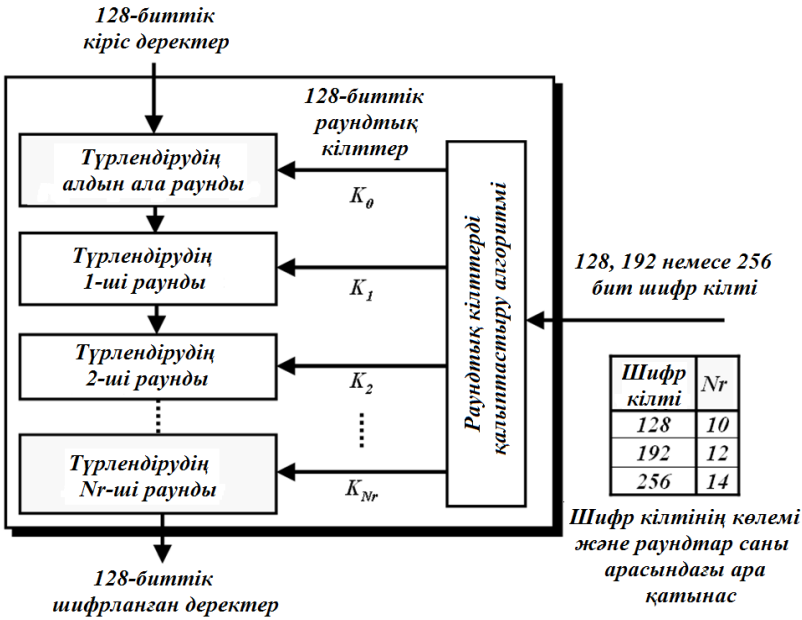
8.4. AES АЛГОРИТМІНІҢ ЖӘНЕ РАУНДТАРДЫҢ ҚҰРЫЛЫМЫ

8.4.1 Алгоритмнің құрылымы

AES – 10, 12 немесе *14* раундтарды қолдана отырып, 128 биттер деректер блогын шифрлайтын және керішифрлайтын *Фейстель-емес* шифры. Кілт өлшемі *128, 192* немесе *256* биттер болу мүмкін және раунд нөміріне тәуелді. 8.8 сурет жалпы сұлбаны көрсетеді: шифрлау алгоритмі (шифр деп аталады); сол кілттер бірақ кері тәртіппен қолданылатын керішифрлау алгоритмі (кері шифр деп аталады).

8.8 суретінде *Nr* раундтар нөмірін анықтайды. Сурет сонымен бірге раундтар саны мен кілттің өлшемі арасындағы қатынасты көрсетеді. Бұл *AES*-тің үш әртүрлі нұсқасы бар екендігін білдіреді; олар *AES-128, AES-192* және *AES-256* ретінде белгіленеді. Дегенмен кілттерді кеңейту алгоритмімен құрылған раунд кілттері әрқашан да 128 бит, олардың өлшемдері алғашқы немесе шифрланған деректер блоктарының өлшемімен бірдей болады.

Кілттерді кеңейту алгоритмімен генерацияланған раунд кілттерінің саны әркезде раундтар санына қарағанда біреуге артық. Басқаша айтқанда, $Nr + 1$ раундтық кілттер саны бар. Раундтық кілттерді келесідей белгілейміз: $K_0, K_1, K_2, ..., K_{Nr}$.



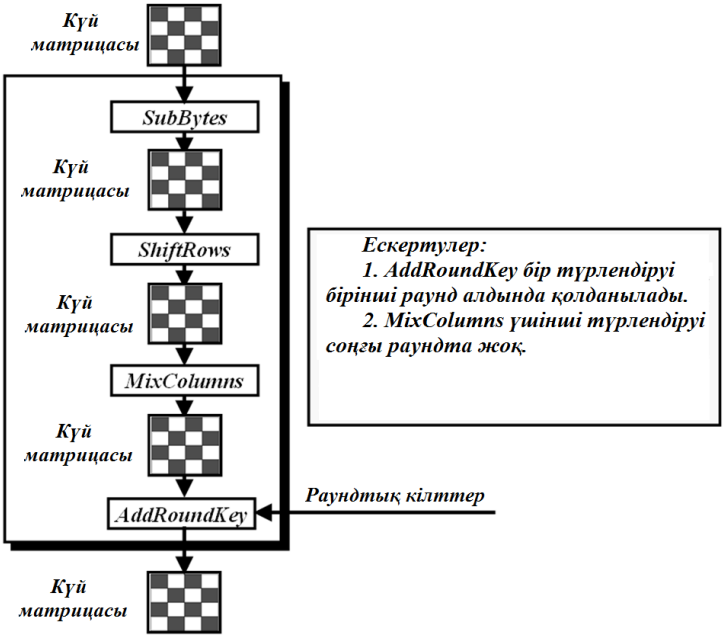
8.8-сурет. AES стандартының шифрлау сұлбасының жалпы құрылымы

8.4.2 Алгоритмнің раундтар құрылымы

8.9 сурет шифрлау жағында әр раундының құрылымын көрсетеді.

Соңғысынан басқа, әр раунд кері болатын төрт түрленуді қолданады. Соңғы раундта тек үш түрлендіру болады.

8.9 сурет көрсеткендей, әр түрлену күй матрицасын қабылдайды және келесі түрлендіру немесе келесі раунд үшін қолданылатын басқа күй матрицасын құрады. Раундтарды ескертіп отыратын секция тек бір түрлендіруді қолданады (*AddRoundKey*); соңғы раунд тек үш түрлендіруді қолданады (*MixColumns* – түрлендіруі жоқ).



8.9 сурет - Шифрлау жағында *AES* стандартының әр раундының құрылымы

8.4.3. Алгоритмнің раундтар саны

Блокта түрлендіруді орындау үшін *K* құпия кілтінен алынған *W* раундтық кілт қолданылады. *AES* стандартында раундтық кілт блоктардан тұрады (*128* бит бойынша). Сонымен қатар *AES* стандартында (*Rijndael*) 8.1 және 8.2 кестелерінде көрсетілгендей, кілттің өлшемі, деректер блогының өлшемі және шифрлау раундтарының саны арасындағы сәйкестік анықталған.

8.1 кесте

Nr шифрлау раундтарының саны *Nk* кілттер өлшемі мен *Nb* деректер блогы өлшемінің функциясы ретінде

Nr	*Nb = 4*	*Nb = 6*	*Nb = 8*
Nk = 4	*10*	*12*	*14*
Nk = 6	*12*	*12*	*14*
Nk = 8	*14*	*14*	*14*

8.2 кесте

AES стандартында *Nk* кілттің өлшемі, *Nb* деректер блогының өлшемі және *Nr* шифрлау раундтарының саны арасындағы сәйкестік

Стандарт	*Nk*	*Nb*	*Nr*
AES-128	*4*	*4*	*10*
AES-192	*6*	*4*	*12*
AES-256	*8*	*4*	*14*

Шифрдың авторлары барлық кілттік кеңістік бойынша толық іріктеуден тиімдірек (қысқартылған шабуыл деп аталатын) шабуыл жасауға мүмкіндігі бар раундтардың ең үлкен санын ескере отырып, раундтар санын анықтады және қосымша раунд түрінде қор қосты.

Деректердің және кілттердің *128*-биттік блоктары бар 6 раундтан тұратын шифрдың оңайлатылған нұсқасы үшін бір де бір оңайлатылған тиімді шабуыл табылмаған. Келесі түсініктерді есептей отырып тағы да 4 раундты қосу беріктіліктің жеткіліктіден көп қоры болады: шифрдың екі раунды толық шашыратуды қамтамасыз етеді, келесі мағынада, *State* блогының әр биты екі раунд бұрынғы сол блоктың барлық биттерінен тәуелді болады, немесе, басқаша айтқанда, *State* блогының бір битын өзгерту екі раундтан кейін үлкен ықтималдықпен бұл блоктың жартысына әсерін көрсетеді. Осылайша қосымша 4 раунд шифрлау процедурасының басы мен аяғында толығымен шашырату қадамын қосу ретінде қарастырылады.

Шифрдың шашырауының жоғары дәрежесі кіріс деректердің барлық биттерін түрлендіретін алгоритмнің тұтас сипатымен шартталған. *Фейстель* желісі үшін (*DES* және *ГОСТ 28147-89* алгоритмдері) раундтық түрлендіру бір қадамда кіріс деректер биттерінің тек жартысын ғана өзгертеді, және тәжірибеде толық шашырау *4* және одан да көп раундтар ішінде қамтылады.

Сызықтық және дифференциалдық криптографиялық талдауда қысқартылған дифференциал әдісімен шабуыл жасау кезінде $(n+1)$-ші немесе $(n+2)$-ші раундты шабуылдау үшін n раундтар барысында бақылауға болатын тәуелділіктерді қолданады. Бұл алтыншы раундқа шабуыл жасау үшін *4* раундтан өтетін тәуелділіктерді қолданатын *Square*-шабуыл үшін де дұрыс. Соңғылар үшін қосымша *4* раунд тәуелділікті қадағалау үшін раундтар санын қосарлатуды білдіреді.

Ұзындау кілті бар шифр нұсқалары үшін шифрлау кілтінің әр қосымша 32 разряды үшін раундтар саны бірге көбееді, келесі түсініктерге қарап: басты мақсаттарының бірі – қысқартылған шабуылдардың мүмкін еместігі, кілт ұзындығының өсуімен қарапайым іріктеу еңбек сыйымдылығы өсетіндіктен, қысқартылған шабуыл көп еңбекті бола алады.

Белгілі кілт (немесе жартылай белгілі) негізінде шабуылдар, "*эквивалентті кілттерді*" қолданумен шабуылдар шифрлаудың алғашқы кілттің биттері туралы ақпараттың және шифрлаудың бірнеше әртүрлі кілттерін қолдану мүмкіндігінде сәйкесінше негізделген. Егер кілт ұзарса, криптографиялық талдаушының мүмкіндіктері де көбееді.

Алты раундты шифр үшін белгілі кілт бойынша немесе "*эквивалентті кілттерді*" қолданумен ешқандай тиімді шабуылдар табылмағандықтан, қалған раундтар беріктіліктің қосымша қоры ретінде қарастырылады.

Кіріс деректер блоктарының өлшемі *128* разрядтан көп *Rijndael* шифрының басқа нұсқалары (стандартқа кірмеген) үшін раундтар саны әр қосымша *32*-разрядты сөзбен бірге көбееді, келесі түсініктерге негізделіп: *128* разрядтан үлкен өлшемді блоктар үшін түрлендірудің үшінші раундын жүргізгеннен кейін толық шашырау болады, яғни раундтың жанама шашырауы кіріс деректер блогының өлшемінің өсуімен азаяды.

Кіріс деректер блогының өлшемінің өсуімен бірге кіріс/шығыс деректер тәуелділігінің мүмкін комбинациялар саны көбееді. Бұл, өз кезегінде, кейде тағы бір раундқа шабуылдың тиімділігін созуға мүмкіндік береді.

Шифрдың авторлары тағы бір раундқа шабуыл тиімділігінің таралу қаупі тіпті 256-разрядты блоктар үшін азықтималды екенін айтады. Сондықтан нөмірі алтыдан көп барлық раундтар шифрдың беріктігінің қосымша қоры ретінде қарастыруға болады.

Rijndael құрастыруының негізіне үш критерий қойылды:

- барлық белгілі шабуылдарға беріктілік;
- кодтың жылдамдығы мен ықшамдылығы;
- дизайнының қарапайымдылығы.

Алдында қарастырылған симметриялық блокты шифрлармен салыстырғанда, *Rijndael Фейстель* құрылымының қандай да бір аналогын қолданбайды. Әр раунд *қабат* деп аталатын үш әртүрлі кері түрлендіруден тұрады:

- сызықты араластырғыш қабат статистикалық байланыстарды қалқалау үшін блок символдарының өзара сіңісуінің жоғары дәрежесін кепілдендіреді;

- сызықтық емес қабат оңтайлы сызықтық емес *S*-блоктар көмегімен жүзеге асырылған және криптографиялық талдаудың дифференциалды, сызықтық және басқа қазіргі заманғы әдістерді қолдану мүмкіндігін болдырмайды;

- кілтпен қосу қабаты тікелей шифрлауды орындайды.

Шифр кілтті қосумен басталады және аяқталады. Бұл белгілі мәтін бойынша шабуыл кезінде бірінші раунд кірісін жабуға және соңғы раунд нәтижесін криптографиялық маңызды етуге мүмкіндік береді.

8.5. AES АЛГОРИТМНІҢ РАУНДТЫҚ ТҮРЛЕНДІРУЛЕРІ

AES (*Rijndael*) алгоритмі деректерді шифрлау үшін төрт әртүрлі байтқа-бағытталған (раундтық) түрлендірулерден тұратын *айналма функциясы* деп аталатын функцияны қолдалады:

- ауыстыру кестесін – *SubBytes(State)* қолданумен күй массив байттарын ауыстыру, яғни 8×256 өлшемі бар тіркелген ауыстыру кестемен *S*-блоктарда байт бойынша ауыстыру;

- позицияның әртүрлі санына күй массивінде жолдарды жылжыту – *ShiftRows(State)*;

- күй матрицасының бағандарын араластыру – *MixCo-lumns(State)* (*GF*(2^4)-да көпмүшелер ретінде қарастыратын күй матрицасы бағандарын $x^4 + 1$ модулі бойынша $g(x)$ үш дәрежелі көпмүшеге көбейту);

- күй матрицасына раундтық кілтті қосу – *Add-RoundKey(State, RoundKey)*, яғни жайылған кілттің ағындағы фрагментімен күй матрицасына разряд бойынша *xor* орындау.

AES (*Rijndael*) алгоритмі деректерді керішифрлау үшін де төрт әртүрлі байтқа-бағытталған (раундтық) түрлендіруден тұратын *айналма функциясын* қолданады, олар шифрлау кезінде қолданған түрлендірулерге сәйкесінше қайтымды болатын *InvSubBytes*(*State*), *InvShiftRows*(*State*) және *InvMixCo-lumns*(*State*) функциялары. *InvAddRoundKey*(*State, RoundKey*) инверсті түрлендіруі *AddRound Key*(*State, RoundKey*) түрлендіруіне ұқсас, өйткені разряд бойынша *xor* операциясын қолданады.

8.5.1. *SubBytes(State)* және *InvSubBytes(State)* күй матрицаның байттарын ауыстыру

Күй матрицаның байттарын ауыстыру – SubBytes()

SubBytes() (байттарды ауыстыру) процедурасы шифрлау жағында қолданылады, сызықтық емес түрлендіру қабатын жүзеге асырады және күйдің әр байтына тәуелсіз орындалатын байттардың сызықтық емес ауыстыруы болады. *S*-блок ауыстыру кестелері кері және кіріс байтының келесі екі түрлендіруінің композициясынан құрылған:

- $GF(2^8)$ өрісінде көбейтуге қатысты мультипликативті кері элементті алу, яғни теңдеуді $s^{-1}(x)$ элементіне қатысты шешу:

$$s(x)\cdot s^{-1}(x) \bmod p(x) = 1 \qquad (8.13)$$

(нөлдік элемент *{ 00 }* өз өзіне айналады);

- төмендегідей түрде анықталған [13, 29, 38] $GF(2^8)$-те аффиндік түрлендіруді қолдану:

$$d(x)\cdot = (a(x)\cdot s^{-1}(x) + b(x)) \bmod (x^8 + 1), \qquad (8.14)$$

бұл жерде $a(x)$ және $b(x)$ – *AES* (*Rijndael*) алгоритмі үшін тіркелген мәні бар полиномдар және олардың мәндері келесіге тең:

$$a(x) = x^7 + x^6 + x^5 + x^4 + 1; \quad b(x) = x^6 + x^5 + x + 1.$$

Rijndael шифрында ауыстыру блоктары маңызды рөл атқарады. *К.Шеннон* қалыптастырған негізгі принциптеріне сәйкес шифрда қолданылатын деректерді түрлендірулер соңғыға екі негізгі қасиеттерді беру қажет – шашырату және араластыру. *Шашырату* ашық деректердің әр битінің, сонымен бірге кілттің әр битінің шифрланған деректердің биттердің үлкен санына ықпалын таратуды болжайды. *Араластыру* шифрлау процесінде ашық деректердің биттері арасындағы әртүрлі тәуелділікті жоғалтуға әкеледі. Яғни шығыстағы деректер егер олар шифрлау алгоритмінің түрлендіру функциясының мәні сияқты емес кездейсоқ алынанғандай болады. Дәл осы екі қасиет екі мүмкін қауіптен қорғауды қамтамасыз етеді: *жалған хабарды жасау* және *оны ашу*.

Осы екі қасиеті қамтамасыз ету үшін *MixColumns()* және *SubBytes()* функциялары жауап береді. *Rijndael* шифрында

шашырату негізінен *MixColumns()* функциясымен қамтамасыз етілетін, орын ауыстыру – негізінен *SubBytes()* функциясымен. Осылайша, *S*-блоктарын ауыстыру кестелерін таңдау мен құру критерийлері бір жағынан, дифференциалды және сызықты криптографиялық талдау мүмкіндіктерін есепке алумен, басқа жағынан, мүмкін алгебралық манипуляцияларын есепке алумен шартталған, мысалы, интерполяция әдісімен шабуыл (төменде қарастырылады). Осылайша, *S*-блоктар ауыстыру кестелерін таңдаудың негізгі критерийлері төмендегілер болды:

- қайтымдылық;

- кіріс және шығыс деректер биттерінің сызықты комбинациясы арасындағы ең ұзын мүмкін тривиалды емес корреляцияны минимизациялау; басқаша айтқанда, кіріс және шығыс деректер арасында тәуелдікті байқауға болатын раундтар саны барлық кездейсоқ таңдалған кіріс деректер үшін минималды болады; дәл осы тәуелдіктерді байқау сызықты криптографиялық талдау негізі болып табылады;

- *xor* кестесінің ең үлкен тривиалды емес мәнінің минимизациясы; дифференциалды криптографиялық талдауға беріктілікті сипаттайды;

- $GF(2^8)$ өрісінде алгебралық көрсетудің қиындығы; алгебралық шабуылға берік болу үшін маңызды;

- сипаттауының қарапайымдылығы.

[7, 29, 38]-де бастапқы үш критерийді қамтамасыз ететін *S*-блоктар кестесін құрастырудың бірнеше әдістері келтірілген. Ауыстырудың кері байттық кестесі үшін кіріс/шығыс деректерінің максималды корреляциясы 2^{-3}-не дейін минимизациялану мүмкін, ал *xor* кестесінің максималды шамасы – *4*-ке дейін минимизациялану мүмкін (2^{-6} дифференциалды ену коэффициентіне сәйкес).

Ауыстыру кестесін қалыптастыру үшін $GF(2^8)$ өрісінде $s \rightarrow s^{-1}$ (мультипликативті инверсия) бейнелеуі таңдалды. Дегенмен таңдалған бейнелеуде алгебралық көрінісі өте қарапайым екені айқын. Бұл шифрға шабуылдарда алгебралық манипуляцияларды қолдануға мүмкінкіндік береді, мысалы, *интерполяция әдісімен шабуыл* [38]. Сондықтан түрлендіру нәтижесіне қосымша (әбден кері) өзгерістер жасалады.

Бұл өзгеріс бастапқы үш критерийге қатысты *S*-блок қасиеттерін азайтпайды, бірақ ауыстыру кестесін төртінші таңдау

критерийіне сәйкес болу талабын қанағаттандыруға мүмкіндік береді. Өзі бөлек сипаттауда қарапайым, бірақ $GF(2^8)$ өрісінде мультипликативті кері элементті табумен бірге күрделі алгебралық көрінісіне ие болатын өзгеріс таңдалды. Ол кейін қосуға болатын $x^8 + 1$ модулі бойынша көпмүшелерді көбейту ретінде көрсетілу мүмкін:

$$\begin{aligned} s(x) &= (a(x)\cdot d(x) + b(x)) \bmod (x^8 + 1) = \\ &= ((x^7 + x^6 + x^5 + x^4 + 1)\cdot d(x)\cdot + \\ &+ (x^6 + x^5 + x + 1)) \bmod (x^8 + 1). \end{aligned} \tag{8.15}$$

Бұл жерде $d(x)$ – көпмүше түрінде түрлендірілетін байт, $s(x)$ – нәтижелік байт. $x^8 + 1$ модулі мүмкіндердің ішінен ең қарапайымы ретінде таңдалды. $x^7 + x^6 + x^5 + x^4 + 1$ көбейткіші ((8.15) өрнегіндегі $a(x)$) қарапайым көрініске ие, $x^8 + 1$ модулімен өзара қарапайым көпмүшелер жиынынан таңдалған. Ал $x^6 + x^5 + x + 1$ көпмүшесі ((8.15) өрнегіндегі $b(x)$) нәтижесінде алынған ауыстыру кестісінде симметрия нүктесі ($S(d(x)) = d(x)$) және кері симметрия нүктесі ($S(d(x)) = d^{-1}(x)$ жоқ болатындай таңдалған.

Басқаша айтқанда (8.15) түрлендіру мағынасы теңдеулермен сипатталуы мүмкін:

$$\begin{aligned} s_i = d_i \oplus d_{(i+7) \bmod 8} \oplus d_{(i+6) \bmod 8} \oplus d_{(i+5) \bmod 8} \oplus \\ \oplus d_{(i+4) \bmod 8} \oplus b_i, \end{aligned} \tag{8.16}$$

бұл жерде d_i мен s_i - d_i мен s_i – сәйкес i-ші биттің алғашқы және түрлендірілген мәні, $i = 0,1,\ldots,7$; $b_0 = b_1 = b_5 = b_6 = 1$, $b_2 = b_3 = b_4 = b_7 = 0$.

(8.16) түрлендіруді қолдануды матрицалық түрде төмендегідей сипаттауға болады:

$$S(x) = A(x)\cdot D(x) \oplus B(x), \tag{8.17}$$

бұл жерде $S(x)$, $D(x)$ және $B(x)$ – 8×1 өлшемі бар вектор-бағандар; $A(x)$ – 8×8 өлшемі бар матрица.

(8.17) өрнегін жайылған түрде көрсетуге болады:

$$s(x) = \begin{vmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{vmatrix} \oplus \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{vmatrix}. \qquad (8.18)$$

(8.18) өрнегінен көрінетіндей *A(x)* матрицасы тең:

$$A(x) = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}. \qquad (8.19)$$

A(x) матрицасына ұқсас матрицалар *Тёплица матрицасы* (теплицалық матрицалар) немесе диагоналды-тұрақты матрицалар деп аталады. Сызықты алгебрада неміс математигі *Отто Тёплица* атымен аталған *Тёплица* матрицасы – бұл барлық бастысына қатарлас диагоналдарда (бастысын қоса алғанда) тең элементтер болатын матрица.

Ескерту. Жоғарыдағы критерийлерге қанғаттандыратын басқа да *S*-блоктар бар. Берілген кестеде «*қара жол»* болуы мүмкін жағдайда ол оңай басқасына ауыса алады. Сонымен бірге шифр құрылымы мен таңдалған раундтар саны *2* мен *3* критерийлерін қанағаттандырмайтын ауыстыру кестесін қолдануға мүмкіндік береді. Бұл қатынасқа тіпті *«орташа»* ауыстыру кестесінің өзі дифференциалды және сызықты криптографиялық талдауға беріктілікті қамтамасыз етеді.

8.3 мысал. SubBytes() функциясының кірісінде кейбір күй байты *{2a}*-ға тең болсын дейік. Полиномиалды көрсетуде бұл *s(t)* = = x^5+

$x^3 + x$ (ω^{166}). Ол үшін $GF(2^8)$ өрісінде мультипликативті кері мәнді анықтау қажет.

Шешім. $GF(2^8)$ өрісінде мультипликативті кері элементті табу үшін примитивті элементтерге қатысты теңдік: $\omega^i \cdot \omega^j = 1$ ($i + j = 255$) орындалу қажет. $i = 166$ (ω^{166}) кезінде j 89 (ω^{89})-ке тең болады. Бұл жағдайда ω^{89} примитивті элементке [9] $s(t) = x^5 + x^3 + x$ көпмүшеге мультипликативті кері болатын $s^{-1}(t) = x^7 + x^4 + x^3$ көпмүше сәйкес келеді. Бұны тексеру үшін $GF(2^8)$ өрісінде $s(t)$ және $s^{-1}(t)$ көпмүшелерін $p(x) = x^8 + x^4 + + x^3 + x + 1$ келтірілмеген көпмүшемен көбейтеміз. Егер көбейтінді бірге тең болса, онда $s(t)$ және $s^{-1}(t)$ шынында мультипликативті кері болады.

$$s(x)\cdot s^{-1}(x) \bmod p(x) = (x^7 + x^4 + x^3)\cdot(x^5 + x^3 + x) = \\ = (x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5 + x^4) \bmod p(x) = 1. \qquad (8.20)$$

Сонымен, *SubBytes()* функциясының кірісінде *{ 2a }* күй байтының мультипликативті кері мәні *{ 98 }* байты болады ($s^{-1}(x) = x^7 + x^4 + x^3$ көпмүшенің эквиваленті).

8.4 мысал. Мультипликативті кері мәнді есептеу нәтижесінде алынған күйдің кейбір байты *{ 98 }-ге* тең болсын. Полиномиалды көріністе бұл $d(x) = s^{-1}(x) = x^7 + x^4 + + x^3$ (8.1 мысалын қараңыз). $GF(2^8)$ өрісінде ол үшін аффиндік түрлендіруді анықтау қажет.

Шешім. (8.16) немесе (8.18) өрнегіне $d(x)$ полиномының мәнін қойып, түрлендіруді жүргізіп және есептеп келесіні аламыз:

$$S(x) = \{ s_7, s_6, s_5, s_4, s_3, s_2, s_1, s_0 \} = \{ 1, 1, 1, 0, 0, 1, 0, 1 \}. \qquad (8.21)$$

(8.21) көрсетілімдерін орындап, $s(x)$ аламыз:

- екілік жүйеде – $s(x) =$*{ 11100101 }*;
- он алтылық жүйеде – $s(x) =$*{ e5 }*.

Егер *{00}*-дан *{ff}*-ге дейінгі барлық мүмкін күйлер (байттар) үшін мультипликативті инверсиялар мен аффиндік түрлендірулерді есептесек, онда 8.3-кестесіне жиналған деректер жиыны болады.

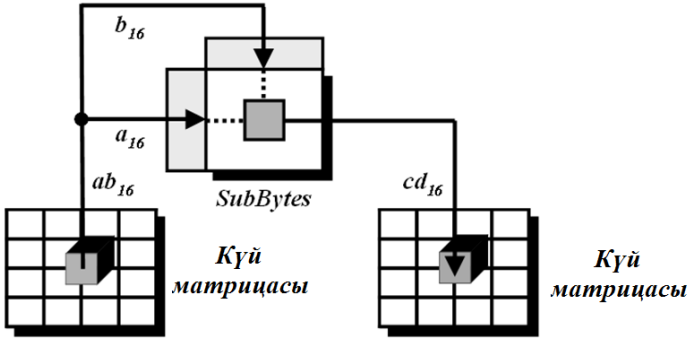
SubBytes(state) түрлендіруі *state s* әр байты үшін $s \leftarrow S(s)$ операциясын орындауға келтіріледі. Байтты ауыстыруын қолдану үшін байтты екі он алтылық сан ретінде түсіну қажет. Сол жақтағы сан 8.3-кестесінің жолын, ал оң жақтағы сан бағанын анықтайды. Осы он алтылық сандармен белгіленген жолдар мен бағандар

қиылысында жаңа байт орналасқан. 8.10 суреті осы идеяны суреттейді.

8.3-кесте

SubBytes() түрлендіру кестесі

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	*63*	*7c*	*77*	*7b*	*f2*	*6b*	*6f*	*c5*	*30*	*01*	*67*	*2b*	*fe*	*d7*	*ab*	*76*
1	*ca*	*82*	*c9*	*7d*	*fa*	*59*	*47*	*fo*	*ad*	*d4*	*a2*	*af*	*9c*	*a4*	*72*	*c0*
2	*b7*	*fd*	*93*	*26*	*36*	*3f*	*f7*	*cc*	*34*	*a5*	*e5*	*f1*	*71*	*d8*	*31*	*15*
3	*04*	*c7*	*23*	*c3*	*18*	*96*	*05*	*9a*	*07*	*12*	*80*	*e2*	*eb*	*27*	*b2*	*75*
4	*09*	*83*	*2c*	*1a*	*1b*	*6e*	*5a*	*ao*	*52*	*3b*	*d6*	*b3*	*29*	*e3*	*2f*	*84*
5	*53*	*d1*	*00*	*ed*	*20*	*fc*	*b1*	*5b*	*6a*	*cb*	*be*	*39*	*4a*	*4c*	*58*	*cf*
6	*do*	*ef*	*aa*	*fb*	*43*	*4d*	*33*	*85*	*45*	*f9*	*02*	*f7*	*50*	*3c*	*9f*	*a8*
7	*51*	*a3*	*40*	*8f*	*92*	*9d*	*38*	*f5*	*bc*	*b6*	*da*	*21*	*10*	*ff*	*f3*	*d2*
8	*cd*	*0c*	*13*	*ec*	*5f*	*97*	*44*	*17*	*c4*	*a7*	*7e*	*3d*	*64*	*5d*	*19*	*73*
9	*60*	*81*	*4f*	*dc*	*22*	*2a*	*90*	*88*	*46*	*ee*	*b8*	*14*	*de*	*5e*	*0b*	*db*
a	*e0*	*32*	*3a*	*0a*	*49*	*06*	*24*	*5c*	*c2*	*d3*	*ac*	*62*	*91*	*95*	*e4*	*79*
b	*e7*	*cb*	*37*	*6d*	*8d*	*d5*	*4e*	*a9*	*6c*	*56*	*f4*	*ea*	*65*	*7a*	*ae*	*08*
c	*ba*	*78*	*25*	*2e*	*1c*	*a6*	*b4*	*c6*	*e8*	*dd*	*74*	*if*	*4b*	*bd*	*8b*	*8a*
d	*70*	*3e*	*b5*	*66*	*48*	*03*	*f6*	*oe*	*61*	*35*	*57*	*b9*	*86*	*c1*	*1d*	*9e*
e	*e1*	*f8*	*98*	*11*	*69*	*d9*	*8e*	*94*	*9b*	*1e*	*87*	*e9*	*ce*	*55*	*28*	*df*
f	*8c*	*a1*	*89*	*0d*	*bf*	*e6*	*42*	*68*	*41*	*99*	*2d*	*0f*	*b0*	*54*	*bb*	*16*



8.10 сурет - *SubBytes()* түрлендірудегі кестелік ауыстыру идеясын түсіндіру

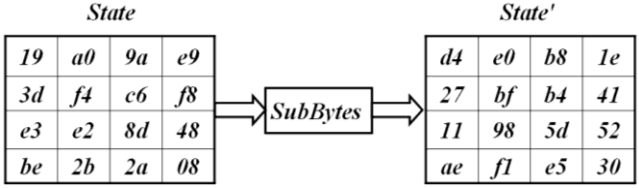
Мысалы, *{ 53 }* байтын түрлендіру нәтижесі *5*-ші жол мен 3-баған қиылысында орналасқан және *{ ed }*-ге тең.

SubByte() түрлендіруінде күй матрицасы *4x4* байттар матрицасы ретінде өңделеді. Бір сәтте бір байт түрлендіруі жүргізіледі. Әр байт мазмұны өзгереді, бірақ матрицада байттардың орналасуы сол қалыпта қалады. Түрлендіру процесінде әр байт басқаларынан тәуелсіз түрленеді – бұл байттаң байтқа он алты түрленуі.

8.5 мысал. AES алгоритмінің *SubBytes()* функциясының кірісіне (*Nb = 4*) келесіге күй матрицасы кірсін:

$$State = 193de3bea0f4e22b9ac68d2ae9f84808_{16}.$$

Егер әр байттың кестелік ауыстыруын жүргізсе (8.3-кестесін қараңыз), онда (8.11 сурет) аламыз:



8.11 сурет - Күй матрицасының әр байтына кестелік ауыстыруды орындау нәтижесі

SubBytes() түрлендіруі араластыруды қамтамасыз етеді. Мысалы, бір битте ғана айырмашылығы бар $5a_{16}$ мен $5b_{16}$ екі байт, сәйкесінше төрт битте айырмашылығы бар be_{16} мен 39_{16}-ға айналды.

InvSubBytes() – күй матрицасы байттарын ауыстыру

Алгоритмде қолданылған кері (инверсті) түрлендірулер табиғи түрде анықталады.

InvSubBytes(State) (байттарды инверсті ауыстыру) процедурасы сызықты емес түрлендірулер қабатын жүзеге асырады және күйдің әр байтымен тәуелсіз орындалатын байттардың мультипликативті инверсты сызықты емес ауыстыруы болып табылады. *InvSubBytes()* процедурасының *S*-блогының ауыстыру кестелері қайтымды болады және кіріс байтының келесі екі түрлендіруінің композициясынан құрастырылған:

1. Келесі түрде анықталған $GF(2^8)$ аффиндік түрлендіруге (8.14) инверсияны қолдану [7, 29, 38]:

$$d^{-1}(x)\cdot = (s(x) + b(x))\cdot a^{-1}(x) \bmod (x^8 + 1), \qquad (8.22)$$

бұл жерде $a^{-1}(x) = (x^7 + x^6 + x^5 + x^4 + 1)^{-1} \bmod (x^8 + 1) = x^7 + x^5 + x^2$.

Басқаша айтқанда (8.22) түрлендіру мағынасы теңдеулермен сипатталуы мүмкін:

$$d_i = c_{(i+7) \bmod 8} \oplus c_{(i+5) \bmod 8} \oplus c_{(i+2) \bmod 8}, \qquad (8.23)$$

бұл жерде $c_i = s_i \oplus b_i$; $b_0 = b_1 = b_5 = b_6 = 1$, $b_2 = b_3 = b_4 = b_7 = 0$, s_i және d_i – i-ші биттің алғашқы және түрлендірілген мәні, $i = 0,1,...,7$.

(8.22) түрлендіруді қолдануды матрицалық түрде төмендегідей сипаттауға болады:

$$d^{-1}(x) = \begin{vmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix} \cdot \left(\begin{vmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{vmatrix} \oplus \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{vmatrix} \right). \qquad (8.24)$$

(8.24) түрлендіруді қолдануды матрицалық түрде төмендегідей сипаттауға болады:

$$D^{-1}(x) = A^{-1}(x)\cdot(S(x) \oplus B(x)),$$

бұл жерде D^{-1}, $S(x)$ және $B(x)$ – 8×1 өлшемді вектор-бағандар; $A^{-1}(x)$ – 8×8 өлшемді матрица.

(8.24) өрнегінен көрінетіндей $A^{-1}(x)$ матрицасы тең:

$$A^{-1}(x) = \begin{vmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

2. $GF(2^8)$ өрісінде көбейтуге қатысты мультипликативті кері элементті алу, яғни теңдеуді шешу

$$d(x) \cdot d^{-1}(x) \bmod p(x) = 1 \qquad (8.25)$$

$d^{-1}(x)$ элементіне қатысты (нөлдік элемент *{ 00 }* өз өзіне ауысады).

8.6 мысал. InvSubBytes() функциясының кірісінде күйдің кейбір байты *{ e5 }*-ке тең болсын. Полиномиалды түрде бұл $s(t) = x^7 + x^6 + x^5 + x^2 + 1$. Ол үшін $GF(2^8)$ өрісінде инверсті аффиндік түрлендіруді анықтау қажет.

Шешім. s(x) көпмүшенің мәнін (8.23) немесе (8.24) өрнегіне қойып, түрлендіруді өткізіп және есептеп, келесіні аламыз:

$$d^{-1}(x) = \{ d_7, d_6, d_5, d_4, d_3, d_2, d_1, d_0 \} = \{ 1, 0, 0, 1, 1, 0, 0, 0 \}. \qquad (8.26)$$

(8.26) орындап, $d^{-1}(x)$ аламыз:

- екілік жүйеде – $d^{-1}(x) = \{ 10011000 \}$;
- он алтылық жүйеде – $d^{-1}(x) = \{ 98 \}$.

8.7 мысал. Инверсті аффиндік түрлендіру *InvSubBytes()* функциясын есептеу нәтижесінде алынған күйдің кейбір байт *{ 98 }* тең болсын. Полиномиалды түрде бұл $d^{-1}(x) = x^7 + x^4 + x^3$ (8.6 мысалын қараңыз). Ол үшін $GF(2^8)$ өрісінде мультипликативті кері мәнді анықтау қажет.

Шешім. $GF(2^8)$ өрісінде мультипликативті кері элементті табу үшін примитивті элементтерге қатысты теңдік орындалу қажет: $\omega^i \cdot \omega^j = 1$ $(i + j = 255)$. $i = 89$ (ω^{89}) болған кезде j 166 (ω^{166})-ға тең болады. Бұл жағдайда ω^{166} примитивті элементіне [9]

$s(x) = x^5 + x^3 + x$ көпмүшесі сәйкес келеді, ол $d^{-1}(x) = x^7 + x^4 + x^3$ көпмүшесіне мультипликативті кері болады.

Сонымен, күй байтының *{ 98 }* $GF(2^8)$ өрісінде *InvSubBytes()* функциясының кірісінде мультипликативті кері мәні *{ 2a }* байты болады ($s(x) = x^5 + x^3 + x$ көпмүше эквиваленті).

Егер *{ 00 }*-ден *{ ff }*-ке дейінгі барлық мүмкін күйлер (байттар) үшін аффиндік және мультипликативті кері түрлендірулердің инверсиясы есептелсе, онда 8.4 кестесіне жазылатын деректер жиыны алынады. Онда *InvSubBytes(state)* түрлендіруі *state* *s*-тің әр байты үшін $s \leftarrow S(s)$ операциясын орындауға әкеледі.

8.3 және 8.4 кестелері өзара қайтымды. Мысалы, егер *{ 2d }* байтына *SubBytes()* түрлендіруін орындаса (8.3 кестесін қараңыз), онда *{ d8 }* аламыз. Егер енді *{ d8 }* байтына *InvSubBytes()* түрлендіруін (8.4 кестесі) қолдансақ, онда *{ 2d }* аламыз.

8.4 кесте

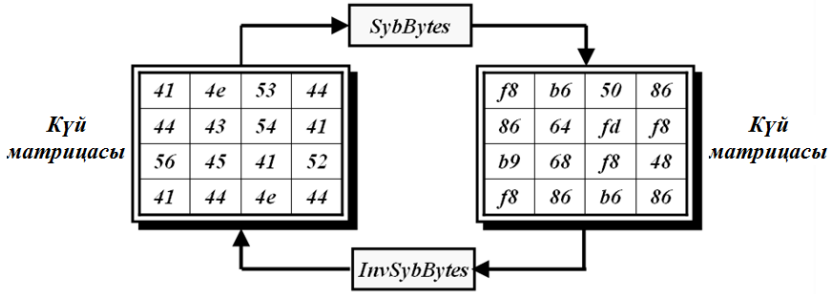
InvSubBytes() түрлендіру кестесі

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	*52*	*09*	*6a*	*d5*	*30*	*36*	*a5*	*38*	*bf*	*40*	*a3*	*9e*	*81*	*f3*	*d7*	*fb*
1	*7c*	*e3*	*39*	*82*	*9b*	*2f*	*ff*	*87*	*34*	*8e*	*43*	*44*	*c4*	*de*	*e9*	*cb*
2	*54*	*7b*	*94*	*32*	*a6*	*c2*	*23*	*3d*	*ee*	*4c*	*95*	*0b*	*42*	*fa*	*c3*	*4e*
3	*08*	*2e*	*a1*	*66*	*28*	*d9*	*24*	*b2*	*76*	*5b*	*a2*	*49*	*6d*	*8b*	*d1*	*25*
4	*72*	*f8*	*f6*	*64*	*86*	*68*	*98*	*16*	*d4*	*a4*	*5c*	*cc*	*5d*	*65*	*b6*	*92*
5	*6c*	*70*	*48*	*50*	*fd*	*ed*	*b9*	*da*	*5e*	*15*	*46*	*57*	*a7*	*8d*	*9d*	*84*
6	*90*	*d8*	*ab*	*00*	*8c*	*bc*	*d3*	*0a*	*f7*	*e4*	*58*	*05*	*b8*	*b3*	*45*	*06*
7	*d0*	*2c*	*1e*	*8f*	*ca*	*3f*	*0f*	*02*	*c1*	*af*	*bd*	*03*	*01*	*13*	*8a*	*6b*
8	*3a*	*91*	*11*	*41*	*4f*	*67*	*dc*	*ea*	*97*	*f2*	*cf*	*ce*	*f0*	*b4*	*e6*	*73*
9	*96*	*ac*	*74*	*22*	*e7*	*ad*	*35*	*85*	*e2*	*f9*	*37*	*e8*	*1c*	*75*	*de*	*6e*
a	*47*	*e1*	*1a*	*71*	*1d*	*29*	*c5*	*89*	*6f*	*b7*	*62*	*0e*	*aa*	*18*	*be*	*1b*
b	*fc*	*56*	*3e*	*4b*	*c6*	*d2*	*79*	*20*	*9a*	*db*	*c0*	*fe*	*78*	*cd*	*5a*	*f4*
c	*1f*	*d*	*a8*	*33*	*88*	*07*	*c7*	*31*	*b1*	*12*	*10*	*59*	*27*	*80*	*ec*	*5f*
d	*60*	*51*	*7e*	*a9*	*19*	*b5*	*4a*	*0d*	*2d*	*e5*	*7a*	*9f*	*93*	*c9*	*9c*	*ef*
e	*a0*	*e0*	*3b*	*4d*	*ae*	*2a*	*f5*	*b0*	*cb*	*eb*	*bb*	*3c*	*83*	*53*	*99*	*61*
f	*17*	*2b*	*04*	*7e*	*ba*	*77*	*d6*	*26*	*e1*	*69*	*14*	*63*	*55*	*21*	*0c*	*7d*

8.8 мысал. 8.12 сурет *SubBytes()* және *InvSubBytes()* қолданумен күй матрицасы қалай түрлендірілетінін көрсетеді.

Сурет сонымен бірге *InvSubBytes()* түпнұсқаны қалпына келтіретінін көрсетеді. Егер екі байттың бірдей мәні болса, онда

олар бірдей түрлендіріледі. Мысалы, күйдің сол матрицасындағы $4e_{16}$ және $4e_{16}$ екі байт күйдің оң матрицасында $b6_{16}$ және $b6_{16}$-ға айналады және керісінше. Себебі әр байт сол түрлендіру кестесін қолданады.



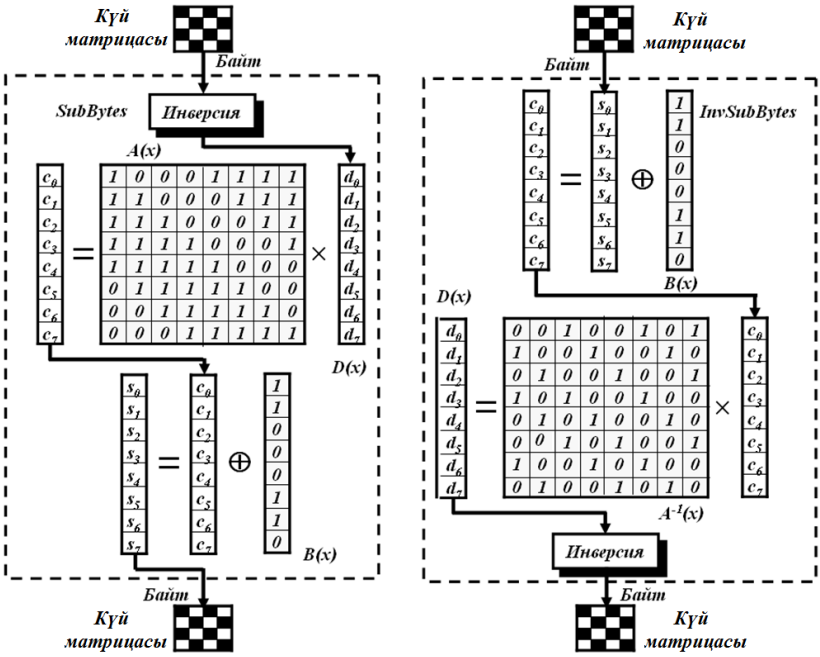
8.12 сурет - 8.8 мысалы үшін *SubBytes()* және *InvSubBytes()* түрлендірулері

Әр байттың орнын ауыстыру үшін 8.3 және 8.4 кестелерін қолдануға болатын болса да, *AES* алгебралық түрлендірулерге 8.13 суретте көрсетілгендей көпмүшелер көмегімен $GF(2^8)$ өрісі негізінде анықтама береді.

SubByte() процедурасында байт (екілік жолда *8* битке) $p(x) = x^8 + x^4 + + x^3 + x + 1$ келтірілмеген көпмүшемен $GF(2^8)$ өрісінде орналасады. 00_{16} байты өзі өзіндік айналымы болып табылады. Кері байт ең кіші биті жоғарыда және үлкен бит төменде орналасқан матрица-баған ретінде түсіндіріледі. Бұл матрица-баған тұрақты төртбұрыш матрицасына *A(x)* көбейтіледі және нәтижесі матрица-баған болады, ол жаңа байтты беретін тұрақты матрица-бағанмен *B(x)* қосылады. Көбейту және биттер сомасы $GF(2^8)$-те болады. *InvSubByte()* сол әрекеттерді істейді, бірақ кері тәртіпте.

Шифрлау процесінде көбейту бірінші операция, қосу – екінші операция болып табылады. Керішифрлау барысында алу (инверсиямен қосу) бірінші, ал бөлу (инверсиямен көбейту) екінші болып табылады.

SubBytes() және *InvSubBytes*() процедураларында матрицаларды қосу және көбейту аффинді типтегі түрлендіру және сызықты, $GF(2^8)$ байтты мультипликативті кері мәнге ауыстыру – сызықты емес. Бұл қадам барлық түрлендірулерді сызықты емес етеді.



8.13 сурет - *SubByte()* және *InvSubByte()* түрлендірулері

8.5.2. *ShiftRows(State)* және *InvShiftRows(Stat)* - күй матрицаның жолдарын жылжыту

ShiftRows(State) процедурасы сызықты түрлендіру қабатын жүзеге асырады. Түрлендірілетін блок жолдарында жылжудың мүмкін комбинацияларынан таңдау келесі критерийлер негізінде жасалды:

- барлық төрт жылжулар әр жол үшін әртүрлі болуы қажет және C0 = 0;

- қысқартылған дифференциалдарды қолданатын шабуылдарға беріктік (төменді, сонымен бірге [20]-ны қараңыз);

- ***Square*** шабуылына беріктік (төменде сипатталады, сонымен бірге [38] қараңыз);

- қарапайымдылық.

Қысқартылған дифференциалдарды қолданумен шабуылдар кейбір жылжу мәндерінің комбинациялары үшін бірден көп раунд

үшін тиімді болады. Басқа комбинацияларға ***Square*** типті шабуылдар тиімді. Жылжудың комбинациялар жиынынан 2 мен 3 тармақтарына жақсы қанағаттандыратын ең қарапайымы таңдалды.

Күйдің соңғы үш жолы циклдық түрде солға қарай байттардың әртүрлі санына жылжиды. *1* жол C_1 байтқа, 2 жол C_2 байтқа, және *3* жол C_3 байтқа жылжиды. *Rijndael* C_1, C_2 және C_3 жылжу мәндері *Nb* блогының ұзындығына тәуелді. Олардың шамалары 8.5 кестесінде көрсетілген.

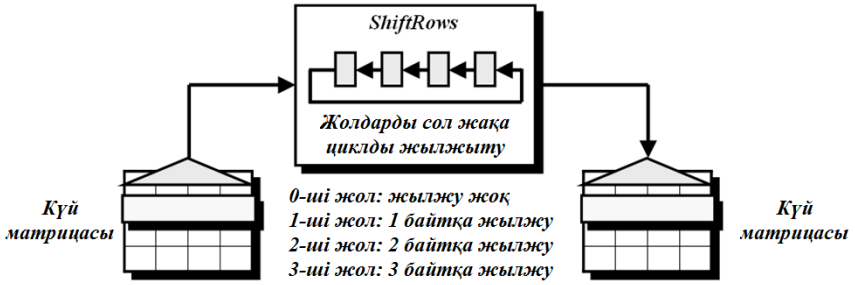
8.5-кесте

Әртүрлі ұзындығы бар деректер блоктары үшін жылжу шамалары

Nb	C_1	C_2	C_3
4	*1*	*2*	*3*
6	*1*	*2*	*3*
8	*1*	*3*	*4*

128 битке тең блоктың жалғыз өлшемі ($N_b = 4$) анықталған *AES* стандартында $C_1 = 1$, $C_2 = 2$ және $C_3 = 3$.

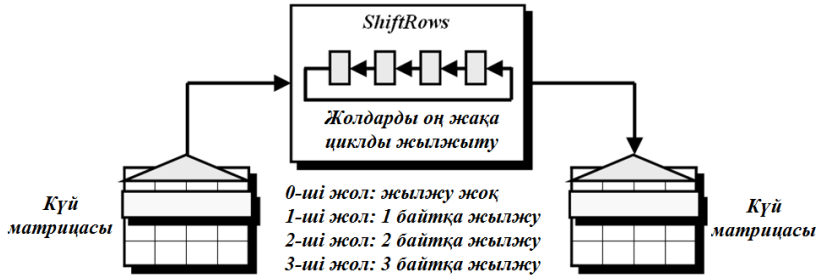
Күй матрицасының соңғы үш жолын жылжыту операциясы *ShiftRows(State)* деп белгіленген. 8.14 суретте күй матрицасы жолдарына түрлендіру әсерін көрсетеді.



8.14 сурет - Күй матрицасы жолдарына *ShiftRows()* түрлендіру әрекеті

ShiftRows() түрлендіруі бір уақытта тек бір ғана жолмен жұмыс істейді.

InvShiftRows(State) түрлендіруі *ShiftRows(State)* керітүрлендіру және *State* әр жолына r_i келесі ереже (8.15 сурет) бойынша әсер етеді: *State* бірінші жолы жылжымайды, яғни $C'_0 = 0$, күй матрицасының соңғы үш жолы оң жақа әртүрлі байтқа жылжиды; екінші жолы C'_1 байтқа жылжиды, үшіншісі – C'_2 байтқа және төртіншісі – C'_3 байтқа; *Rijndael* C'_1, C'_2 және C'_3 жылжу мәндері блок ұзындығына *Nb* тәуелді, оның шамалары 8.5 кестеде келтірілген.

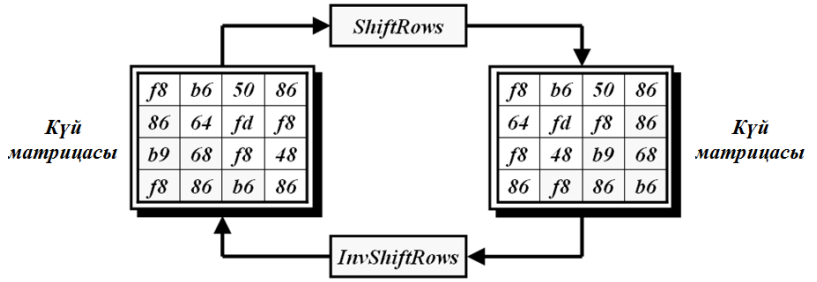


8.15 сурет - *InvShiftRows()* түрленуінің күй матрицасының жолдарына әсері

8.9 мысал. AES (*Nb = 4*) алгоритмінің *ShiftRows()* функциясының кірісіне төмендегіге тең күй матрицасы кіреді:

$$State = f886b9f8b664688650fdf8b686f84886_{16}.$$

8.16 суретінде күй матрицасы жолдарының *ShiftRows()* функциясы көмегімен солға және *InvShiftRows()* функциясы көмегімен оңға жылжуының нәтижесі көрсетілген.



8.16 сурет - Күй матрицасы жолдарына *ShiftRows()* және *InvShiftRows()* түрлендірулердің нәтижесі

8.5.3. *MixColumns(State)* және *InvMixColumns(State)* - күй матрицаның бағандарын араластыру

MixColumns(State) түрлендіруі сызықты түрлендіру қабатын жүзеге асырады және *State* блогында әр бағанға s_i *(i = 0,1,...,3)* әсер етеді, яғни әр машиналық сөзге, ереже бойынша

$$s'_i(x) = g(x) \cdot \otimes \ s_i(x) \bmod (x^4 + 1). \qquad (8.27)$$

Бұл түрлендіруде күй матрицасының бағандары (төрт байт) $GF(2^8)$ көпмүшесі ретінде қарастырылады және төмендегідей түрде көрсетілетін $g(x)$ көпмүшесіне $x^4 + 1$ модулі бойынша көбейтіледі [29, 38]:

$$g(x) = \{03\} \cdot x^3 + \{01\} \cdot x^2 + \{01\} \cdot x + \{02\}. \qquad (8.28)$$

$GF(2^8)$ өрісінің элементтері не биттік вектор ретінде, не байт ретінде жазылады. Мысалы, $\{03\}$ байты – бұл $x + 1$ көпмүшесі ретінде $m(x)$ модулі бойынша көрсетілетін $GF(2^8)$ өрісінің элементі.

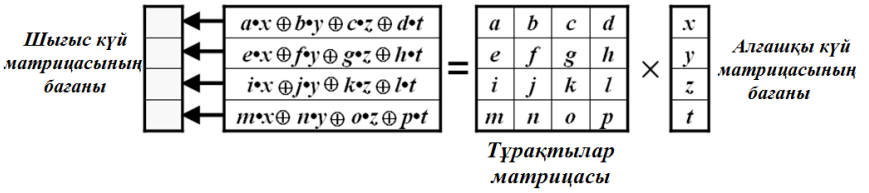
Көпмүшеге көбейту - сызықты операция болғандықтан, оны матрица әрекеті түрінде көрсетуге болады:

$$\begin{vmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{vmatrix} = C \cdot \begin{vmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \cdot \begin{vmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{vmatrix}, \qquad (8.29)$$

бұл жерде c – *State* ($c = 0,1,...,3$) массивінің баған нөмірі, C – $g(x)$ көпмүшесінің коэффициенттер матрицасы.

Бұл жерде көбейту нәтижесінде баған байттары $s_{o,c}$, $s_{1,c}$, $s_{2,c}$ және $s_{3,c}$ сәйкесінше келесі байттарға ауыстырылады:

$$\begin{aligned} s'_{0,C} &= \{02\} \bullet s_{0,C} \oplus \{03\} \bullet s_{1,C} \oplus s_{2,C} \oplus s_{3,C}; \\ s'_{1,C} &= s_{0,C} \oplus \{02\} \bullet s_{1,C} \oplus \{03\} \bullet s_{2,C} \oplus \bullet s_{3,C}; \\ s'_{2,C} &= s_{0,C} \oplus s_{1,C} \oplus \{02\} \bullet s_{2,C} \oplus \{03\} \bullet s_{3,C}; \\ s'_{3,C} &= \{03\} \bullet s_{0,C} \oplus s_{1,C} \oplus s_{2,C} \oplus \{02\} \bullet s_{3,C}. \end{aligned} \qquad (8.30)$$



8.17 сурет - *MixColumns()* әрекеті–матрица бағандарын араластыру

Күй матрицасының барлық төрт бағанына осы операцияны қолдану *MixColumns(State)* ретінде белгіленеді. 8.17 суреті күй матрицасының бағандарына *MixColumns()* түрленуін қолдануды көрсетеді.

MixColumns() шашырату функциясы *4 bytes → 4 bytes* мүмкінді сызықты түрлендірулерден келесі таңдау критерийлері негізінде таңдалды (күй матрицасы бағандарының төрт байты төрт байтқа):

1. Қайтымдық.
2. *gf(2^4)* өрісіндегі сызықтық, бұл *MixColumns()* функциясының қасиеті тұра керішифрлау алгоритмін қолдануға мүмкіндік береді.
3. Жеткілікті күшті шашырау.
4. *8*-разрядты процессорларда орындау жылдамдығы.
5. Симметриялық, яғни *MixColumns()* барлық деректермен бір түрде жұмыс істеу қажет.
6. Сипаттау қарапайымдылығы.

2, *5* және *6* $x^4 + 1$ модулі бойынша көпмүшелерді көбейтуді таңдауға шарттады. *1*, *3* және *4* критерийлері көбейткіштердің коэффициенттерін таңдауды анықтады. *4* критерийі коэффициенттерді таңдауды анықтады (артық көрінушілік ретінде) - *{ 00 }, { 01 }, { 02 }, { 03 }, ... { 00 }* мәні ешқандай түрлендіруді болжамайды, *{ 01 }* – көбейтуді жүргізу қажет емес, *{ 02 }* – көбейту *xtime()* функциясы көмегімен орындалу мүмкін және *{ 03 }* – *xtime()* функциясы және содан кейінгі *xor* көмегімен орындалады.

Үшінші критерий коэффициенттерді таңдаудың күрделі шарттарын анықтайды. Шифрды құрастырушылардың стратегиясы *MixColumns()* түрлендіруіне келесі талаптарды ескерді. *Z* байт векторларына орындалатын сызықты түрлендіру болсын, және вектордың байттық салмағы ондағы нөлдік емес байттар саны болсын. Онда оның шашырату күшін сипаттайтын сызықты түрлендірудің таралу коэффициенті деп келесі шаманы атаймыз:

$$min_{a \neq 0} \{ Z(a) + Z(F(a)) \},$$

бұл жерде *Z(a)* – байттық салмақ.

Нөлге тең емес байтты *белсенді байт* деп атаймыз. *MixColumns()* кіріс *32*-разрядты *State* массивінің бір белсенді байты кезінде шығыста максимум *4* белсенді байттар болады, себебі бұл функция бағандарды жеке түрлендіреді. Осылайша, таралу коэффициентінің жоғары шегі *5*-ке тең. *{ 01 }*, *{ 02 }* және *{ 03 }* коэффициенттері таралу коэффициентінің максималды мүмкін

мәнін алу үшін іріктелген. Оның 5-ке теңдігі кіріс деректердегі бір байтқа айырмашылығы шығыс деректерінің төрт байтына шашырайды (таралады), кіріс деректеріндегі екі байтқа айырмашылығы, аз дегенде, шығыс деректерінің үш байтына шашырайтындығын айтады. Оның үстіне, кіріс және шығыс деректер биттері арасында сызықты тәуелділік биттерді әрқашан да *MixColumns()* функциясының кірісі мен шығысында аз дегенде бес әртүрлі байттарға қозғайды деп айтуға болады.

8.5 мысал. AES (*Nb = 4*) алгоритмінің *MixColumns()* функциясының кірісіне келесіге тең күй матрицасы келсін:

$$State = d4bf5d30e0b452aeb84111f11e2798e5_{16},$$

сонымен бірге

$$\begin{array}{llll} s_{0,0} = d4; & s_{0,1} = e0; & s_{0,2} = b8; & s_{0,3} = 1e; \\ s_{1,0} = bf; & s_{1,1} = b4; & s_{1,2} = 41; & s_{1,3} = 27; \\ s_{2,0} = 5d; & s_{2,1} = 52; & s_{2,2} = 11; & s_{2,3} = 98; \\ s_{3,0} = 30; & s_{3,1} = ae; & s_{3,2} = f1; & s_{3,3} = e5. \end{array}$$

Күй матрицасы шығыста (8.29) өрнегімен анықталатындығы белгілі. (8.30) өрнегін қолданып, c = *0* жағдайы үшін күй матрицасының бағанын шығысында анықтаймыз. Бұл жағдайда

$$s'_{0,0} = \{02\} \bullet s_{0,0} \oplus \{03\} \bullet s_{1,0} \oplus s_{2,0} \oplus s_{3,0} =$$

$$= \{02\} \bullet \{d4\} \oplus \{03\} \bullet \{bf\} \oplus \{5d\} \oplus \{30\} =$$

$$= \{02\} \bullet \{d4\} \oplus \{02\} \bullet \{bf\} \oplus \{bf\} \oplus \{5d\} \oplus \{30\}.$$

$\{d4\}$, $\{bf\}$, $\{5d\}$ және $\{30\}$ мәндерін көпмүше түрінде көрсетеміз:

$$\{d4\} \rightarrow x^7 + x^6 + x^4 + x^2,$$

$$\{bf\} \rightarrow x^7 + x^5 + x^4 + x^3 + x^2 + x + 1,$$

$$\{5d\} \rightarrow x^6 + x^4 + x^3 + x^2 + 1,$$

$$\{30\} \rightarrow x^5 + x^4.$$

$\{02\} \bullet \{d4\}$ көбейтіндісін есептейміз:

$$\{02\}\bullet\{d4\} \rightarrow x \cdot (x^7 + x^6 + x^4 + x^2) \bmod p(x) =$$
$$= (x^8 + x^7 + x^5 + x^3) \bmod (x^8 + x^4 + x^3 + x + 1) =$$
$$= x^7 + x^5 + x^4 + x + 1.$$

$\{02\}\bullet\{bf\}$ көбейтіндісін есептейміз:

$$\{02\}\bullet\{bf\} \rightarrow x \cdot (x^7 + x^5 + x^4 + x^3 + x^2 + x + 1) \bmod p(x) =$$
$$= (x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x) \bmod (x^8 + x^4 + x^3 + x + 1) =$$
$$= x^6 + x^5 + x^2 + 1.$$

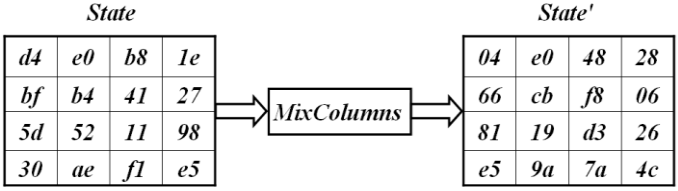
$s'_{0,0}$ элементін есептейміз:

$$s'_{0,0} = (x^7 + x^5 + x^4 + x + 1) \oplus (x^6 + x^5 + x^2 + 1) \oplus$$
$$\oplus (x^7 + x^5 + x^4 + x^3 + x^2 + x + 1) \oplus (x^6 + x^4 + x^3 + x^2 + 1) \oplus$$
$$\oplus (x^5 + x^4) = x^2 \rightarrow \{04\}.$$

$s'_{1,0}$, $s'_{2,0}$ және $s'_{3,0}$ мәндерін сәйкесінше есептеп, келесіні аламыз:

$$s'_{1,0} = x^6 + x^5 + x^2 + x \rightarrow \{66\};$$
$$s'_{2,0} = x^7 + 1 \rightarrow \{81\};$$
$$s'_{3,0} = x^7 + x^6 + x^5 + x^2 + 1 \rightarrow \{e5\}.$$

c = 1, 2 және *3* кезінде күй матрицасының бағандар мәндерін көпмүше түрінде көрсете отырып және сәйкесінше түрлендіру жүргізіп *MixColumns()* функциясының шығысында күй матрицасын келесі түрде аламыз (8.18 сурет):



8.18 сурет - Күй матрицасы бағандарын *MixColumns()* түрлендіру нәтижесі

MixColumns() түрленудіруінің (8.29)-ға кіретін коэффициенттер матрицасы *C* $GF(2^8)$ құлдырамаған, сондықтан *MixColumns()*

операциясы қайтымды, ал оған кері әрекет бастапқы C-ға кері C^{-1} матрицасымен жүзеге асырылады.

InvMixColumns(State) түрлендіруінде күй бағандары $GF(2^8)$ көпмүше ретінде қарастырылады және келесі көпмүшеге $x^4 + 1$ модулі бойынша көбейтіледі:

$$s'(x) = (s(x) \otimes g^{-1}(x)) \bmod (x^4 + 1),$$

бұл жерде $g^{-1}(x) = \{0b\} \cdot x^3 + \{0d\} \cdot x^2 + \{09\} \cdot x + \{0e\}$ төмендегі теңдеудің шешімімен алынады:

$$g(x) \otimes g^{-1}(x) \bmod (x^4 + 1) = 1 \qquad (8.31)$$

$g(x) = \{03\} \cdot x^3 + \{01\} \cdot x^2 + \{01\} \cdot x + \{02\}$ кезде $g^{-1}(x)$ қатысты.

Бұл матрицалық түрде келесідей көрсетілуі мүмкін:

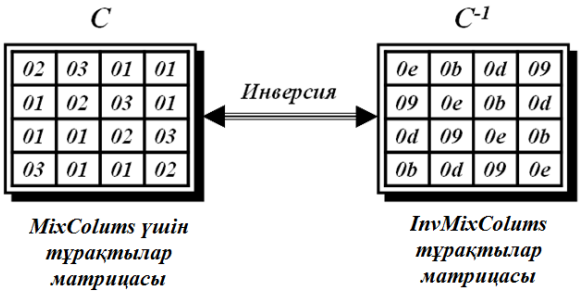
$$\begin{vmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{vmatrix} = C^{-1} \cdot \begin{vmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{vmatrix} = \begin{vmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{vmatrix} \begin{vmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{vmatrix},$$

бұл жерде $c - C^{-1}$ ($c = 0,1,...,3$) матрицаның баған нөмірі.

Осындай көбейту нәтижесінде баған байттары $s_{o,c}$, $s_{1,c}$, $s_{2,c}$ және $s_{3,c}$ сәйкесінше байттарға ауыстырылады:

$$s'_{0C} = (\{0e\} \bullet s_{0C}) \oplus (\{0b\} \bullet s_{1C}) \oplus (\{0d\} \bullet s_{2C}) \oplus (\{09\} \bullet s_{3C});$$
$$s'_{1C} = (\{09\} \bullet s_{0C}) \oplus (\{0e\} \bullet s_{1C}) \oplus (\{0b\} \bullet s_{2C}) \oplus (\{0d\} \bullet s_{3C});$$
$$s'_{2C} = (\{0d\} \bullet s_{0C}) \oplus (\{09\} \bullet s_{1C}) \oplus (\{0e\} \bullet s_{2C}) \oplus (\{0b\} \bullet s_{3C});$$
$$s'_{3C} = (\{0b\} \bullet s_{3C}) \oplus (\{0d\} \bullet s_{0C}) \oplus (\{09\} \bullet s_{1C}) \oplus (\{0e\} \bullet s_{2C}).$$

8.17 суреті *MixColumns()* мен *InvMixColumns()* түрлендірулері үшін қолданылатын тұрақтылар матрицаларын көрсетеді.



8.19 сурет - *MixColumns()* мен *InvMixColumns()* түрлендірулері үшін қолданылатын тұрақтылар матрицалары

Егер элементтері $GF(2^8)$ коэффициенттерімен 8 биттен (немесе көпмүшелер) тұратын сөз ретінде түсіндірілсе, бұл екі матрица бір біріне кері [29, 38].

MixColumns() түрленуі баған деңгейінде жұмыс жасайды; ол күй матрицасының әр бағанын жаңа бағанға айналдырады. Бұл түрлендіру – шын мәнінде күй матрицасының бағанының және тұрақтылар төртбұрыш матрицасының матрицалық көбейтіндісі. Күй матрицасы бағаны мен тұрақтылар матрицасындағы байттар $GF(2^8)$-те коэффициенттерімен 8 биттен (немесе көпмүшелер) тұратын сөз ретінде түсіндіріледі. Байттарды көбейту $GF(2^8)$ $p(x) = x^8 + x^4 + x^3 + x + 1$ келтірілмеген көпмүшемен орындалады. Қосу – бұл *8* бит сөздерге *xor* операциясын қолдану.

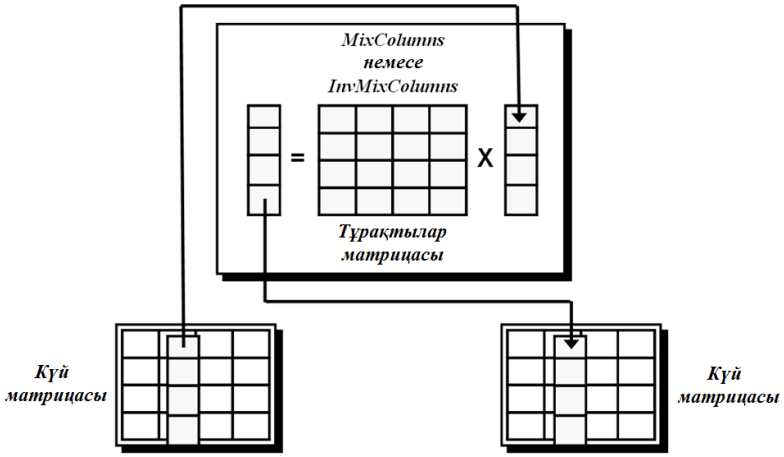
InvMixColumns() түрлендіруі *MixColumns()*-түрлендіруіне ұқсас. 8.20 сурет *MixColumns()* немесе *InvMixColumns()* қолданумен түрлендіру принципін көрсетеді.

8.11 мысал. *MixColumns()* түрлендіру кірісіне келесі күй матрицасы кірсін:

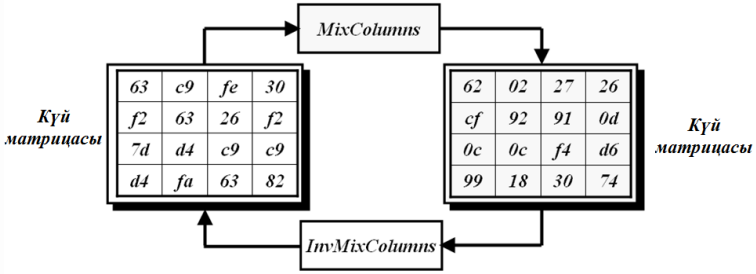
$$State = 63f27dd4c963d4fafe26c96330f2c982_{16}.$$

8.21 суреті егер *MixColumns()* түрлендіруін қолданса, күй матрицасы қалай өзгеретінін көрсетеді. Сурет сонымен бірге *InvMixColumns()* түрлендіруі күй матрицасының алғашқы мәнін құратындығын көрсетеді.

Ескі күй матрицасында өзара тең байттар жаңа күй матрицасында тең болмайды. Мысалы, *f2* екі байт екінші жолда *cf* және *0d* өзгертілген.



8.20 сурет - *MixColumns()* немесе *InvMixColumns()* қолданумен түрлендіру принципі



8.21 сурет - 8.11 мысалы үшін *MixColumns()* және *InvMixColumns()* түрлендірулері

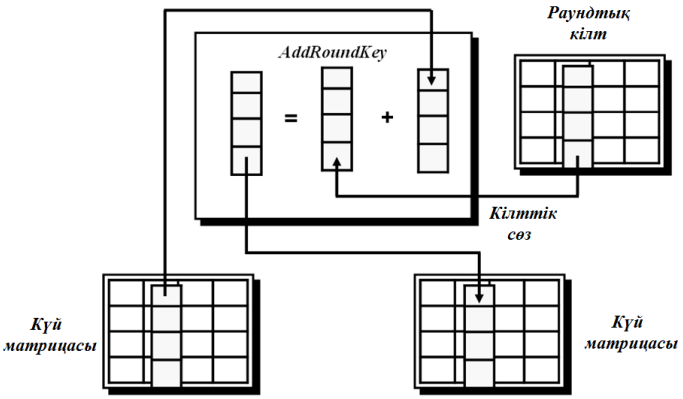
8.5.4. *AddRoundKey(State, RoundKey)* күй матрицамен раундтық кілтті қосу

Ең маңызды түрлендіру - шифр кілті бар түрлендіру болуы ықтимал. Алдынғы барлық түрлендірулер кері және белгілі алгоритмдерді қолданады. Бұл жағдайда құпияны сақтайтын тек шифр кілті болады.

AddRoundKey() бір уақыт сәтінде бір бағанды өңдейді. *MixColumns()* ұқсас түрлендіру тұрақтылардың төтрбұрыш

матрицасын күй матрицасының әр бағанына көбейтеді. *AddRoundKey()* раундтың кілттік сөзін күй матрицасының әр бағанымен қосады. *MixColumns()* матрицалық көбейтуді, *AddRoundKey()* – қосу және алу операцияларын қолданады. Шектелген өрісте қосу және алу сондай болып қалатындықтан, *AddRoundKey()* өз өзіне кері. 8.22 сурет *AddRoundKey()* түрлендіру принципін көрсетеді.

AddRoundKey(State, RoundKey) түрлендіруі өңделетін деректерді (күйлерді) раундтық кілтпен қосу текшесін жүзеге асырады. Берілген операцияда раундтық кілт күй матрицасына қарапайым разряд бойынша *xor* көмегімен қосылады. Раундтық кілт *W* шифрлау кілтінен *K* кілттерді өндіру алгоритмімен өндіріледі. Раундтық кілт (*32*-разрядты сөздерде) ұзындығы блок ұзындығына *Nb* тең.



8.22 сурет - *AddRoundKey()* түрлендіру принципі

8.12 мысал. AES (*Nb = 4*) алгоритмінің *AddRoundKey()* функциясының кірісіне келесі келсін:

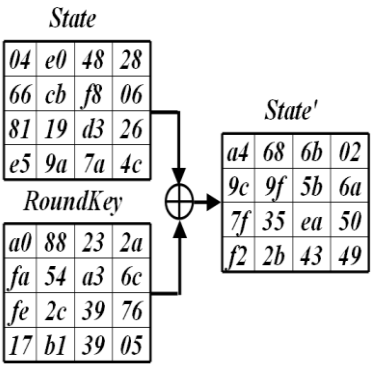
- күй матрицасы:

$$State = d4bf5d30e0b452aeb84111f11e2798e5_{16};$$

- раундтық кілт матрицасы:

$$RoundKey = 046681e5e0cb199a48f8d37a2806264c_{16}.$$

Күй матрицасы шығысында күй матрицасының және раундтық кілт матрицасының бағандарын *2* модулі бойынша разрядтармен қосу (*xor* операциясын орындау) арқылы анықталатыны белгілі.



8.23 сурет - 8.12 мысал үшін *AddRoundKey()* күй матрицасының байттарына әрекет

Осындай әрекеттерді күй матрицасының және раундтық кілт матрицасының бағандарына орындап *AddRoundKey(State, Round Key)* функциясының шығысындағы күй матрицасын аламыз (8.23 сурет).

Байттың мәнін өзгертетін *SubByte()* түрлендірумен орындалатын ауыстыру тек алғашқы мәнге және кестеге кірісте негізделген; үрдіске көрші байттар кірмейді. *SubByte()* – ішкі байттық түрлендіру деп айтуға болады. *ShiftRows()* – түрлендірумен орындалатын алмастыру байттарды орындармен ауыстырады, бірақ байттардағы биттерді алмастырмайды.

ShiftRows() – байттармен алмасу түрлендіруі деп айтуға болады. Енді бізге байттардағы биттерді ауыстыратын және көрші байттардағы биттерге негізделген ішкі байттық түрлендіру қажет. Разрядтық деңгейде шашыратуды қамтамасыз ету үшін байттарды араластыру қажет.

MixColumns() араластыру түрлендіруі әр байттың мазмұнын өзгертеді, төрт жаңа байтты алу үшін бір уақытта төрт байтты аударады және біріктіреді. Әр жаңа байттың басқасынан айырмашылығы болатынын (барлық төрт байт тұра сондай болса да) кепілдеу үшін, үрдіс алдымен әр байтты тұрақтылардың әртүрлі жинағына көбейтеді және оларды араластырады. Араластыру матрицалық көбейтумен қамтамасыз етілу мүмкін. Төртбұрышты матрицаны баған-матрицаға көбейту кезінде, нәтиже - жаңа баған-матрицасы. Матрицаны күй матрицасындағы жол мәндеріне көбейтуден кейін жана матрицадағы әр элемент ескі матрицаның барлық төрт элементінен тәуелді болады.

8.6. AES ДЕРЕКТЕРДІ ШИФРЛАУ ҮШІН КІЛТТІ ЖАЮ АЛГОРИТМІ

Кілтті жаю алгоритмі раундтық кілттерді *W* шифрлаудың алғашқы кілтінен *K* алу ретін анықтайды. Раундтық кілттер шифрлау кілтінен кілттерді өндіру алгоритмі көмегімен алынады. Ол екі компоненттен тұрады:

- кілтті кеңейту (*Key Expansion*);
- раундтық кілтті таңдау (*Round Key Selection*).

Алгоритмнің негізқұрайтын принциптері келесі:

- раундтық кілттердің жалпы саны блок ұзындығын раундтар санына көбейтуге және *1* қосуға тең (мысалы, блок ұзындығы *128* бит және *10* раунд үшін *1408* бит раундтық кілттер қажет);
- шифрлау кілті кеңейтілген кілтке жайылады (*Expanded Key*);
- раундтық кілттер кеңейтілген кілттен келесі ретпен алынады: бірінші раундтық кілтте бірінші *Nb* сөздер, екіншіде - келесі *Nb* сөздер болады және ары қарай.

8.6.1. Кілтті кеңейту *(KeyExpansion)*

Әр раунд үшін кілтті құру үшін, *AES* кілттік кеңейту үрдісін қолданады. Егер раундтар саны *Nr* болса, онда кілттерді кеңейту процедурасы бір *128*-биттік шифрлау кілтінен әр қайсысы *128* биттен тұратын *Nr + 1* раундтық кілтті құрады. Бірінші раундтық кілттер раундтар алдындағы (*0* раунд) түрлендіру үшін *AddRoundKey()* түрлендіруін қолданумен қолданылады; қалған раундтық кілттер болашақ *AddRoundKey()* түрлендірулері үшін әр раундтың соңында қолданылады.

Кілттерді кеңейту процедурасы сөзден кейін сөзді құрады, олардан ары қарай раундтық кілттер қалыптастырылады. Сөз - төрт байттан тұратын массив. Кілттерді кеңейту процедурасы $4 \times (Nr + 1)$ сөздерді құрайды, олар келесідей белгіленеді

$$W = w_0, w_1, w_2 \ldots, w_{4 \times (Nr + 1)}, w_{4 \times (Nr + 1) - 1}.$$

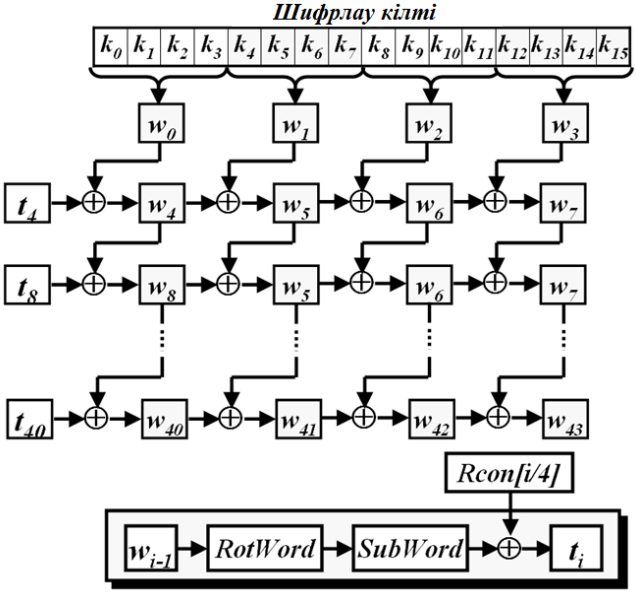
Басқа сөзбен, *AES-128* (*Nr = 10*) нұсқасында *W = 44* сөз болады; *AES-192* (*Nr = 12*) нұсқасында – *W = 52* сөз; ал *AES-256* (*Nr = 14*) нұсқасында - *W = 60* сөз. Әр раундтық кілтте төрт сөз болады. 8.6 кестесі раундтар және сөздер арасындағы қатынасты көрсетеді.

8.6 кесте

AES алгоритмінің әр раунды үшін сөздер

Раунд	*Сөздер*			
0	w_0	w_1	w_2	w_3
1	w_4	w_5	w_6	w_7
2	w_8	w_9	w_{10}	w_{11}
. . .	. . .	. . .	. . .	. . .
Nr	$w_{4 \times Nr}$	$w_{4 \times Nr+1}$	$w_{4 \times Nr+2}$	$w_{4 \times Nr+3}$

AES-128 нұсқасы үшін кілтті кеңейту процессін қарастырамыз; басқа нұсқалары үшін процесстер кейбір үлкен емес өзгертулерге қарамағанда тұра осындай. 8.24 сурет *128* биттік алғашқы кілттен *44* сөзді қалай алатынын көрсетеді.



8.24 сурет - *AES-128* кілттерді кеңейту процессін түсіндіру

AES-128 кілтті кеңейту процессі келесі:

1. Бірінші төрт сөз (w_0, w_1, w_2, w_3) шифрлау кілтінен шығады. Шифрлау кілті *16* байттық (k_0 ден k_{15} дейін) массив ретінде көрсетілген. Бірінші төрт байт (k_0 ден k_3 дейін) w_0 болады; келесі төрт

байт (k_4 ден k_7 дейін) w_1 болады және ары қарай. Басқа сөзбен, бұл топта сөздерді тізбекті жалғау (конкатенация) шифрлау кілтін көшіреді.

2. Қалған w_i $i = 4$ ден $i = 43$ дейін сөздердің бөліктері келесідей алынады:

а) егер $(i \bmod 4) \neq 0$, онда $w_i = w_{i-1} \oplus w_{i-4}$ (8.24 суретке сәйкес әр сөз бір сол жақтағыдан және бір жоғарғысынан алынады);

б) егер $(i \bmod 4) = 0$, онда $w_i = t_i \oplus w_{i-4}$ (бұл жерде t_i – уақытша сөз, w_{i-1} сөзімен *SubWord* және *RotWord* екі процессті және раунд тұрақтысымен *Rcon[i/4] xor* операциясын қолдану нәтижесі).

Басқа сөзбен

$$t_i = SubWord(RotWord(w_{i-4})) \oplus Rcon[i/4]. \qquad (8.32)$$

RotWord (*Rotate Word*) – тек бір бағанға қолданатын *ShiftRows()* түрлендіруіне ұқсас процедура. Процедура сөзді төрт байттан тұратын массив ретінде қабылдайды және әр байтты жоғары жақа айырбастаумен (байттарды циклды жылжыту) жылжытады.

SubWord (*Substitute Word*) – тек бір бағанға қолданатын *SubBytes()* түрлендіруіне ұқсас процедура. Процедура сөздегі әр байтты қабылдайды және басқамен ауыстырады.

Раундтың әр тұрақтысы *Rcon* (*RoundConstants*) – бұл *4*-байттық мән, оның ішінде шеткі оң жақтағы (үлкен) үш байт әрқашан нөлдік.

Раундтың тұрақтысының *Rcon[i/4]* мәні келесі өрнекпен анықталады

$$Rcon[i/4] = R_{con}[j] = (2^{j-1}) \bmod p(x). \qquad (8.33)$$

(8.33) өрнегі бойынша есептелген *Rcon* раундтық тұрақтылар мәндері 8.7 кестесінде келтірілген.

8.7 кесте

Раунд тұрақтыларының *Rcon* (*Nk = 4*) мәндері

i	*i/Nk*	*Rcon[i/Nk]*	*i*	*i/Nk*	*Rcon[i/Nk]*
4	*1*	*{ 01000000 }*	*24*	*6*	*{ 20000000 }*
8	*2*	*{ 02000000 }*	*28*	*7*	*{ 40000000 }*
12	*3*	*{ 04000000 }*	*32*	*8*	*{ 80000000 }*
16	*4*	*{ 08000000 }*	*36*	*9*	*{ 1b000000 }*
20	*5*	*{ 10000000 }*	*40*	*10*	*{ 36000000 }*

Кілтті кеңейту процедурасы сөздерді есептеген кезде жоғарыда көрсетілген кестені немесе шеткі сол жақтағы биттерді динамикалық түрде төменде көрсетілгендей есептеу кезінде $GF(2^8)$ өрісін қолдана алады.

$$Rcon[i = 4/Nk]|_{Nk=4} = Rcon[1] = 2^{1-1} \bmod p(x) = 01_{16};$$
$$Rcon[i = 8/Nk]|_{Nk=4} = Rcon[2] = 2^{2-1} \bmod p(x) = 02_{16};$$
$$Rcon[i = 12/Nk]|_{Nk=4} = Rcon[3] = 2^{3-1} \bmod p(x) = 04_{16};$$
$$Rcon[i = 16/Nk]|_{Nk=4} = Rcon[4] = 2^{4-1} \bmod p(x) = 08_{16};$$
$$Rcon[i = 20/Nk]|_{Nk=4} = Rcon[5] = 2^{5-1} \bmod p(x) = 10_{16};$$
$$Rcon[i = 24/Nk]|_{Nk=4} = Rcon[6] = 2^{6-1} \bmod p(x) = 20_{16};$$
$$Rcon[i = 28/Nk]|_{Nk=4} = Rcon[7] = 2^{7-1} \bmod p(x) = 40_{16};$$
$$Rcon[i = 32/Nk]|_{Nk=4} = Rcon[8] = 2^{8-1} \bmod p(x) = 80_{16};$$
$$Rcon[i = 36/Nk]|_{Nk=4} = Rcon[9] = 2^{9-1} \bmod p(x) = 1b_{16};$$
$$Rcon[i = 40/Nk]|_{Nk=4} = Rcon[10] = 2^{10-1} \bmod p(x) = 36_{16}.$$

Rcon[i] деп белгіленген байт - бұл x^{i-1}, бұл жерде i – раунд нөмірі.

Кілтті жаю процессін (*Key Expansion*) көрсететін бағдарламаның псевдокоды төменде келтірілген.

```
//================================================
//= Кілтті жаю функциясының псевдокоды KeyExpansion() =
//================================================
KeyExpansion (byte key[4*Nk], word w[Nb*(Nr + 1)], Nk)
begin
    word temp
    i = 0
    while (i < Nk)
        w[i] = word(key[4*i], key[4*i+l], key[4*i+2], key[4*i+3])
        i = i+1
    end while
    i = Nk
    while (i < Nb (Nr+1))
        temp = w[i-l]
        if (i mod Nk = 0)
                temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
                else if (Nk > 6 and i mod Nk = 4) temp = SubWord(temp)
        end if
        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
```

```
end
//=============================================
```

Ескерту. Бұл жерде толық сипаттау мақсатымен барлық мүмкінді кілттер ұзындығы үшін алгоритм келтірілген, тәжірибеде оның толық жүзеге асырылуы көп жағдайларда қажет емес.

Бірінші *Nk* сөзде шифрлау кілті болады. Барлық қалған сөздер кіші индекстері бар сөздерден рекурсивті анықталады. Кілттерді қалыптастыру алгоритмі *Nk* шамасынан тәуелді.

8.13 мысал. AES (*Nb = 4, Nk = 4*) шифрлау алгоритмінің *KeyExpansion()* функциясының кірісіне шифрлау кілті *K* келсін:

Кеңейтілген кілттің төртбайттық сөздерін қалыптастыру: w_4, w_5, w_6 және w_7.

w_0	w_1	w_2	w_3
2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

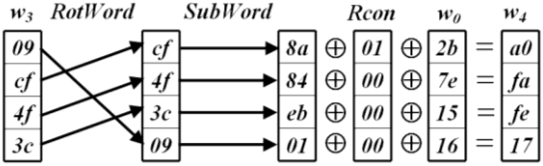
Шешім. w_4 сөзін қалыптастырамыз. Келесі болғандықтан

$$i \bmod N_k = 4 \bmod 4 = 0,$$

онда w_4 сөзі (8.32) қолданумен қалыптастырылады

$$w_4 = SubWord(RotWord(w_3)) \oplus w_0 \oplus R_{con}[1]$$

немесе



w_5, w_6 және w_7 сөздерін қалыптастырамыз. $i = 5, 6, 7$ кезінде $i \bmod Nk \neq 0$ болғандықтан, сөздер (8.32) қолданумен келесі түрде қалыптастырылады:

$$w_5 = w_4 \oplus w_1; \; w_6 = w_5 \oplus w_2; \; w_7 = w_6 \oplus w_3$$

немесе

w_4		w_1		w_5
a0	⊕	28	=	88
fa	⊕	ae	=	54
fe	⊕	d2	=	2c
17	⊕	a6	=	b1

w_5		w_2		w_6
88	⊕	ab	=	23
54	⊕	f7	=	a3
2c	⊕	15	=	39
b1	⊕	88	=	39

w_6		w_3		w_7
23	⊕	09	=	2a
a3	⊕	cf	=	6c
39	⊕	4f	=	76
39	⊕	3c	=	05

Сондықтан, шифрлау кілтінің *K* төртбайттық сөздерін және қалыптастырылған төртбайттық сөздерді есепке алып w_4, w_5, w_6 және w_7, кеңейтілген кілттің сөздері келесі түрде болады

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7
2b	28	ab	09	a0	88	23	2a
7e	ae	f7	cf	fa	54	a3	6c
15	d2	15	4f	fe	2c	39	76
16	a6	88	3c	17	b1	39	05

8.6.2. Раундтық кілтті таңдау *(Round Key Selection)*

Раундтық кілт *i*-ші раунд үшін 8.25 суретінде көрсетілгендей w_{Nb-i}-дан $w_{Nb(i+1)-1}$ дейін раундтық кілт массивінің сөздерінен алынады.

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	. . .
Раундтық кілт 0				*Раундтық кілт 1*				*Раундтық кілт 2*				. . .	

8.25 сурет - *Nk = 4* үшін кілтті кеңейту және раундтық кілтті таңдау процедуралары

8.14 мысал. AES (*Nb = 4, Nk = 4*) алгоритмінің *KeyExpansion()* функциясын орандаудан кейін кеңейтілген кілттің келесі төртбайттық сөздері алынсын:

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}
2b	28	ab	09	a0	88	23	2a	f2	7a	59	73
7e	ae	f7	cf	fa	54	a3	6c	c2	96	35	59
15	d2	15	4f	fe	2c	39	76	95	b9	80	f6
16	a6	88	3c	17	b1	39	05	f2	43	7a	7f

Деректерді шифрлаудың нөлінші, бірінші және басқа раундтар үшін раундтық кілттерді қалыптастыру.

Шешім. Нөлінші раунд үшін раундтық кілт (K_0) раундтық кілт массивінің сөздерінен шығады $w_{Nb \cdot 0} = w_0$ ден $w_{Nb \cdot (0+1)-1} = w_3$ дейін. Бірінші раунд үшін раундтық кілт (K_1) раундтық кілт массивінің сөздерінен шығады $w_{Nb \cdot 1} = w_4$ ден $w_{Nb \cdot (1+1)-1} = w_7$ дейін. Екінші раунд үшін раундтық кілт (K_2) раундтық кілт массивінің сөздерінен шығады $w_{Nb \cdot 2} = w_8$ ден $w_{Nb \cdot (2+1)-1} = w_{11}$ дейін. Сонымен, раундтық кілттер оны 8.25 суретте көрсетілген ұсынуды есепке алумен келесі түрде болады

Раундтық кілт 0			
2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Раундтық кілт 1			
a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

Раундтық кілт 2			
f2	7a	59	73
c2	96	35	59
95	b9	80	f6
f2	43	7a	7f

Ескерту. Кілтерді өндіру алгоритмін *w[i]* массивін қолданусыз да орындауға болады. Жадыға қойылатын талап маңызды болатын жүзеге асыруларда раундтық кілттер *Nk* сөздерден тұратын буферлерді қолдану арқылы *"орындау уақытында"* есептелу мүмкін. Кеңейтілген кілт әрқашан шифрлау кілтінен жасалу керек және тікелей көрсетілмейді. Шифрлау кілтін таңдауға ешқандай шектеулер жоқ.

Кілтті жаю алгоритмі келесі шабуылдар түрлеріне беріктілікті қамтамасыз ету керек:

- шифрлаудың алғашқы кілтінің бөлігі криптографиялық талдаушыға белгілі шабуылдарға;
- шифрлау кілті алдын ала белгілі немесе таңдалуы мүмкін шабуылдарға, мысалы, егер шифр хэштеу кезінде деректерді қысу үшін қолданылса;
- "эквивалентті кілттер" шабуылдарға [38]; бұндай шабуылдарға беріктіліктің қажетті шарты екі әртүрлі алғашқы шифрлау кілттерінен өте үлкен бірдей раундтық кілттер жинағының болмауы.

Кілтті жаю процедурасы сонымен қатар келесілерді аластатуда маңызды рөлді атқарады:

- бірраундты түрлендірудің симметриясын, ол барлық кіріс байттарды біртүрде өңдейді; оны болдырмау үшін кілтті жаю процедурасында раундтық кілттің әр бірінші *32*-разрядты сөзді алу

үшін *SubWord(RotWord())* функциясы және раундтық тұрақты қолданылады;

- раунд арасындағы симметрияны (раундтық түрлендіру цикл түрлендірудің барлық итерацияларына бірдей); оны бұзу үшін кілтті жаю алгоритміне тұрақтылар енгізілген, олардың мәндері әр раунд үшін әртүрлі.

Сонымен, жаю алгоритмі келесі критерийлерге сәйкес таңдалған:

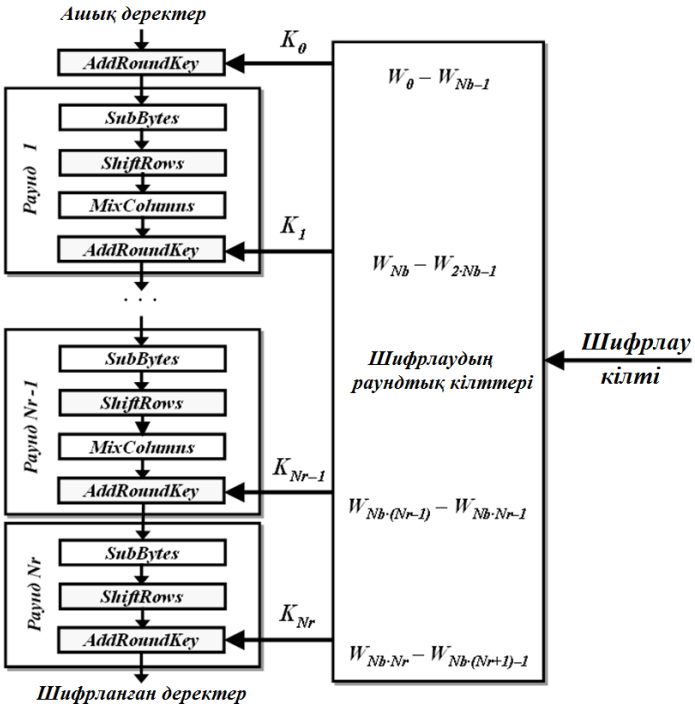
- қолданатын түрлендірулердің керігі, яғни кеңейтілген кілттің кез келген *Nk* тізбекті сөздерінен толық кеңейтілген кілтті бірмәнді қалпына келтіруге болады;
- әртүрлі процессорлар типтерінде жақсы жылдамдық;
- симметриялықты азайту үшін раундтық тұрақтылардың бар болуы;
- қалыптасатын раундтық кілтке шифрлаудың алғашқы кілтінде өзгерістердің жақсы шашырауы;
- раундтық немесе алғашқы кілттің биттерінің бөлігін білу негізінде қалған биттерін есептеу мүмкіндігін болдырмау;
- кеңейтілген кілттің биттер арасындағы тәуелдікті алғашқы кілтте осындай тәуелдікті білу негізінде толық анықтаудың алдын алу үшін жеткілікті сызықтық еместігі;
- сипаттаудың қарапайымдылығы.

8-разрядты процессорларда тиімді жүзеге асырылуы мүмкіндігі бар алгоритмнің байтқа-бағытталған сұлбасы таңдалды. *SubBytes()* қолдану түрлендірудің сызықты еместігін қамтамасыз етеді және *8*-разрядты процессорларда үлкен емес қосымша жады кеңістігін талап етеді.

Керішифрлау үшін раундтық кілттер шифрлаудың раундтық кілттеріне қатынасты кері ретпен қолданылады.

8.7. AES ДЕРЕКТЕРДІ ШИФРЛАУ

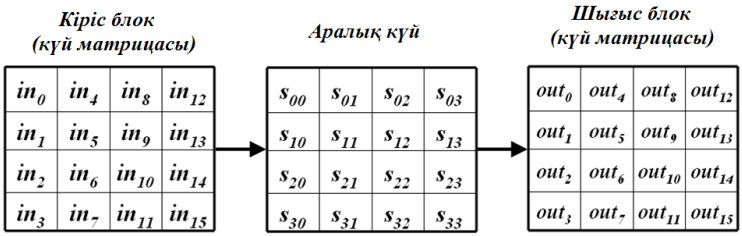
AES (*Rijndael)* шифрының құрылымы келесіден тұрады (8.26 сурет):



8.26 сурет - *AES* (*Rijndael*) криптографиялық алгоритмінде шифрлау процедурасының сұлбасы

- раундтық кілттің алғашқы қосылуынан;
- деректерді шифрлаудың $Nr-1$ раундынан;
- *MixColumns()* операциясы жоқ деректерді шифрлаудың соңғы раундынан.

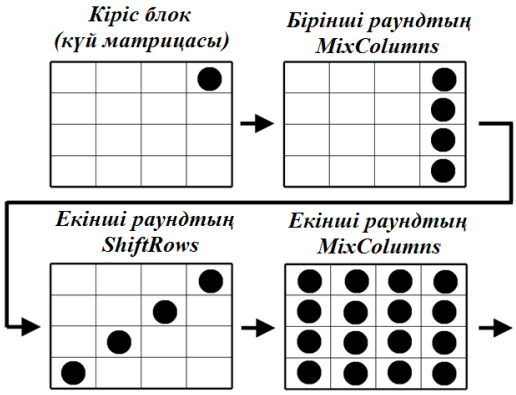
Алгоритмнің кірісіне деректер блоктары *State* (ашық деректер) жіберіледі, түрлендіру барысында блоктар мазмұны өзгереді және шығыста шифрланған деректер болады, оларда 8.27 суретінде көрсетілгендей *State* блоктары түрінде ұйымдастырылады, бұл жерде $Nb = 4$, in_m және out_m – кіріс және шығыс блоктарының m-ші байттары, $m = 0,1,...,15$, $s_{i\ j}$ – *State* массивінің i-ші жол және j-ші баған қыйылысындағы байт, $i = 0,1,...,3$ және $j = 0,1,...3$.



8.27 сурет - *State* блок түрінде ұйымдастырылған деректерді түрлендіру жолы

Деректерді шифрлаудың бірінші раунд алдында алғашқы шифрлау кілтімен *2* модулі бойынша қосу орындалады (раундтық кілт *0*), содан кейін - кілт ұзындығына тәуелді *10*, *12* немесе *14* раунд барысында күй матрицасы түрлендіріледі. Соңғы раундта *MixColumns()* бағандарда байттарды араластыру функциясы орындалмайды. Бірінші қарағанда бұл шифрдың құрылымын нашарлататын түсініксіз шешім болып көрінеді. Бірақ бұл олай емес.

8.28 сурет шифрдың шашырату және араластыру қасиеттерін көрсетеді. Екі раунд толық шашырату мен араластыруды қамтамасыз ететінін көрсетеді.



8.28 сурет - *AES* (*Rijndael*) криптографиялық алгоритмінің жұмыс істеу принципі: ● – өзгертілген байт

AES (*Rijndael*) – *Cipher* криптографиялық алгоритмімен деректерді шифрлау процессін көрсететін бағдарламаның псевдокоды төменде келтірілген.

```
//===========================================
//===== Деректерді шифрлау процедурасы псевдокодта ===
//===========================================
Cipher (byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey(state, w[0, Nb-1])
    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
    out = state
end
//===========================================
```

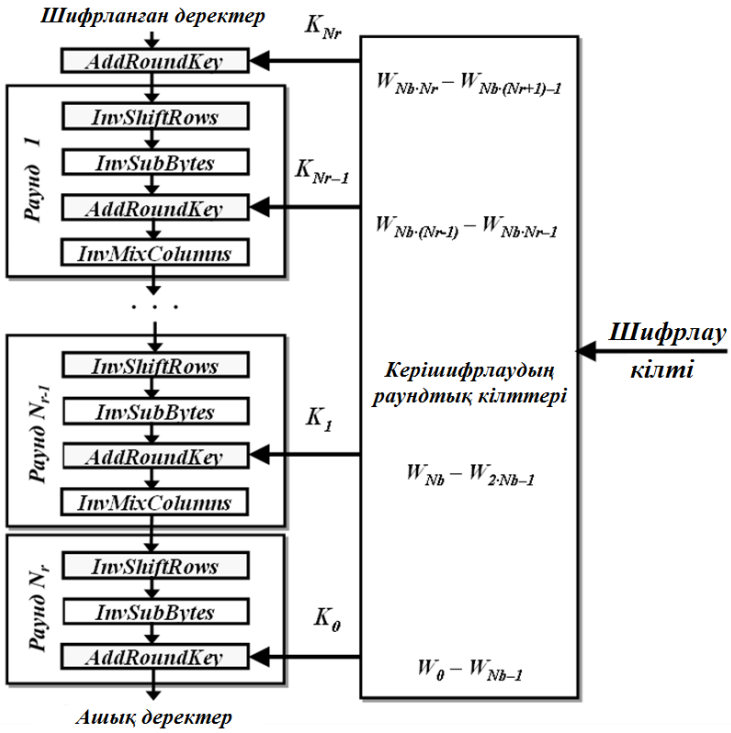
8.8. AES ДЕРЕКТЕРДІ КЕРІШИФРЛАУ

8.8.1. Деректерді қайта (инверсті) керішифрлау

Шифррау алгоритмінің *SubBytes(), ShiftRows(), MixCo-lumns()* және *AddRoundKey()* түрлендірулерін *B*, *R*, *C* және *K* сәйкес белгілейміз. Кілт ұзындығы *128* бит болған жағдайда шифрлау алгоритмінің барлық әрекеттерін сызықты тізбек түрінде жазамыз:

$$\underbrace{K\,B\,R\,C\,K}_{\text{1-й раунд}}\,\underbrace{B\,R\,C\,K}_{\text{2-й раунд}}\ldots\underbrace{B\,R\,C\,K}_{\text{9-й раунд}}\,\underbrace{B\,R\,K}_{\text{10-й раунд}}. \qquad (8.34)$$

Егер (8.34) тізбегінде *SubBytes(), ShiftRows(), MixColumns()* және *AddRound-Key()* орындарына оларға инверсты түрлендірулерді орындаса және раундтарда түрлендірулерді қайта топтаса, онда қайта керішифрлау функциясын құруға болады (8.29 сурет).



8.29 сурет - *AES* (*Rijndael*) криптографиялық алгоритмінде қайта (инверсты) керішифрлау процедурасының сұлбасы

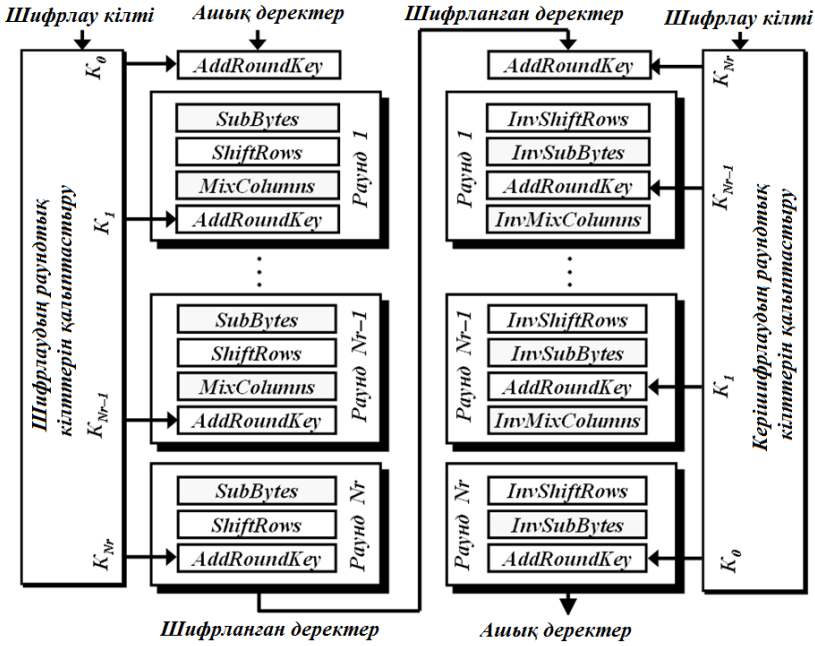
8.29 суретінен көрініп тұрғандай раундтық кілттерді қолдану реті шифрлау кезіндегіге қарағанда кері болады.

Төменде деректерді қайта (инверсты) керішифрлау (*InvDecipher*) процессін көрсететін бағдарламаның псевдокоды келтірілген.

```
//=============================================
//= Қайта керішифрлау процедурасы псевдокодта =
//=============================================
InvDecipher (byte in[4*Nb], word w[Nb*(Nr+1)], byte out[4*Nb])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey (state, w[Nr*Nb,(Nr+1)*Nb-1])
    InvShiftRows (state)
```

```
    InvSubBytes (state)
    for round = 1 step 1 to Nr-1
        AddRoundKey (state, w[(Nr-round)*Nb,
        (Nr+1-round)*Nb-1])
        InvMixColumns (state)
        InvShiftRows (state)
        InvSubBytes (state)
    end for
    AddRoundKey (state, w[0,Nb-1])
    out = state
end
//=============================================
```

8.29 суретінде *InvAddRoundKey()* алғашқы түрлендіруде және соңғы раундта *AddRoundKey()* ауыстырылған. Бұл *2* модулі бойынша қосу және алу операциялары бірдей болғандықтан мүмкін болып тұр.



8.30 сурет - *AES* (*Rijndael*) криптографиялық алгоритмінде шифрлау және қайта керішифрлау процесстерін түсіндіру

AES бірінші (алғашқы) жобасында шифрлау және керішифрлау кезінде әр раундтағы түрлендірулер реті бірдей болмады (8.30 сурет).

8.30 суретінен көрініп тұрғандай, біріншіден, керішифрлау кезінде *SubBytes* (*InvSubBytes*) және *ShiftRows* (*InvShiftRows*) түрлендірулерін орындау реті өзгереді; екіншіден, керішифрлау кезінде *MixColumns* (I*nvMixColumns*) және *AddRoundKey* түрлендірулерді орындау реті өзгертілген. Бұл ретте өзгертулер керішифрлау кезінде түрлендірулердің жұмыс ретін шифрлау кезіндегі жұмыс ретіне инверсты жасау үшін қажет. Сонымен, керішифрлау алгоритмі - шифрлау алгоритмінің инверсиясы. 8.30 суретінде тек үш раунд көрсетілген, басқаларында да осындай түрі бар.

Керішифрлау кезінде раунд кілттері өзгертілген (кері) ретпен қолданатынына және бірінші жобада шифрлау және керішифрлау алгоритмдері сәйкес келмейтініне назар аудару қажет.

8.8.2. Деректерді тұра (эквивалентті) керішифрлау

Шифрлау және керішифрлау үшін сәйкес келетін алгоритмдерді талап ететін қосымшалар үшін тұра (эквивалентті) керішифрлау алгоритмі құрылды. Альтернативті деп аталатын сондай нұсқада керішифрлау кезіндегі түрлендірулер шифрлау кезіндегідей болатындай құрылған.

Егер (8.34) B және R, сонымен қатар K және C нәтижені өзгертусіз орындармен ауыстырса

$$\underbrace{KRBKC}_{1\text{-й раунд}}\underbrace{RBKC}_{2\text{-й раунд}}\ldots\underbrace{RBKC}_{Nr-1\text{-й раунд}}\underbrace{RBK}_{Nr\text{-й раунд}},$$

онда бұл тізбекті оң жақтан сол жаққа қарай оқылған кезде келесі түрде жазуға болады

$$\underbrace{KBRCK}_{1\text{-й раунд}}\underbrace{BRCK}_{2\text{-й раунд}}\ldots\underbrace{BRCK}_{9\text{-й раунд}}\underbrace{BRK}_{10\text{-й раунд}}. \qquad (8.35)$$

Енді (8.35) тізбек (8.34) толық сәйкес келеді. Бұл (8.34) блокты шифрлау кезінде қолданған әрекеттер тізбегін қолдану және раундық кілттерді кері ретпен қолдану арқылы керішифрлауға болатынын білдіреді.

Шынымменде, *SubBytes()* түрлендіруін *ShiftRows()* түрлендіруімен орын ауыстыруға болады. *InvSubBytes()* және *InvShiftRows()* түрлендірулер үшін де бұл дұрыс болады. Бұның себебі *SubBytes()* және *InvSubBytes()* функциялары байттармен жұмыс істейді, ал *ShiftRows()* және *InvShiftRows()* операциялары байттарды жылжытады, бірақ олардың мәндеріне тиіспейді.

SubBytes() түрлендіруі күй матрицасының байттар ретін өзгертпей әр байттың мазмұнын өзгертеді; *ShiftRows()* байттардың мазмұнын өзгертпей күй матрицасында байттар ретін өзгертеді. Сондықтан, тұтас алгоритмнің қайтымдығына тиіспей керішифрлау кезінде бұл екі түрлендірудің ретін өзгертуге болады; 8.31 сурет бұл идеяны көрсетеді.

8.31 сурет - *SubBytes()* және *InvShiftRows()*, сонымен қатар *ShiftRows()* және *InvSubBytes()* түрлендірулердің орындау ретін өзгерту

Шифрлау және керішифрлау кезінде екі түрлендірудің комбинациясы бір біріне инверсты екенін белгілейміз.

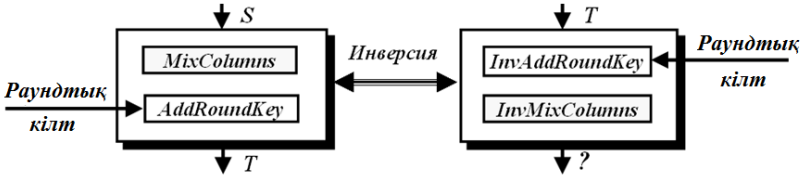
Жоғарыда сипатталған қайта (инверсты) керішифрлау алгоритмі шифрлау алгоритмінде қолданатын түрлендірулер ретіне кері ретті қолданады, бірақ сол параметрлерді қолданады (кеңейтілген кілтті). Бірақ *AES* (*Rijndael*) шифрлау алгоритмінің кейбір қасиеттері керішифрлау үшін кейбір параметрлерді өзгерту арқасында, нақты айтқанда - кеңейтілген кілтті, тұра сондай түрлендірулер ретін (шифрлауда қолданған) қолдануға мүмкіндік береді:

$$\begin{aligned} &InvMixColumns(AddRoundKey()) = \\ &= InvMixColumns(State \oplus RoundKey) = \\ &= InvMixColumns(State) \oplus InvMixColumns(RoundKey). \end{aligned} \qquad (8.36)$$

Керішифрлау алгоритмінің функцияларының бұл қасиеттері *AddRounKey()* және *InvMixColumns()* түрлендірулерін қолдану ретін өзгертуге мүмкіндік береді. (8.36) шығатындай орындау ретінің бұндай өзгеруі келесі шарттпен мүмкін: керішифрлаудың барлық

раундтық кілттері (алғашқы раундтық кілт - K_0 және соңғы – K_{Nr} кілттен басқа) алдын ала *InvMixColumns()* функциясынан өткізілген (түрлендірілген) болу қажет.

InvMixColumns() және *AddRoundKey()* түрлендірулерде тек өздеріне тән әртүрлі қасиеттері бар. Бірақ, егер *InvMixColumns()* түрлендіруінде қолданатын тұрақтылар матрицасының инверсиясын раундтық кілттер матрицасына көбейтсе, бұл түрлендірулер бір біріне инверсия болу мүмкін. Жаңа түрлендіруді *InvAdd-RoundKey()* деп атаймыз. 8.32 сурет жаңа конфигурацияны көрсетеді.



8.32 сурет - *MixColumns()* және *InvAddRoundKey()*, сонымен қатар *AddRoundKey()* және *InvMixColumns()* түрлендірулердің орындау ретін ауыстыру

Бұл екі түрлендіру *MixColumns()* және *AddRoundKey()* енді бір біріне инверсты екенін көрсетуге болады.

Шифрлау кезінде кіріс күй матрицасын S деп белгілейміз, ал шығысты – T, ал керішифрлау кезінде кіріс күй матрицасын T деп белгілейміз. Бұл жағдайда алғашқы күй матрица S екенін көрсетеміз.

MixColumns() түрлендіруі нақты түрде тұрақтылар матрицасын C күй матрицасына S көбейту нәтижесі екеніне назар аудару қажет. Сонымен, шифрлау нәтижесі - күй матрицасы T

$$T = C \bullet S \oplus K.$$

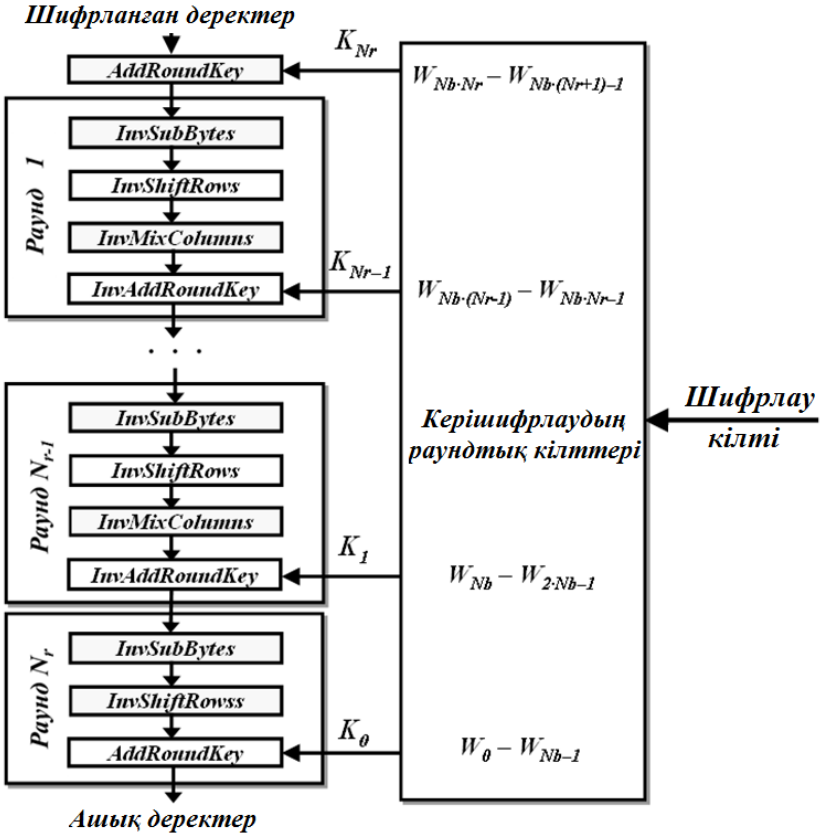
Керішифрлау кезінде

$$C^{-1} \cdot T \oplus C^{-1} \cdot K = C^{-1} \cdot (C \cdot S \oplus K) \oplus C^{-1} \cdot K =$$

$$= C^{-1} \cdot C \cdot S \oplus C^{-1} \cdot K \oplus C^{-1} \cdot K = S,$$

себебі $C^{-1} \cdot C \cdot S = I \cdot S = S$, а $C^{-1} \cdot K \oplus C^{-1} \cdot K = 0$. Бұл жерде I – бірлік матрица.

(8.36) есепке алумен 8.33 суретінде көрсетілген деректерді тұра (эквивалентті) керішифрлауды аламыз.



8.33 сурет - *AES* (*Rijndael*) криптографиялық алгоритмінде тұра (эквивалентті) керішифрлау процедурасының сұлбасы

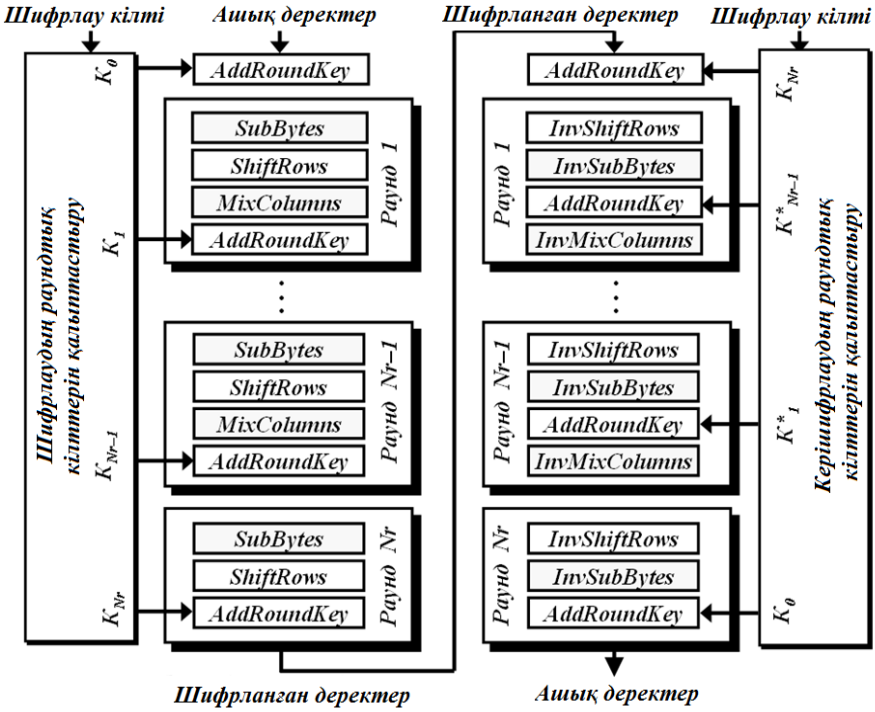
8.33 суретінде көрсетілгендей екі *AddRoundKey()* және тоғыз *InvAddRoundKey()* түрлендірулерін қолдану қажет екеніне назар аудару керек.

Егер 8.33 суретіндегі сұлбада K_1 ден K_{Nr-1} дейін раундтық кілттерді *InvMixColumns()* функциясы көмегімен айналдырса, яғни есептеулерді келесі өрнек көмегімен өткізсе:

$$K_i^* = InvMixColumns(K_i),\ i = 1 \ldots Nr-1, \qquad (8.37)$$

онда керішифрлау үшін шифрлау кезіндегі ретті қолдануға болады, бірақ оларға қайта.

Бұл жағдайда *AES* (*Rijndael*) криптографиялық алгоритмінің шифрлау және тұра керішифрлау кезіндегі үрдістер 8.34 суретінде көрсетуге болады.



8.34 сурет - *AES* (*Rijndael*) криптографиялық алгоритмінде шифрлау және тұра керішифрлау үрдістерін түсіндіру

Төменде деректерді шифрлау және тұра керішифрлау үрдістерін көрсететін бағдарламаның (*EqDeCipher*) псевдокоды келтірілген.

```
//================================================
//===  Тұра керішифрлау процедура псевдокодта  ===
//================================================
EqDeCipher (byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])
begin
    byte state[4,Nb]
```

```
    state = in
    AddRoundKey(state, dw[Nr*Nb, (Nr+1)*Nb-1])
    for round = Nr-1 step -1 downto 1
        InvSubBytes(state)
        InvShiftRows(state)
        InvMixColumns(state)
        AddRoundKey(state, dw[round*Nb, (round+1)*Nb-1])
    end for
    InvSubBytes(state)
    InvShiftRows(state)
    AddRoundKey(state, dw[0, Nb-1])
    out = state
end
//=============================================
```

(8.37) есепке алып жайылған кілтті қалыптастыру үшін кілтті жаю процедурасына келесі кодты қосу қажет

```
//=============================================
EqKeyExpansion (byte key[4*Nk], word w[Nb*(Nr + 1)], Nk)
begin
    for i = 0 step 1 to (Nr+1)*Nb-1
        dw[i] = w[i]
    end for
    for round = 1 step 1 to Nr-1
        InvMixColumns(dw[round*Nb, (round+1)*Nb-1])
    end for
//=============================================
```

Ескерту. Соңғы операторда (*InvMixColumn()* функциясында) деректер типтерін түрлендіру орындалады, себебі жайылған кілт сызықты *32*-разрядты сөздер массиві түрінде сақталады, ал функцияның кіріс параметрі - байттардың екі өлшемді массиві.

8.8 кестеде шифрлау процедуралары, сонымен қатар *AES* (*Rijndael*) екіраундтық нұсқасын қолдану кезінде керішифрлау процедурасының екі эвивалентті нұсқалары келтірілген.

Керішифрлау (қайта) функциясының бірінші нұсқасы - шифрлау функциясының қарапайым инверсиясы. Керішифрлау (тұра) функциясының екінші нұсқасы біріншіден үш жұпта түрлендіру ретін өзгертуден кейін алынған: *InvShiftRows – InvSubBytes* (екі рет) және *AddRoundKey – InvMixColumns*.

8.8 кесте

AES (*Rijndael*) екіраундтық нұсқасында түрлендірулер тізбегі

AES (*Rijndael*) екіраундтық нұсқасында түрлендірулер тізбегі		
шифрлау	қайта керішифрлау	тұра керішифрлау
AddRoundKey	*AddRoundKey*	*AddRoundKey*
SubBytes	*InvShiftRows*	*InvSubBytes*
ShiftRows	*InvSubBytes*	*InvShiftRows*
MixColumns	*AddRoundKey*	*InvMixColumns*
AddRoundKey	*InvMixColumns*	*AddRoundKey*
SubBytes	*InvShiftRows*	*InvSubBytes*
ShiftRows	*InvSubBytes*	*InvShiftRows*
AddRoundKey	*AddRoundKey*	*AddRoundKey*

Берілген жұптарда операцияларды орындаудың алғашқы ретінен қайта ретке өту кезінде түрлендіру нәтижесі өзгермейтіні айқын.

Шифрлау процедурасы және керішифрлаудың тұра процедура нұсқасы раундтық кілттерді (*AddRoundKey()* операцияны орындау барысында), ауыстыру кестелерін (*SubBytes()* және *InvSubBytes()* операцияларын орындау барысында) және түрлендіру матрицаларын (*MixColumns()* және *InvMixColumns()* операцияларын орындау барысында) қолдану ретіне дейін сәйкес келетіні көрініп тұр.

8.9. AES ҚАУІПСІЗДІГІ

8.9.1. Симметриялық қасиеттері және әлсіз кілттер

Деректермен жұмыс істегенде шифрға тән симметриялықтың (біркелкілігін) мәнді деңгейін есепке алып, оның әсерін кішірейтуге мүмкіндік беретін арнайы шаралар қолданылды. Оған әр раунд үшін әртүрлі тұрақтылар арқысында жетеді (кілтті жаю алгоритмін таңдауды негіздеуді қараңыз). Шифрлау және керішифрлау алгоритмдері әртүрлі операциялар-функциялардан тұратын факт, тәжірибелік түрде әлсіз және жартылай әлсіз кілттердің болу мүмкіндігін аластатады (бұл *DES* орын алды). Кілтті жаю

процедурасының сызықты еместігіде шифрлаудың әртүрлі алғашқы кілттерін жаюдан кейін эквивалентті раундтық кілттерді алу мүмкіндігін аластатады.

8.9.2. Дифференциалды және сызықты криптографиялық талдаулар

Алдында айтылғандай, дифференциалды криптографиялық талдау бірінші рет *Э. Бихаммен* және *А. Шамирмен* сипатталған болатын. Сызықты криптографиялық талдау бірінші рет *М. Мацуимен* сипатталған болатын.

Дифференциалды криптографиялық талдау

Дифференциалды криптографиялық талдау – бұл ашық деректерді ары қарай шифрлау үшін ерікті іріктеу мүмкіндігінде негізделген шифрға шабуыл. Деректер блоктарының жұптары арасындағы тәуелдіктер шифрды қолдану алдында және қолданғаннан кейін талданады. *ДК*-шабуылдар мүмкін болады егер осындай тәуелдіктердің шамамен барлық (*2–3* басқа) шифрлауыш түрлендірудің раундтарынан енуін табуға болтын кезде. Бұл жерде шифрлау кезінде кіріс айырмашылығынан шығыс айырмашылығының тәуелдігін анықтауға болатын деректер биттерінің тіркелген жұптарының салыстырмалы саны (бұл санды ену коэффициенті деп атаймыз) 2^{l-n} көп үлкен болу қажет, n – кіріс деректердің биттермен берілген ұзындығы. Дифференциалды криптографиялық талдаудың бұл критерийін түсіндіруге тырысамыз.

Біз әрқайсысы n биттен тұратын ашық деректердің екі әртүрлі блогын шифрлау үшін және ары қарай алынған екі шифрланған блокты талдау үшін таңдауға мүмкіндік алдық дейік. Еске саламыз, кіріс деректерде, олардың айырмашылығы шығыс деректерде анықталған биттер жұбында бит арасындағы айырмашылығы қандай болатынын білдіретін бит жұптарын іздейміз.

Сонымен, екі ашық деректер блоктарынан бір анықталған биттерден таңдаймыз және барлық мүмкін кіріс деректер мәндері үшін шифрлау нәтижелерін талдаймыз. Енді әртүрлі мәнді қабылдай алатын әр блокта $n–1$ бит қалғандықтан, кіріс деректердің әр блогында 2^{n-1} барлығы мәнді талдау қажет. Және егер кіріс деректерінің таңдалған жұп биттері арасындағы айырмашылық шығыс де-

ректерде биттер жұптарында айырмашылықты болжайтын жұп табылса, және бұл оқиға кездейсоқ емес болса (яғни оның ықтималдығы $1/2^{n-1} = 2^{1-n}$ үлкен), онда шифрлау түрлендіруде айқын әлсіздік бар деп және дифференциалды криптографиялық талдауды орындауға болады деп санауға болады.

Дифференциалды криптографиялық талдау кезінде шифрлау раундын өзінің кіріс және шығыс деректері бар деректерді түрлендірудің бөлек айырма сессиясы ретінде қарастыру ыңғайлы. Сонымен, толық айырма сессиясы бөлек сессиялардан тұрады - шифрлау раундтарынан, ал шифрлау кезінде толық ену коэффициенті құрамды сессиялардың ену коэффициенттерінің көбейтіндісі болады.

Осылайша, *ДК*-шабуылдарға беріктілікті қамтамасыз ету үшін бірнеше раундтан кейін ену коэффициенті 2^{1-n} үлкен болмау керек. Ара қарай түрлендірулердің төрт раундынан кейін ену коэффициенті 2^{-150} үлкен емес болатынының (ал *8* раундтан кейін – 2^{-300} аз) дәлелі келтіріледі. Бұл 8.1 кестенің деректерін есепке алып шифрдың әртүрлі *n* кезінде беріктілігін дәлелдейді.

Сызықты криптографиялық талдау

Сызықты криптографиялық талдауда ашық деректерді ары қарай шифрлау үшін іріктеу мүмкіндігі бар шабуылдарда қолданылады. Ол кіріс деректердің биттер арасындағы сызықты тауелдіктерді талдауда негізделген, олар шығыс деректер арасындағы анықталған тәуелдіктерді шарттайды.

$A[i_1, i_2, i_3, \ldots, i_n] - A$ деректер массивінің биттерінің *2* модулі бойынша қосындысы болсын, ал $i_1, i_2, i_3, \ldots, i_n$ мәндері олардың индекстері болып табылады, яғни

$$A[i_1, i_2, i_3, \ldots, i_n] = A[i_1] \oplus A[i_2] \oplus A[i_3] \oplus \ldots \oplus A[i_n].$$

Онда сызықты тәуелдік деп келесі түрі бар өрнекті айтамыз

$$P[i_1, i_2, i_3, \ldots, i_n] = C[i_1, i_2, i_3, \ldots, i_n] \oplus K[i_1, i_2, i_3, \ldots, i_n].$$

бұл жерде *P*, *C* және *K* кіріс деректер, шығыс деректер және кілт биттерінің массивтері.

Осылайша, егер *P* және *C* жеткілікті үлкен саны үшін берілген өрнекке сәйкес келетін *K* табуға болатын болса, онда кіріс және шығыс деректерінің сызықты тәуелдігі негізінде кілт туралы бір бит ақпараты ашылды деп айтуға болады. Сызықты тәуелдіктерді

талдау негізінде кілт туралы бір биттен көп ақпаратты анықтау әдістері бар.

Егер осындай сызықты тәуелдіктердің (басқаша корреляциялардың) бар болуын шамамен барлық (*2–3* басқа) шифрлаушы түрлендіру раундтарынан кейін анықтауға болатын болса, онда *СК*-шабуылдар тиімді болады. Сонымен бірге деректерде сондай корреляциялардың салыстырмалы саны (коэффициенты) $2^{-n/2}$ (яғни бір бит жұбынан көп) көп үлкен болу қажет. Корреляциялардың белгісі болады, яғни кіріс деректердегі тәуелдік шығыс деректердегі тәуелдікке сәйкес келетініне немесе оған қайтымды болатынына тәуелді теріс және оң болу мүмкін. Бұл раундтық кілттер биттерін табуға кілтті (сөз ойыны) береді. Шифрлау кезінде жалпы корреляция кіріс және шығыс деректерінде болжанатын биттер комбинациясы бар әр бөлек сызықты сессияда (яғни әр түрлендіру раундында) көрінетін барлық корреляциялар композициясы (немесе қосындысы) болып саналады.

СК-шабуылға беріктілігінің жеткілікті шарты жалпы корреляция коэффициенті $2^{-n/2}$ үлкен сызықты сессиялардың болмауы. Ары қарай төрт түрлендіру раундынан кейін корреляция коэффициенті 2^{-75} (ал 8 раундтан кейін 2^{-150} аз) аз болатынының дәлелдеуі келтіріледі.

Айырмалық және сызықты сессиялардың тиімділігі

[38] келесі көрсетілген:

- айырмалық сессияның толық ену коэффициенті берілген айырмалық сессиядағы барлық белсенді *S*-блоктардың ауыстыру кестелерінің ену коэффициенттерінің көбейтіндісіне жуықтатылу мүмкін;
- сызықты сессияның жалпы корреляциясы берілген сызықты сессиядағы барлық белсенді *S*-блоктардың ауыстыру кестелерінің кіріс және шығыс деректер арасындағы корреляция коэффициентерінің көбейтіндісіне жуықтатылу мүмкін.

Осылайша, кез келген шифрдың түрлендіру сессиясының стратегиясы келесіде болу қажет:

- кіріс және шығыс деректер арасындағы максималды ену коэффициенті және максималды корреляция коэффициенті мүмкінше аз болатындай *S*-блок таңдау қажет; *AES* (*Rijndael*) *S*-блоктары үшін олар 2^{-6} және 2^{-3} тең;

- белсенді *S*-блоктардың аз санымен көпраундты түрлендірулер болмайтындай шашыратуды құрастыру.

Ары қарай кез келген *4*-раундтық айырмалық немесе сызықты сессияда белсенді *S*-блоктардың саны *25* тең екені көрсетіледі. Бұл кез келген *4*-раундтық айырмалық сессия үшін ену коэффициентін 2^{-150} деп және кез келген *4*-раундтық сызықты сессия үшін корреляция коэффициентін көп дегенде 2^{-75} деп береді. Бұл шифр блоктарының барлық өлшемдеріне дұрыс және раундтық кілт мәнінен тәуелсіз.

Ескерту. Мүмкінді кері *8*-разрядты ауыстыру блоктар жинағынан кездейсоқ таңдалған *S*-блоктың сызықтық еместігі аз тиімді болады. Ауыстыру кестелерінің кіріс және шығыс деректері арасындағы максималды ену коэффициенті үшін типті мәндері 2^{-5} дан 2^{-4} дейін және корреляция коэффициенті үшін 2^{-2} [38].

Белсендік бейнелердің енуі

Дифференциалды криптографиялық талдау үшін бір раундтағы белсенді *S*-блоктар саны раундтың кіріс деректерінде айырмашылықтарды беретін нөл емес байттар санымен (екі кіріс деректер массивтеріне *xor* операциясын қолданудан кейін нөл емес байттар саны) анықталады. Белсенді *S*-блоктар позицияларын анықтайтын *State* массивінің бейнесі (*pattern*) *"белсендіктің айырмалық бейнесі"* терминімен белгіленсін, және *айырмалық байттық салмақ* бейнедегі белсенді (нөл емес) байттар санын белгілесін.

Сызықты криптографиялық талдау үшін белсенді *S*-блоктар саны кіріс деректер массивінде нөл емес байттар санымен анықталады. Белсенді *S*-блоктар позицияларын анықтайтын бейне *"корреляциялық белсендік бейнесі"* терминмен белгіленсін, және *корреляциялық байттық салмақ Z(a) a* бейнесіндегі белсенді (нөл емес) байттар саны болсын. Оған қоса, егер бағанда кемінде бір белсенді байт болса, онда *белсенді бейнедегі* баған *белсенді* деп аталсын. $Z_c(a) - a$ белсенді баған саны болсын, яғни бұл бейненің баған салмағы. $Z_j(a)$ деп белгіленетін *a* бейненің *j* бағанының байттық салмағы бұл бағандағы белсенді байттар санын береді.

Сессияның жалпы салмағы оның құрамына кіретін әр раунд үшін кіріс деректердің белсендік бейнелердің салмақтарының қосындысына тең. Айырмалық (сызықтық) белсендік бейнелерінің

түрлендіру раундтары арасынан енуін кіріс/шығыс деректердің тәуелдігінің айырмалық (сызықтық) сессияда енуі ретінде қарастыруға болады. Бұл 8.35 суретінде көрсетілген, қарамен белсенді байттар белгіленген.

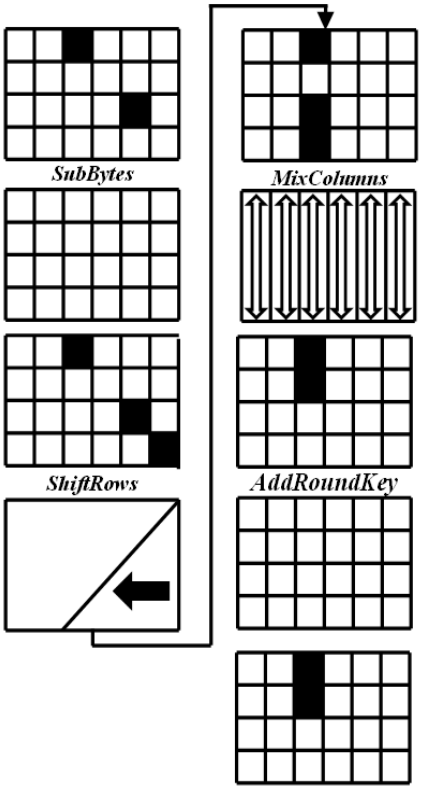
SubBytes() және *AddRound Key()*. Белсендік бейнелері, байттық және бағандық салмақтар өзгермейді.

ShiftRows(). Байттық салмақ өзгермейді, себебі байттардың мәндері өзгермейді.

MixColumns(). Бағандардың әрқайсысы бөлек түрлендірілетіндіктен бағандық салмақ өзгермейді.

Осылайша, *SubBytes()* және *AddRoundKey()* функциялары белсенді бейнелердің енуінде ешқандай рөл атқармайды, және сондықтан қарастыру кезінде түрлендіру раунд әсері *ShiftRows()* және *MixColumns()* функциялар әсеріне тұйылады.

Ары қарай *SubBytes()* және *AddRoundKey()* есепке алынбайды.

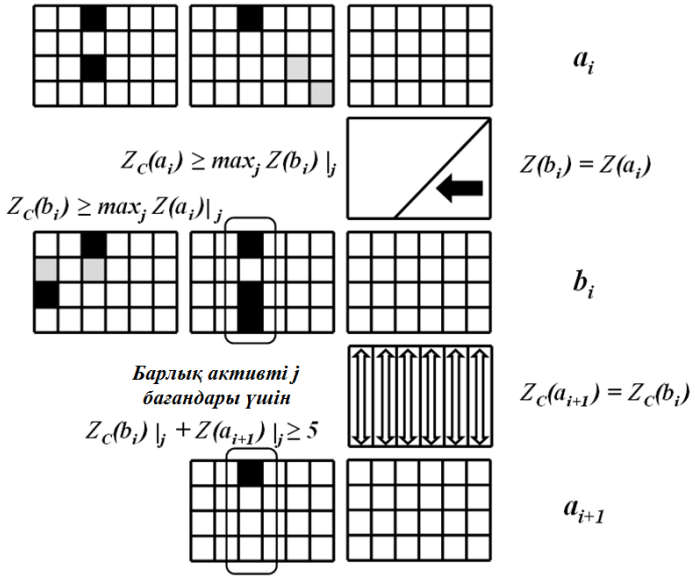


8.35 сурет - Бірраундтық түрлендіру арқылы белсендік бейнелердің енуі

Жоғарыда айтылғандай, *MixColumns()* функциясының тарату коэффициенті 5 тең. Бұл кіріс (немесе шығыс) деректер бейнесінен кез келген белсенді баған үшін раунд кірісінде және шығысында байттық салмақ қосындысы төмен жақтан 5 мәнімен шектелетін болатынын білдіреді.

Өз кезегімен функцияда келесі қасиеттер бар:

- шығыс деректердің бағандар салмағы төмен жақтан кіріс деректерден бағанның максималды байттық салмағымен шектелген;
- кіріс деректердің бағандар салмағы төмен жақтан шығыс деректерден бағанның максималды байттық салмағымен шектелген.



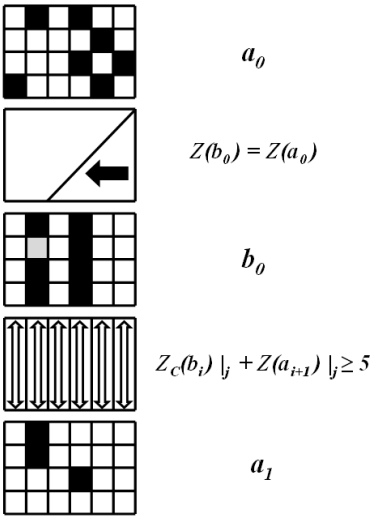
8.36 сурет - Бір раунд түрлендірулерінде бейнелер енулері

Сонымен, біздің сипаттамамызда ары қарай белсендік бейнесі *i*-ші раунд кірісінде a_{i-1} деп белгіленсін, ал *ShiftRows()* функциясын қолданғаннаң кейін b_{i-1} деп белгіленсін. Раундтардың нөмірленуі бірден басталады, сондықтан алғашқы белсендік бейне a_0 деп белгіленеді. Онда a_i және b_i тек *ShiftRows()* функциясымен бөлінген және байттық салмақтары бірдей болады, ал b_{i-1} және a_i тек *MixColumns()* функциясымен бөлінген және оларда бағандық салмақ бірдей болады. *m*-раундтық сессияның жалпы салмағы a_0 ден a_{m-1} дейін бейне салмақтарының қосындысына тең. Белсендік бейнелердің ену қасиеттері 8.36 суретінде көрсетілген, қара түспен белсенді байттар белгіленген, сұр түспен - белсенді бағандар.

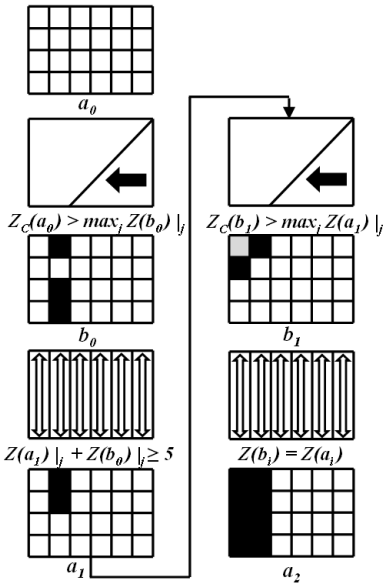
8.1 теорема. Q белсенді бағандары бар екі раундтық сессияның жалпы байттық салмағы екінші раунд кірісінде төмен жақтан *5Q* мәнімен шектелген.

Дәлелдеу. MixColumns() функциясының тарату коэффициенты *5* тең фактісі әр бағаң үшін b_0 және a_1 бейнелерінің бағандарында байттық салмақтар қосындысы төмен жақтан *5* мәнімен шектелгенін білдіреді. Сондықтан, егер a_1 бағандық салмағы *Q* тең болса, онда бұл b_0 және a_1 байттық салмақтарының қосындысы

үшін төмен жақтан $5Q$ шектеуін береді. a_0 және b_0 байттық салмақтары бірдей болғандықтан, бұл шектеу a_0 және a_1 байттық салмақтар қосындысына да жүреді. Бізге осыны дәлелдеу қажет болды. 8.1 теореманың суреттемесі 8.37 суретінде көрсетілген.



8.37 сурет - $Q = 2$ кезде 8.1 теореманың суреттемесі



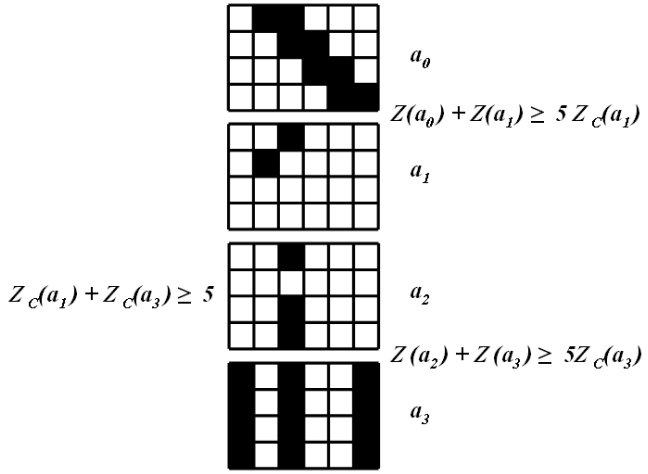
8.38 сурет - a_1 бейнесінде бір белсенді бағанмен 8.1 лемманың суреттемесі

Бұл жерден кез келген екі раундтық сессияда кемінде 5 белсенді S-блок бар екені шығады.

8.1 лемма. Екіраундтық сессияда кіріс деректердегі белсенді бағандар және шығыс деректердегі белсенді бағандар қосындысы 5 кем емес. Басқа сөзбен a_0 және a_2 бейнелердің бағандық салмақтар қосындысы 5 кем емес.

Дәлелдеу. *ShiftRows()* функциясы a_i бейнесінен кез келген баған байттарын b_i бейнесінен кез келген бағанға жылжытады және керісінші. Бұл жерден келесі шығады: a_i бейнесінің бағандық салмағы төмен жақтан b_i бөлек бағандарының байттық салмағымен шектелген. Тұра солай b_i бейнесінің бағандық салмағы төмен жақтан a_i бөлек бағандарының байттық салмағымен шектелген.

Сессияда a_1 бейнесінің кемінде бір бағаны белсенді (немесе, b_0 бейнесінің, тұра солай). Ол бағанды g бағаны деп белгілейміз. *MixColumns()* функциясының тарату коэффициенті *5* тең болғандықтан, онда b_0 бейнесінде және a_1 бейнесінде g бағанының байттық салмақтар қосындысы *5* кем болмайды. Бірақ a_0 бейнесінің бағандық салмағы төмен жақтан b_0 бейнесінің g бағанының байттық салмағымен шектелген. Ал b_1 бейнесінің бағандық салмағы төмен жақтан a_1 бейнесінің g бағанының байттық салмағымен шектелген. Сондықтан, a_0 және b_1 бейнелерінің бағандық салмақтарының қосындысы төмен жақтан *5* мәнімен шектелген. a_2 бейнесінің бағандық салмағы b_1 бейнесінің бағандық салмағына тең болғандықтан, лемма дәлелденген деп санауға болады. 8.1 лемманың суреттемесі 8.38 суретінде келтірілген.



8.39 сурет - 8.2 теореманың суреттемесі

8.2 теоремасы. 4 раундтан тұратын кез келген сессияда кемінде *25* белсенді байт болады.

Дәлелдеу. 4-раундтық сессияның жалпы байттық салмағы $Z(a_0)$, $Z(a_1)$, $Z(a_2)$ және $Z(a_3)$ байттық салмақтарының қосындысынына тең. 8.1 теоремасын бірінші екі раундқа (8.39 суретінде *1* және *2*) және соңғы екі раундқа (8.39 суретінде *3* және *4*) қолданып, *4*-раундтық сессияның жалпы байттық салмағы төмен жақтан a_1 және a_3 бейнелерінің бағандық салмақтарының қосындысын беске көбейтумен

шектелгенін аламыз. Ал 8.1 леммасына сәйкес a_1 және a_3 бейнелері үшін бағандық салмақтарының қосындысы *5* кем емес.

Бұл жерден *4*-раундтық сессияның байттық салмағы төмен жақтан *25* мәнімен шектелгені шығады. 8.2 теоремасы 8.39 суретінде суреттелген.

8.9.3. Қысқартылған дифференциалдар әдісімен шабуыл

Қысқартылған дифференциалдар тұжырымдамасы бірінші рет *Л. Кнудсен* жұмысында жарық көрді [7, 20]. Бұл кластың шабуылдары кейбір шифрлар үшін айырымдық сессияларда кластеризациялауға беталысы барда негізделген [38].

Бұл кіріс және шығыс деректерде сызықты тәуелдіктердің кейбір комбинациялары үшін осындай тәуелдіктердің көбін бақылауға мүмкіндік беретін раундтар саны өте үлкен болатынын білдіреді. Басқа сөзбен осындай деректер комбинациялары үшін дифференциалдық криптографиялық талдау тиімділігі қатты өседі. Барлық түрлендірулері тегістелген кіріс деректер массивінің блоктарына орындалатын шифрлар осындай шабуыл түрлеріне ұшырауға беталысы бар. *AES* (*RIJNDAEL*) түрлендіру функциялары бүтін байттармен жұмыс істейтіндіктен оның осындай қасиеті бар, сондықтан оның бұл шабуылдар түріне ұшырайтындығы арнайы зерттелген болатын. *6* үлкен раундтар саны үшін барлық кілт кеңістігіне толық талдау шабуылдан тиімдісі жоқ екені анықталды.

8.9.4. “Төртбұрыш” шабуылы

"Төртбұрыш" шабуылы арнайы *Square* аттас шифры үшін құрылғын (авторлар *Й. Демен* (*J. Daemen*), *Л. Кнудсен* (*L. Knudsen*), *В. Раймєн* (*V. Rijmen*)). Шабуыл орындау кезінде шифрдың байтқабағытталған құрылымын қолданады. Оның сипаттамасы шифрмен бірге жұмыста жарық көрді [38]. *AES* (*Rijndael*) *Square* шифрының көп қасиеттерін мұраланғанын есепке алып бұл шабуылды оған қолдануға болатынын айтуға болады. Ары қарай *AES* (*Rijndael*) *"Төртбұрыш"* шабуылын қолдану сипатталған.

"Төртбұрыш" шабуылы шабуылшының кейбір ашық деректерді ары қарай шифрлау үшін еркін іріктеу мүмкіндігінде негізделген. Ол *S*-блоктардың ауыстыру кестелерінен, *MixColumns()* функциясының көпмүшесінен және кілтті жаю тәсілінен тәуелсіз.

Бұл *6*-раундтық *AES* (*Rijndael*) шифры үшін шабуыл барлық кілттік кеңістікке толық талдаудан тиімді. *4*-раундтық *AES* (*Rijndael*) базалық шабуылды сипаттаудан кейін бұл шабуылды *5, 6* раундқа дейін қалай созуға болатынын көрсетіледі. Бірақ *7* раунд үшін *"Төртбұрыш"* толық талдаудан тиімсіз болады.

Алғышарттар

λ-жинақ *256* кіріс блогынан (*State* массивінен) тұратын жинақ болсын, олардың әрқайсысында мәндері барлық *256* блок үшін әртүрлі байттары (белсенді деп атаймыз) бар. Қалған байттар (пассивті деп атаймыз) *λ*-жинағынан барлық *256* блогы үшін бірдей болып қалады. Яғни барлық *x* және *y* үшін:

$$x,y \in \lambda : \begin{cases} x_{i,j} \neq y_{i,j}, & \text{егер } ij \text{ нөмірімен байт белсенді} \\ x_{i,j} = y_{i,j}, & \text{кері жағдайда} \end{cases}$$

SubBytes() және *AddRoundKey()* функциялармен өңделген *λ*-жинақтың блоктары нәтижесінде сол позицияларда белсенді байттары бар басқа *λ*-жинақты береді. *ShiftRows()* функциясы бұл байттарды *State* массивтің жолдарында берілген жылжытуларға сәйкес жылжытады. *MixColumns()* функциясынан кейін *λ*-жинақ жалпы жағдайда *λ*-жинақ болып қалуы міндетті емес (яғни түрлендіру нәтижесі *λ*-жинақ анықтамасына сәйкес болмауы мүмкін). Бірақ *MixColumns()* функция нәтижесінің әр байты сол бағанның төрт кіріс байттың сызықты комбинациясы (кері коэффициенттерімен) болғандықтан

$$b_{ij} = 2 \cdot a_{ij} \oplus 3 \cdot a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j},$$

кірісте бір белсенді байтпен баған нәтижеде шығыста барлық төрт белсенді байттармен баған береді.

4 раундқа "Төртбұрыш" базалық шабуыл

Барлық блоктарда тек бір байт белсенді *λ*-жинағын қарастырамыз. Басқаша айтқанда, бұл байттың мәні барлық *256* блокта әртүрлі, ал қалған байттар бірдей (мысалы, нөлге тең). Бұл байттың дамуын үш раундта бақылаймыз. Бірінші раундта

MixColumns() функциясы бір белсенді байтты *4* белсенді байты бар бағанға түрлендіреді. Екінші раундта бұл *4* байт *ShiftRows()* функциясының түрлендіру нәтижесінде *4* әртүрлі бағанға кетеді. Үшінші раундтың *MixColumns()* функциясы бұл байттарды белсенді байттарынан тұратын төрт бағанға түрлендіреді. Бұл жинақ үшінші раундтың *MixColumns()* функциясының кірісіне барғанша λ-жинақ болып қалады.

Бұл жерде қолданатын λ-жинақтың негізгі қасиеті келесі: осындай жинақтың барлық блоктарының *2* модулі бойынша разрядтық қосындысы әрқашан нөлге тең. Разрядтық *xor* операцияның анықтамасы бойынша белсенді емес (бірдей мәндері бар) байттардың разрядтық қосындысы нөлге тең, ал белсенді байттар барлық *256* мәндерді өтіп разрядты қосу кезінде нөл береді. Енді үшінші раундта *a* кіріс деректер миссивін *MixColumns()* функциясымен *b* шығыс деректер массивіне түрлендіру нәтижесін қарастырамыз. Бұл жағдайдада шығыс жинағының барлық блоктарының разрядты қосындысы нөлге тең болатынын көрсетеміз, яғни:

$$\bigoplus_{b=MixColumns(a),a\in\lambda} b_{ij} = \bigoplus_{a\in\lambda} (2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j} =$$

$$= 2 \bigoplus_{a\in\lambda} a_{ij} \oplus 3 \bigoplus_{a\in\lambda} a_{(i+1)j} \oplus \bigoplus_{a\in\lambda} a_{(i+2)j} \oplus \bigoplus_{a\in\lambda} a_{(i+3)j} .$$

Осылайша, төртінші раунд кірісінде барлық деректер теңдестірілген (яғни олардың толық соммасы нөлге тең). Бұл теңгерім жалпы жағдайда *SubBytes()* функциясымен деректерді түрлендірумен бұзылады.

Енді төртінші раунд соңғы деп есептейік, яғни оның ішінде *MixColumns()* функциясы жоқ. Онда бұл раундтың шығыс деректерінің әр байты кіріс деректерінің бір байтынан тәуелді. Егер төртінші раундтың шығыс деректерінің байтын *a* деп, ал кіріс деректерінің байтын *b* деп және раундтық кілттің сәйкес байтын *k* деп белгілесек, онда келесіні жазуға болады:

$$a_{ij} = SubBytes(b_{ij}) \oplus k_{ij} .$$

Бұл жерден k_{ij} мәнін болжап белгілі a_{ij} бойынша b_{ij} есептеуге болады, ал содан кейін k_{ij} мәні туралы болжамның дұрыстығын

тексеруге болады: егер берілген k_{ij} кезінде алынған b_{ij} байттарының мәндері барлық блоктар бойынша теңдестірілген болмаса (яғни разрядталған қосу кезінде нөлдік нәтиже бермесе), онда болжам дұрыс емес.

Раундтың кілт байтының масималды 2^8 нұсқасын іріктеп, оның шын мәнін табамыз.

Осы принцип негізінде раундтық кілттің басқада байттарын анықтауға болады. Кілттің әр байты үшін іздеу бөлек жүргізілу мүмкін (қатар деп оқы) болғандықтан, раундтық кілттің барлық мәнін іріктеу жылдамдығы өте үлкен болады. Кілтті жаюдың алгоритмі белгілі болса толық раундтық кілт мәні бойынша шифрлаудың алғашқы кілтін қалпына келтіру үлкен жұмыс болмайды.

"Төртбұрыш" базалық шабуыл басына алтыншы раундты қосу

Негізгі идея келесіде: бірінші раундтаң соң шығыста бір белсенді байты бар λ-жинағын беретін ашық деректер жинағын іріктеу қажет. Бұл бірінші раундтың алдында *AddRoundKey()* функциясымен қолданатын кілттің төрт байтының мәні туралы болжамды талап етеді.

Екінші раунд кірісінде тек бір белсенді байт болу үшін бірінші раундта *MixColumns()* функциясының шығысында бір белсенді байт қалғаны жеткілікті. Бұл деген бірінші раундтың *MixColumns()* кірісінде *256* блоктан тұратын жинақ үшін бағанның *a* байттары сызықты түрлендіру нәтижесінде келесідей болу қажеттігін білдіреді:

$$b_i = 2 \cdot a_i \oplus 3 \cdot a_{i+1} \oplus a_{i+2} \oplus a_{i+3}, \quad 0 \le i \le 3,$$

бұл жерде i – жол нөмірі. Бір анықталған i үшін әртүрлі *256* мән берілді, ал сол уақытта қалған үш i мәндерінің әрқайсысы үшін бұл түрлендіру нәтижесі тұрақты болып қалу қажет.

Қайтадан бірінші раундтың түрлендіру функциялары ретімен *ShiftRows()* қарай жылжуда берілген шартты бағандар бойынша сәйкес таратылған *4* байтқа қолдану қажет. *SubBytes()* функциясын қолдануды және *4*-байттық раундтық кілт мәнімен қосуды есепке алып теңдеулерді батыл құрастыруға және ары қарай нәтижені

талдау үшін шифрлауға жіберілетін ашық ақпараттың байт мәндерін іріктеуге болады:

$$b_{ij} = 2 \cdot SubBytes(a_{ij} \oplus k_{ij}) \oplus 3 \cdot SubBytes(a_{(i+1)(j+1)} \oplus$$

$$\oplus k_{(i+1)(j+1)}) \oplus SubBytes(a_{(i+2)(j+2)} \oplus k_{(i+2)(j+2)}) \oplus$$

$$\oplus SubBytes(a_{(i+3)(j+3)} \oplus k_{(i+3)(j+3)}), \quad 0 \le i, j \le 3.$$

Осылайша бұзудың келесі алгоритмін аламыз. Анықталған *i* және *j* үшін барлығы 2^{32} әртүрлі мәндері бар. Қалған байттар барлық блоктарға бірдей (пассивті байттар). Бірінші раунд кілтінің *k* төрт байт мәндерін болжап, *256* блоктардан жинақты іріктейміз (жоғарыда сипатталған шарты қарап). Бұл *256* блок бірінші раундтан кейін *λ*-жинақ болады. Бұл *λ*-жинаққа *4* раунд үшін базалық шабуылды қолдануға болады. Оның көмегімен іріктелген соңғы раунд кілтінің бір байты тіркеледі.

Енді бірінші раундтың кілтінің *k* сол *4* байт мәндері үшін *256* блоктан тұратын жаңа жинақ іріктеледі. Тағы соңғы раунд кілтінің бір байтын беретін базалық шабуыл жүргізіледі. Егер бірнеше талпыныстан кейін бұл байт мәні өзгермесе, онда бұзу дұрыс жолда. Басқа жағдайда бірінші раунд кілтінің *k 4* байт мәні тұралы болжамды ауыстыру қажет. Бұндай әрекеттер алгоритмі соңғы раунд кілтінің барлық байттарын толық қалпына келтіруге жеткілікті тез алып келеді.

Тиімділік және жадыға талаптар

Осылайша *"Төртбұрыш"* шабуылы *AES* (*Rijndael*) шифрының *6* раундына қолдануы мүмкін. Ол толық барлық кеңістікке толық іріктеуге қарағанда тиімдірек болады. *"Төртбұрыш"* шабуылы үшін жұмыскөлемдік және талаптар 8.9 кестеге жалпыланған [38]. *"Төртбұрыш"* шабуылының кез келген *7* және одан көп раундқа белгілі жалғасының жұмыскөлемділігі қарапайым кілттің толық іріктеуіне қарағанда жоғары болады.

8.9 кесте

"Төртбұрыш" шабуылын өткізу үшін қажетті ресурстар

Шабуыл типі	*Қажетті ашық деректердің блоктар саны*	*Шифрлау процедурасын қайталау саны*	*Жадыға талаптар*
Базалық *4*-раундтық шабуыл	29	29	Үлкен емес
Кеңейтілген, соңына раунд қосумен	211	240	Үлкен емес
Кеңейтілген, басына раунд қосумен	232	240	232
Кеңейтілген, соңына және басына раунд қосумен	232	272	232

8.9.5. Интерполяция әдісімен шабуыл

[7, 20, 38] жұмысында *Т. Джéфферсон* және *Л. Кнудсен* блокты шифрларға жаңа шабуыл түрін ұсынды. Оның мәні келесіде: шабуылшы оған белгілі кіріс/шығыс деректер жұптарынан көпмүшені алуға тырысады. Бұл тек түрлендіру құрамына кіретін функциялар алгебралық түрде жеткілікті тығыз көрсетілу, ал бүкіл түрлендіру – шешілетін қиындығы бар алгебралық өрнек ретінде көрсетілу мүмкін болғанда мүмкін болады.

Шабуыл келесі фактіде негізделеді: егер алынған көпмүшенің (немесе ұтымды өрнектің) дәрежесі үлкен болмаса, онда кілттің мәнімен тікелей байланысты көпмүшенің коэффициент мәндерін іріктеу үшін кіріс/шығыс деректердің бірнеше жұптары қажет. *GF(2^8)* өрісінде *SubBytes()* функциясының алгебралық көрсету қиындығы *MixColumns()* функциясы шашыратуымен бірге бұл шабуыл типін көпраундтық *AES* (*Rijndael*) шифрына қолдануды мүмкін емес жасайды. Мысалы, *SubBytes()* функциясы үшін алгебралық өрнек:

$$\{63\} + \{8f\}\cdot x^{127} + \{b5\}\cdot x^{191} + \{01\}\cdot x^{223} + \{f4\}\cdot x^{239} +$$
$$+ \{25\}\cdot x^{247} + \{f9\}\cdot x^{251} + \{09\}\cdot x^{253} + \{05\}\cdot x^{259}.$$

8.9.6. Әлсіз кілттер туралы

Қолдануы кіріс деректерді (жеркілікті емес шашырату немесе араластыру) жаман түрлендіруге алып келетін кілттер әлсіз деп саналады.

Ереже ретінде, бұндай кемшілік кілттер мәнінен тәуелді болатын сызықты емес операциялары бар шифрларға тән. Бұл олай емес *AES* (*Rijndael*)-де кілтті шифрлау процедурасына қосу *xor* операциясын қолданумен жүзеге асырылады, ал сызықты емес түрлендірулерді тіркелген ауыстыру кестелері бар *S*-блоктар жүзеге асырады. Осылайша, *AES* (*Rijndael*) кілттерді таңдауда шектеулер жоқ.

8.9.7. "Эквивалентті кілттер" шабуылы

Э. Бихэм [38] "*эквиваленттік кілттер*" *шабуылын* сипаттады. Кешірек *Д. Келси, Б. Шнайер* және *Д. Вагнер* [20, 47] жұмысында кейбір шифрлар бұл шабуылға жеткілікті берік болмайтынын көрсетті.

"*Эквиваленттік кілттер*" *шабуылында* криптографиялық талдаушы берілген өзара тәуелдіктері бар әртүрлі (толық немесе бөлшекті белгісіз) кілттерді қолданып шифрлауды орындайды. Кілтті жаю кезінде шашыратудың жоғары деңгейін есепке алып *AES* (*Rijndael*) бұл шабуылдың мүмкіндігінің азықтималды екенін айтуға болады.

Бақылау сұрақтары және тапсырмалар

1. *AES* үш нұсқасы үшін параметрлерді (блок көлемі, кілт көлемі және раундтар саны) айтыңыз.
2. *AES* әр нұсқасында қанша түрлендіру бар? Әр нұсқа үшін қанша раундтық кілт қажет?
3. *DES* және *AES* салыстырыңыз. Олардың қайсысы битпен жұмысқа бағытталған, ал қайсысы байтпен жұмысқа бағытталған?
4. *AES* күй матрицасын анықтаңыз. *AES* әр нұсқасында қанша күй матрицасы бар?
5. *AES* үшін анықталған төрт түрлендірудің қандайлары байт құрамын өзгертеді, ал қандайлары өзгерпейді?

6. *DES* және *AES* ауыстыруды салыстырыңыз. Неге *AES* бір ауыстыру кестесі бар, ал *DES* бірнеше (*S*-блоктар)?

7. *DES* және *AES* алмастыруларды салыстырыңыз. Неге *DES* алмастыруды кеңейту және қысу қажет, ал *AES* қажет емес?

8. *DES* және *AES* раунд кілттерін салыстырыңыз. Қай шифрда раунд кілтінің көлемі блок көлеміне тең?

9. Негі деректерді араластыратын *MixColumns* түрлендіруі *AES* қажет, бірақ *DES* қажет емес?

10. *AES* алгоритмінің жұмыс істеу режимдерін айтыңыз.

11. Шегі бар өрістің екі элементін $x^7 + x^3 + x + 1$ және $x^6 + x^3 + x^2 + 1$ қосыңыз.

12. Шегі бар өрісте $x^{13}+x^{11}+ +x^9+x^7+x^5+x+1$ элементін $x^6+x^3+x^2+1$ элементіне бөлудің нәтижесін және қалдығын анықтаңыз.

13. Шегі бар өрістің $GF(2^8)$ коэффициенттерімен екі көпмүшені $\{f5\}\cdot x^7+\{03\}\cdot x^3+\{09\}\cdot x+\{02\}$ және $\{07\}\cdot x^7+ +\{1b\}\cdot x^5+\{1e\}\cdot x+\{1f\}$ қосуды орындаңыз. Келтірілмейтін көпмүше $p(x) = x^8+x^4+x^3+x+ +1$.

14. Шегі бар өрістің $GF(2^8)$ коэффициенттерімен екі көпмүшені $\{02\}\cdot x^3+\{05\}\cdot x^2+\{03\}\cdot x+\{04\}$ және $\{03\}\cdot x^3+\{05\}\cdot x^2+ +\{04\}\cdot x+\{01\}$ көбейтіңіз. Келтірілмейтін көпмүше $p(x) = x^8+ +x^4+x^3+x+1$.

15. Шегі бар өрістің $GF(2^8)$ коэффициенттерімен екі көпмүшені $\{02\}\cdot x^3+\{05\}\cdot x^2+\{03\}\cdot x+\{04\}$ және $\{03\}\cdot x^3+\{05\}\cdot x^2+\{04\}\cdot x+\{01\}$ көбейтіңіз. Көбейту нәтижесі *4*-байттық сөзбен көрсетілсін, бұл жерде $m(x) = x^4+1$ көпмүшені қолданыңыз.

16. Шегі бар өрістен $GF(2^8)$ бір байт ұзындығы бар көпмүшенің $x^6+x^3+x^2+1$ аддитивті инверсиясын анықтаңыз.

17. Шегі бар өрістен $GF(2^8)$ бір байт ұзындығы бар көпмүшенің $x^6+x^3+x^2+1$ мультипликативті инверсиясын анықтаңыз. Келтірілмейтін көпмүше $p(x) = x^8+ +x^4+x^3+x+1$.

18. *AES* алгоритмінде шегі бар өрістен $GF(2^8)$ бір байт ұзындығы бар көпмүшенің x^5+x^2+1 аффинды түрлендіру нәтижесін анықтаңыз. Келтірілмейтін көпмүше $p(x) = x^8+ +x^4+x^3+x+1$.

19. *AES-128* криптографиялық жүйесінің *SubBytes()* функциясының кірісінде деректердің күй матрицасының байттар мәндері он алтылық санақ жүйесінде келесіге тең: *a49c7ff2689f352b6b5bea43026a5049*. *SubBytes()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

20. *AES-128* криптографиялық жүйесінің *InvSubBytes()* функциясының кірісінде деректердің күй матрицасының байттар

мәндері он алтылық санақ жүйесінде келесіге тең: *f196db453b53027789d2de491a87397f*. *InvSubBytes()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

21. *AES-128* криптографиялық жүйесінің *ShiftRows()* функциясының кірісінде деректердің күй матрицасының байттар мәндері он алтылық санақ жүйесінде келесіге тең: *49ded28945db96f17f39871a7702533b*. *ShiftRows()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

22. *AES-128* криптографиялық жүйесінің *InvShiftRows()* функциясының кірісінде деректердің күй матрицасының байттар мәндері он алтылық санақ жүйесінде келесіге тең: *f196db453b53027789d2de491a87397f*. *InvShiftRows()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

23. *AES-128* криптографиялық жүйесінің *MixColumns())* функциясының кірісінде деректердің күй матрицасының байттар мәндері он алтылық санақ жүйесінде келесіге тең: *49db873b453953897f02d2f177de961a*. *MixColumns()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

24. *AES-128* криптографиялық жүйесінің *InvMixColumns()* функциясының кірісінде деректердің күй матрицасының байттар мәндері он алтылық санақ жүйесінде келесіге тең: *31adb22485c967caedd5e4a2ff9e4ac3*. *InvMixColumns()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

25. *AES-128* криптографиялық жүйесінің *AddRoundKey()* функциясының кірісінде деректердің күй матрицасының байттар мәндері он алтылық санақ жүйесінде келесіге тең: *584dcaf11b4b5aacdbe7caa81b6db0e5*. *AddRoundKey()* функциясының кірісінде раундтық кілт күй матрицасының байттар мәні – *f2c295f27a9bb9435935807a7359f67f*. *AddRoundKey()* функциясының шығысында деректер күй матрицасының байттар мәнін анықтаңыз.

26. *AES-128* криптографиялық жүйесінің шифрлаудың тоғызыншы раунды үшін раундтық кілт матрицасының байттар мәні он алтылық санақ жүйесінде: *ac7766f319fadc2128 d12941575c006e*. Онынші раунд үшін раундтық кілт матрицасының байттар мәнін анықтау.

27. *AES–192* шифрлаудың жайылған кілттің бірінші алты сөзінің мәндері он алтылық санақ жүйесінде: *w[0] = = 00010203*; *w[1] = 04050607*; *w[2] = 08090a0b*; *w[3] = 0c0d0e0f*; *w[4] = 10111213*; *w[5]*

= 14151617. *AES–192* шифрлаудың жайылған кілттің келесі алты сөз мәндерін анықтаңыз.

28. *AES–256* шифрлаудың жайылған кілттің бірінші сегіз сөзінің мәндері он алтылық санақ жүйесінде: *w[0] = = 00010203*; *w[1] = 04050607*; *w[2] = 08090a0b*; *w[3] = 0c0d0e0f*; *w[4] = 10111213*; *w[5] = 14151617*; *w[6] = 18191a1b*; *w[7] = = 1c1d1e1f*. *AES–256* шифрлаудың жайылған кілттің келесі сегіз сөз мәндерін анықтаңыз.

9 тарау

ДЕРЕКТЕРДІ ШИФРЛАУДЫҢ ЕУРОПАЛЫҚ СТАНДАРТТАРЫ

9.1. ЕУРОПАЛЫҚ СТАНДАРТТАРДЫ ҚҰРУ ТАРИХЫ

Жақында өткен заманға дейін коммерциялық және қаржылық мекемелерде негізгі қолданыста болған блокты симметриялық криптографиялық алгоритм *DES* болды. Бірақ есептеу қуаттығының жоғарлауы және бір қатар криптографиялық талдаудың озық математикалық әдістерінің пайда болуы *DES* қауіпсіздіктің талап етілетін деңгейін қамтамасыз етпейтінін көрсетті. Бірінші қатарда бұл кілттің және блоктың қысқа ұзындығымен байланысты, ол жеткілікті қысқа уақытта арнайы криптографиялық процессорларда қарапайым талдауды жүргізуге мүмкіндік береді. Ескірген *DES* үшін лайықты ауыстыруды табу қажет болды. Бұл проблеманы бірінші американдықтар шешті, олар 1998 жылы жаңа американдық шифрлау стандартына *AES* байқауын ұйымдастырды. Бірнеше кезең барысында XX ғасырдың стандарты *Rijndael* таңдалды, ол блокты симметриялық криптографиялық алгоритмдер үшін жариясыз жаңа стандартты берді. Бұл *128* биттен кем емес блок ұзындығын және *128* биттен кем емес кілттер ұзындығын қолдану.

2000-2002 жылдары Еуропада ұқсас байқау өткізілді *NESSIE* (*New European Schemes for Signatures, Integrity and Encryption – Қол қою үшін, тұтастыққа және шифрлауға жаңа еуропалық сұлбалар*), оның мақсаты криптографиялық еуропалық стандартты таңдау [29]. Криптографиялық аналитикалық шабуылдар әдістерінің жақсаруымен байланысты үміткерлерге жоғарлатылған талаптар қойылды. Үш қауіпсіздік класы қарастырылды. Біріншіге, ең жоғарғыға, блок ұзындығы *128* және кілт ұзындығы *256* биттен кем емес криптографиялық алгоритмдер жатады; ортанғысына -

блок ұзындығы *128* және кілт ұзындығы *128* биттен кем емес криптографиялық алгоритмдер жатады; үшіншісіне, жаратымдыға - блок ұзындығы *64* және кілт ұзындығы *128* бит криптографиялық алгоритмдер жатады.

Іріктеу кезінде *AES* қатысқан алгоритмдерге қойылған талаптарға ұқсас талаптар да бағаланды: жаңа ұсыныстарды есепке алумен криптографиялық аналитикалық шабуылдардан алгоритмдердің қорғалғандығы; алгоритмдердің статистикалық қауіпсіздігі; математикалық базасының сенімділігі; шифрлаудың/керішифрлаудың есептеу қиындығы (жылдамдығы); бағдарламалық, аппараттық және бағдарламалық-аппараттық жүзеге асырудың қиындығы.

Байкау нәтижесінде криптографиялық алгоритмдер іріктелді [29]:

- жаратымды қауіпсіздік класы: *IDEA* (*Швейцария*); *Khazad* (*Бразилия* және *Бельгия*); *MISTY1* (*Жапония*); *SAFER++64* (*Швейцария*);

- орташа қауіпсіздік класы: *Camellia* (*Жапония*); *SAFER++128* (*Жапония*) және *RC6* (*Швеция* және *АҚШ*);

- жоғарғы қауіпсіздік класы: *Shacal* (*Франция*).

Сарапшылар олармен бірге финалистердің бірі ретінде *АҚШ AES* және *Triple DES* шифрлау стандарттарын қарастырды, олар *NESSIE* байқауына қатыспасада.

AES алгоритмі *АҚШ* жаңа стандарты болды және де-фато – бүкілге жуық әлем үшін блокты симметриялық стандарты.

Еуропалық стандарттармен нашарырақ: *АҚШ-та AES* іріктеумен уәкілетті мемлекеттік институт *NIST* айланысты, ал Еуропада – ғылыми ұйымдар (мысалы, Католикалық университет, *Лювен қаласы, Бельгия*) және ірі корпорациялар (мысалы, *Siemens*), және де сарапшылардың тұжырымдары ұсыну сипаттамасымен болды. Сонымен қатар, *NESSIE* байқауы өте кең болды – оны үш жыл өткізу барысында сарапшылар үлкен сан алгоритмдерін жете зерттеуге уақыттары жетпеді деген ой тұдырады.

NESSIE байқауының 2003 жылдың ақпанында біткеніне қарамастан жалпыеуропалық криптографиялық стандарттар әлі күнге дейін жоқ.

Іріктелген криптографиялық алгоритмдер ішінде құру көз қарасынан ең қызықты: *IDEA* және *Camellia*. Бұл алгоритмдерді толығырақ қарастырайық.

9.2. IDEA ДЕРЕКТЕРДІ ШИФРЛАУДЫҢ БЛОКТЫ АЛГОРИТМІ

9.2.1. IDEA алгоритмін құру тарихы

IDEA алгоритмі швейцариялық *Ascom* фирмасымен патенттелген симметриялық блокты шифр.

IDEA алгоритмінің бірінші нұсқасы 1990 жылы *DES* орнына ұсынылған, оның авторлары – Швейцариялық *ETH Zürich* (*Hasler Foundation* контракт бойынша, содан кейін *Ascom – Tech AG* қосылды) институтынан *Сюэцзя Лай* (*Xuejia Lai*) және *Джеймс Мэсси* (*James Massey*), шифрды *PES* (ағыл. *Proposed Encryption Standard – ұсынылған шифрлау стандарты*) деп атады. Содан кейін, *PES* дифференциалды криптографиялық талдау бойынша *Бихам* және *Шамирдің* жұмыстарын жариялағаннан кейін, алгоритм криптографиялық беріктілікті нығайту үшін жақсартылған және *IPES* (ағыл. *Improved Proposed Encryption Standard – жақсартылған ұсынылған шифрлау стандарты*) деп аталған. Бір жылдан кейін оның атын *IDEA* деп ауыстырды.

Алгоритмнің қайта қарастырылған, дифференциалды криптографиялық шабуылдарға қарсы қорғау құралдарымен нығайтылған нұсқасы 1991 жылы көрсетілген және 1992 жылы толық сипатталған.

Алғашында *DES* ауыстыратын алгоритмдердің ішіндегі симметриялық криптографиялық алгоритмдерінің бірі *IDEA.*

9.2.2. IDEA алгоритмін құру принциптері

Көптеген басқа блокты шифрлар сияқты *IDEA* алгоритмі шифрлау кезінде араластыру және шашырату процесстерін қолданады, барлық процесстер аппараттық және бағдарламалық құралдарымен жеңіл жүзеге асырылады.

IDEA 64-биттік деректер блоктарымен жұмыс істейді. *IDEA* алгоритмінің күмәнсіз артықшылығы - оның кілтінің ұзындығы *128* бит. Шифрлау және керішифрлау үшін бір алгоритм қолданылады.

IDEA құрастыру мақсаты жеткілікті қарапайым жүзеге асырумен және беріктілігімен криптографиялық алгоритмді құру.

IDEA келесі сипаттамалары оның криптографиялық беріктілігін сипаттайды:

Блок ұзындығы: алғашқы мәтіннің барлық статистикалық сипаттамаларын жасыратындай болатын блок ұзындығы болу қажет. Басқа жақтан, криптографиялық функцияны жүзеге асыру қиындығы блок көлеміне сәйкес экспоненциалды өседі. Көлемі *64* бит блокты қолдану 90-шы жылдары жеткілікті күшті білдірген. Осығын қоса, *CBC* шифрлау режимін қолдану алгоритмнің бұл аспектісін ары қарай нығайту туралы айтады.

Кілт ұзындығы: кілттің ұзындығы жеткілікті ұзын болу қажет, кілтті қарапайым талдау мүмкіндігін болдырмайтындай. Кілт ұзындығы *128* битпен *IDEA* жеткілікті қауіпсіз деп есептеледі.

Конфузия: шифрланған деректер кілттен қиын және шатасқан тәсілмен тәуелді болу қажет.

Диффузия: алғашқы деректердің әр биты шифрланған деректердің әр битіне әсер ету қажет. Бір шифрланбаған биттің көптеген шифрланған биттерге таралуы алғашқы деректердің статистикалық құрылымын жасырады. Шифрланған деректердің статистикалық сипаттамалары алғашқы деректердің статистикалық сипаттамаларынан қалай тәуелді болатынын анықтау қарапайым болмау керек. Бұл көз қарастан *IDEA* өте тиімді алгоритм.

IDEA сонғы екі тармақ үш операция көмегімен орындалады. Бұл оның *DES* айырмашылығы. *DES xor* операция негізінде және кішкентай сызық емес ауыстырудың *S*-блоктарында құрылған.

Әр операция екі *16*-биттік кіріске орындалады және бір *16*-биттік шығысты құрады. Бұл операциялар:

1. Биттік *xor*, $\oplus$ деп белгіленеді.

2. 2^{16} (*65536*) модулі бойынша бүтін сандар қосындысы, бұл жерде кірістер және шығыстар белгілері жоқ *16*-биттік бүтіндер деп есептеледі. Бұл операцияны $\boxplus$ деп белгілейді.

3. $2^{16} + 1$ (*65537*) модулі бойынша бүтін сандарды көбейту, бұл жерде кірістер және шығыстар белгілері жоқ *16*-биттік бүтіндер деп есептеледі. Тек нөлдерден тұратын блокты 2^{17} деп есептейді. Бұл операцияны $\otimes$ деп белгілейді.

Бұл үш операция келесі мәндерде үйлеспейді:

1. Дистрибутивті заңға жаратымды үш операциядан жұп болмайды. Мысалы

$$a \otimes (b \boxplus c) \neq a \otimes b \boxplus a \otimes c.$$

2. Ассоциативті заңға жаратымды үш операциядан жұп болмайды. Мысалы

$$a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c.$$

Осы үш операциядан қиыстыруды қолдану кірістің кешенді өзгеруін қамтамасыз етеді, криптографиялық талдауды қиындатады тек *xor* операциясында негізделген *DES* алгоритміне қарағанда.

9.2.3. IDEA алгоритмінің құрылымы

IDEA шифрлаудың жалпы сұлбасын қарастырамыз (9.1 сурет). Кез келген шифрлау алгоритмі сияқты, бұл жерде де екі кіріс бар: шифрланбаған блок және кілт. Бұл жағдайда шифрланбаған блоктың ұзындығы *64* бит, ал кілт ұзындығы *128* бит.

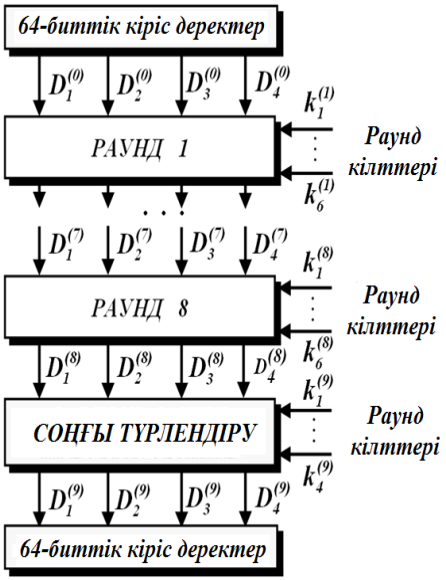
IDEA алгоритмі сегіз раундтан тұрады, содан кейін соңғы түлендіру болады. Алгоритм блокты төрт *16*-биттік ішкі блоктарға бөледі. Әр раунд кіріске төрт *16*-биттік ішкі блокты алады және шығыста төрт *16*-биттік ішкі блокты құрады.

Соңғы түрлендіруде кіріске төрт *16*-биттік ішкі блокты алады және шығыста төрт *16*-биттік ішкі блокты құрады.

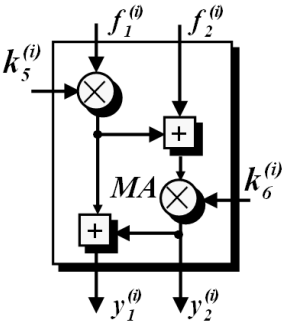
Әр раунд алты *16*-биттік раундтық кілтті қолданады, соңғы түрлендіру төрт ішкі кілтті қолданады, яғни алгоритмде жалпы *52* ішкі кілт қолданылады.

Диффузияны қамтамасыз ететін алгоритмнің негізгі элементтердің бірі *MA* (көбейту/қосу) деп аталатын құрылым.

MA (көбейту/қосу) құрылғының құрылымы 9.2 суретінде көрсетілген.



9.1 сурет – *IDEA* алгоритмінің құрылымы



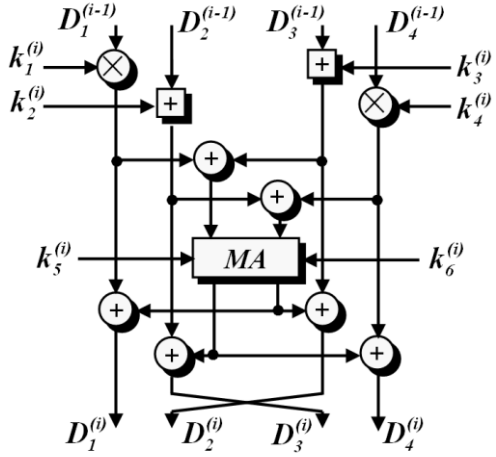
9.2 сурет - *MA* құрылғының құрылымы

Бұл құрылымның кірісіне екі *16*-биттік мәндер ($f_1^{(i)}$ және $f_2^{(i)}$) және екі *16*-биттік раундтық кілт ($k_1^{(i)}$ және $k_2^{(i)}$) беріледі, шығыста екі *16*-биттік мән ($y_1^{(i)}$ және $y_2^{(i)}$) құрылады.

Толық компьютерлік тексеру бұл құрылымның шығысының әр биты шифрланбаған деректер блогының әр кіріс битінен және раундтық кілттің әр битінен тәуелді екенін көрсетеді. Бұл құрылым алгоритмде сегіз рет қайталанады және жоғарытиімді диффузияны қамтамасыз етеді.

Бөлек раунд түрлендіру ретін қарастырамыз.

Раунд *16* бит ұзындығы бар деректердің төрт кіріс ішкі блогын: $D_1^{(i)}$, $D_2^{(i)}$, $D_3^{(i)}$ және $D_4^{(i)}$ *16* бит ұзындығы бар төрт раундтық кілттермен: $k_1^{(i)}$, $k_2^{(i)}$, $k_3^{(i)}$ және $k_4^{(i)}$ қосу және көбейту операцияларын қолданып құрамдастыратын түрлендіруден басталады (9.3 суретке қараңыз).

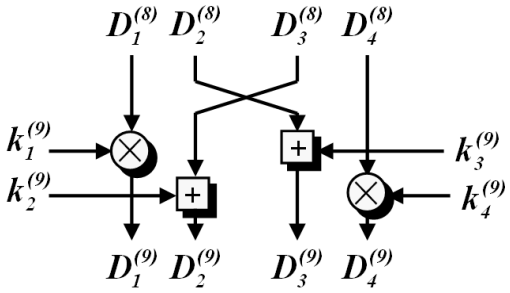


9.3 сурет - *IDEA* деректердің бір раунд түрлендіруі

Бұл түрлендірудің төрт шығыс блогы *MA* құрылымының кірісі болатын екі *16*-биттік ішкі блоктарды қалыптастыру үшін *xor* операциясын қолданып құрамдастырады. Сонымен қоса, *MA* құрылымының кірісінде екі раундтық кілт бар және шығыста ол екі *16*-биттік шығыс ішкі блоктарын құрады.

Соңында бірінші түрлендірудің төрт шығыс ішкі блогы *MA* құрылымының екі шығыс ішкі блогымен берілген итерацияның төрт шығыс ішкі блогын құру үшін *xor* қолдану арқылы құрамдастырылады. Бөлшекті екінші және үшінші кіріспен құрылатын екі шығыс екінші және үшінші шығысты ($D_2^{(i)}$ және $D_3^{(i)}$) құру үшін орындарын ауыстырады. Бұл биттерді араластыруды жоғарлатады және дифференциалды криптографиялық талдау үшін алгоритмнің беріктілігін жоғарлатады.

Соңғы түрлендіру деп белгіленген алгоритмнің тоғызыншы раундын қарастырамыз (9.4 суретті қараңыз).



9.4 сурет - *IDEA* соңғы түрлендіру (9 раунд)

Бұл құрылым жоғарыда сипатталған. Бір айырмашылық: екінші және үшінші кіріс орындарымен ауысады. Бұл керішифрлаудың құрылымы шифрлаумен бірдей болу үшін жасалған. Тоғызыншы раунд (соңғы түрлендіру) тек төрт кіріс раундтық кілтті талап етеді, ал бірінші сегіз раундтың әрқайсысы үшін алты кіріс раунд кілттері қажет екенін байқайық.

9.2.4. IDEA деректерді шифрлау үшін раундтық кілттерді генерациялау

IDEA деректерді шифрлау үшін раундтық кілттерді генерациялау

Кілтті кеңейту алгоритмі алғашқы шифрлау кілтінен *K* раундтық кілттерді алу ретін анықтайды. Раундтық кілттер шифрлау кілтінен кілттерді өндіру алгоритмі арқылы алынады. Ол екі компоненттен тұрады:

- шифрлау кілтін *K* кеңейту;
- раундтық кілттерді таңдау.

Алгоритмнің негізқұратын принциптері келесі:

- шифрлау кілті *K* кеңейтілген кілтке K_P кеңейтіледі;
- әр раунд кілтінің бит саны деректердің ішкі блок ұзындығына тең;
- раундтық кілттер саны келесі есеппен анықталады: деректерді шифрлау және керішифрлаудың әр раунды үшін алты кілт (*16* бит ұзындығы бар *48* кілт) және деректердің соңғы түрлендіру үшін төрт кілт (әрқайсысының ұзындығы *16* бит).

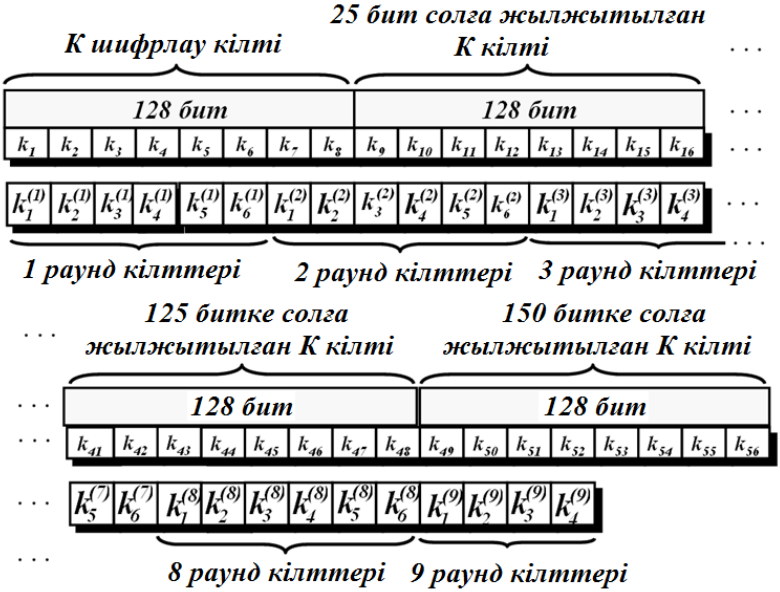
Барлығы елу екі *16*-биттік раундтық кілт генерациялану қажет.

Шифрлау кілтін *K* кеңейту келесі ретпен орындалады. Шифрлау кілті *K* әрқайсысы *16* битке тең сегіз бөлікке бөлінеді. Нәтижесінде деректерді шифрлау үшін бірінші сегіз кілт алынады, оларды k_1, k_2, ..., k_8 деп белгілейміз, k_1 кілті шифрлау кілтінің *K* бірінші *16* битіне тең, k_2 шифрлау кілтінің *K* келесі *16* битіне тең және ары қарай. Содан кейін шифрлау кілті *K* сол жақа *25* битке циклды жылжыйды. Алынған *128*-разрядты тізбек *16* биттен сегіз бөлікке бөлінеді. Нәтижесінде деректерді шифрлау үшін келесі сегіз кілт алынады, олар k_8, k_9, ..., k_{17} деп белгіленеді. Бұл процедураны *56* кілтті алғанға дейін қайталайды. Сонғы төрт кілт (k_{53}, k_{54}, k_{55} және k_{56}) *IDEA* шифрында қолданылмайды және лақтырылып тасталады.

Шифрлаудың раундтық кілттерін кеңейтілген кілттен K_P таңдау келесі ретпен орындалады: бірінші раунд кілттері ретінде $k_1^{(1)}$, $k_2^{(1)}$, $k_3^{(1)}$, $k_4^{(1)}$, $k_5^{(1)}$ және $k_6^{(1)}$ кеңейтілген кілттің K_P бірінші алты кілті алынады - k_1, k_2, k_3, k_4, k_5 және k_6; екінші раунд кілттері ретінде $k_1^{(2)}$, $k_2^{(2)}$, $k_3^{(2)}$, $k_4^{(2)}$, $k_5^{(2)}$ және $k_6^{(2)}$ кеңейтілген кілттің K_P келесі алты кілті алынады - k_7, k_8, k_9, k_{10}, k_{11} және k_{12} және ары қарай. Соңғы

түрлендіру (тоғызыншы раунд) кілттері ретінде $k_1^{(9)}$, $k_2^{(9)}$, $k_3^{(9)}$ және $k_4^{(9)}$ кеңейтілген кілттің K_P соңғы төрт кілті k_{49}, k_{50}, k_{51} және k_{52} алынады.

Шифрлау кілтінен K кеңейтілген кілтті K_P алу процессі, сонымен қатар раундтық кілттерді таңдау 9.5 суретінде көрсетілген және 9.1 кестеге жиналған.



9.5 сурет – Шифрлаудың раундтық кілттерін алу процессі

9.1 кесте

IDEA әр шифрлау раунды үшін кілттер

Раунд нөмірі	*Раундта кілт нөмірі*					
	1	*2*	*3*	*4*	*5*	*6*
1	$k_1^{(1)}$	$k_2^{(1)}$	$k_3^{(1)}$	$k_4^{(1)}$	$k_5^{(1)}$	$k_6^{(1)}$
2	$k_1^{(2)}$	$k_2^{(2)}$	$k_3^{(2)}$	$k_4^{(2)}$	$k_5^{(2)}$	$k_6^{(2)}$
3	$k_1^{(3)}$	$k_2^{(3)}$	$k_3^{(3)}$	$k_4^{(3)}$	$k_5^{(3)}$	$k_6^{(3)}$
4	$k_1^{(4)}$	$k_2^{(4)}$	$k_3^{(4)}$	$k_4^{(4)}$	$k_5^{(4)}$	$k_6^{(4)}$

Раунд нөмірі	*Раундта кілт нөмірі*					
	1	*2*	*3*	*4*	*5*	*6*
5	$k_1^{(5)}$	$k_2^{(5)}$	$k_3^{(5)}$	$k_4^{(5)}$	$k_5^{(5)}$	$k_6^{(5)}$
6	$k_1^{(6)}$	$k_2^{(6)}$	$k_3^{(6)}$	$k_4^{(6)}$	$k_5^{(6)}$	$k_6^{(6)}$
7	$k_1^{(7)}$	$k_2^{(7)}$	$k_3^{(7)}$	$k_4^{(7)}$	$k_5^{(7)}$	$k_6^{(7)}$
8	$k_1^{(8)}$	$k_2^{(8)}$	$k_3^{(8)}$	$k_4^{(8)}$	$k_5^{(8)}$	$k_6^{(8)}$
Соңғы түрлендіру (*9* раунд)	$k_1^{(9)}$	$k_2^{(9)}$	$k_3^{(9)}$	$k_4^{(9)}$		

Раундтың әр бірінші кілті кілттің *K* өз ішкі көптік биттерінен алынатынын айтып кетейік. Егер бүкіл кілтті *K[1..128]* деп белгілесек, онда сегіз раундта бірінші кілттерге *K* кілтінің ішкі көптік биттері кіреді (9.2 кестені қараңыз).

9.2 кесте

IDEA әр шифрлау раунды үшін бірінші кілттер мәндері

Раунд нөмірі	*Кеңейтілген кілттің кілттері* K_P	*Раунд кілті*	*Шифрлау кілтінің биттері K*
1	k_1	$k_1^{(1)}$	*K[1...16]*
2	k_7	$k_1^{(2)}$	*K[97...112]*
3	k_{13}	$k_1^{(3)}$	*K[90...105]*
4	k_{19}	$k_1^{(4)}$	*K[83...98]*
5	k_{25}	$k_1^{(5)}$	*K[77...91]*
6	k_{31}	$k_1^{(6)}$	*K[44...59]*
7	k_{37}	$k_1^{(7)}$	*K[37...52]*
8	k_{43}	$k_1^{(8)}$	*K[30...45]*
9	k_{49}	$k_1^{(9)}$	*K[23...38]*

Бірінші және сегізіншіден басқа әр раундта шифрлау кілтінің *K* тек *96* биты қолданса да, ол кілттің биттер көптігі әр итерацияда қиылыспайды, және әртүрлі раундтың кілттер арасында қарапайым жылжу қатынасы жоқ. Бұл келесі себеппен орындалады: әр раундта

тек алты раундтық кілт қолданылады, ал кілттің әр айналымында кеңейтілген кілттің K_P сегіз кілті алынады.

9.1 мысал. Деректерді шифрлау кілті K келесіге тең болсын:

$$K = 0001\ 0002\ 0003\ 0004\ 0005\ 0006\ 0007\ 0008_{16.}$$

Шифрланған деректердің раундтық кілттер мәндерін анықтау қажет.

Шешім. Кеңейтілген кілттің K_P кілттерін шифрлау кілтін K сегіз бөлімге бөлу және деректерді шифрлау кілтінің K биттерін циклдық жылжыту жолымен анықтаймыз. Деректерді 9.3 кестеге жинаймыз.

9.3 кесте

IDEA шифрлаудың әр раунды үшін кілт мысалдары

Раунд	$k_1^{(i)}$	$k_2^{(i)}$	$k_3^{(i)}$	$k_4^{(i)}$	$k_5^{(i)}$	$k_6^{(i)}$
1	*0001*	*0002*	*0003*	*0004*	*0005*	*0006*
2	*0007*	*0008*	*0400*	*0600*	*0800*	*0a00*
3	*0c00*	*0e00*	*1000*	*0200*	*0010*	*0014*
4	*0018*	*001c*	*0020*	*0004*	*0008*	*000c*
5	*2800*	*3000*	*3800*	*4000*	*0800*	*1000*
6	*1800*	*2000*	*0070*	*0080*	*0010*	*0020*
7	*0030*	*0040*	*0050*	*0060*	*0000*	*2000*
8	*4000*	*6000*	*8000*	*a000*	*c000*	*e001*
9	*0080*	*00c0*	*0100*	*0140*	-	-

IDEA деректерді керішифрлау үшін раундтық кілттерді генерациялау

Деректерді керішифрлау үшін есептеу әдісі шифрлау кезіндегідей. Тек бір айырмашылығы бар, керішифрлау үшін басқа раундтық кілттер қолданылады. Керішифрлау процессінде раундтық кілттер шифрлау процессіне қарағанда кері ретпен қолдану керек.

Керішифрлаудың раундтық кілттерін қалыптастыру үшін алдымен шифрлаудың раундтық кілттерін қалыптастырады.

Керішифрлаудың раундтық кілттерін құру процессі келесімен анықталады. Керішифрлаудың *i*-ші раундының бірінші және төртінші кілттері мультипликативті инверсиямен шифрлаудың (*10*-

i)-ші раундының бірінші және төртінші кілттерінен жасалады. *1*-ші және *9*-ші раундтар үшін керішифрлаудың екінші және үшінші кілттері аддитивті инверсия арқылы шифрлаудың *9*-ші және *1*-ші раундтың екінші және үшінші кілттерінен жасалады. *2*-ші раундтан *8*-ші раундқа дейін керішифрлаудың екінші және үшінші кілттері аддитивті инверсиямен шифрлаудың *8*-ші раундтан *2*-ші раундқа дейінгі раундардың екінші және үшінші кілттерінен жасалады. Керішифрлаудың *i*-ші раундының соңғы екі кілті шифрлаудың (*9*-*i*)-ші раундтың соңғы екі кілтіне тең.

Раундтық кілт *k* мультипликативті инверсиясын

$$k^{-1} = \frac{1}{k}$$

деп белгілейміз және сонда

$$\frac{1}{k} \cdot k \bmod (2^{16} + 1) = 1. \qquad (9.1)$$

$2^{16} + 1$ – жай сан болғандықтан, онда әр бүтін нөлге тең емес *k* $2^{16} + +1$ модулі бойынша бірегей мультипликативті инверсиясы болады.

Раундтық кілттің *k* аддитивті инверсиясын $-k$ деп белгілейміз және сонда

$$(-k + \cdot k) \bmod (2^{16} + 1) = 0. \qquad (9.2)$$

IDEA алгоритмін жүзеге асыру үшін болжам қабылданды: нөлдік субблок $2^{16} = -1$ тең, ал мультипликативті кері шама *0* тең болады *0* [7, 29]. Мультипликативті кері шамалар мәндерін есептеу кейбір шығындарды талап етеді, бірақ оны тек бір рет керішифрлаудың әр кілтіне орындайды.

Көрсетілгенге байланысты керішифрлау кілттері әртүрлі раундтар үшін 9.4 кестеде берілген.

9.2 мысал. Деректерді шифрлау кілті *K* келесі болсын

$$K = 0001\ 0002\ 0003\ 0004\ 0005\ 0006\ 0007\ 0008_{16}.$$

Деректерді керішифрлаудың раундтық кілттер мәндерін анықтау қажет.

9.4 кесте

IDEA керішифрлаудың әр раунды үшін кілттер

Раунд нөмірі	*Раундта кілт нөмірі*					
	1	*2*	*3*	*4*	*5*	*6*
1	$1/k_1^{(9)}$	$-k_2^{(9)}$	$-k_3^{(9)}$	$1/k_4^{(9)}$	$k_5^{(8)}$	$k_6^{(8)}$
2	$1/k_1^{(8)}$	$-k_3^{(8)}$	$-k_2^{(8)}$	$1/k_4^{(8)}$	$k_5^{(7)}$	$k_6^{(7)}$
3	$1/k_1^{(7)}$	$-k_3^{(7)}$	$-k_2^{(7)}$	$1/k_4^{(7)}$	$k_5^{(6)}$	$k_6^{(6)}$
4	$1/k_1^{(6)}$	$-k_3^{(6)}$	$-k_2^{(6)}$	$1/k_4^{(6)}$	$k_5^{(5)}$	$k_6^{(5)}$
5	$1/k_1^{(5)}$	$-k_3^{(5)}$	$-k_2^{(5)}$	$1/k_4^{(5)}$	$k_5^{(4)}$	$k_6^{(4)}$
6	$1/k_1^{(4)}$	$-k_3^{(4)}$	$-k_2^{(4)}$	$1/k_4^{(4)}$	$k_5^{(3)}$	$k_6^{(3)}$
7	$1/k_1^{(3)}$	$-k_3^{(3)}$	$-k_2^{(3)}$	$1/k_4^{(3)}$	$k_5^{(2)}$	$k_6^{(2)}$
8	$1/k_1^{(2)}$	$-k_3^{(2)}$	$-k_2^{(2)}$	$1/k_4^{(2)}$	$k_5^{(1)}$	$k_6^{(1)}$
9	$1/k_1^{(1)}$	$-k_2^{(1)}$	$-k_3^{(1)}$	$1/k_4^{(1)}$		

Шешім. 9.1 мысалында анықталған деректерді шифрлаудың раундтық кілттерінің мәндерін қолданамыз. Олар 9.3 кестеде келтірілген. Деректерді керішифрлаудың раундтық кілттер мәндерін 9.4 кесте және (9.1) және (9.2) өрнектері көмегімен анықтаймыз. Деректерді керішифрлаудың анықталған раундтық кілттер мәндерін 9.5 кестесіне жинаймыз.

9.5 кесте

IDEA шифрлаудың әр раунды үшін кілт мысалдары

Раунд	$k_1^{(i)}$	$k_2^{(i)}$	$k_3^{(i)}$	$k_4^{(i)}$	$k_5^{(i)}$	$k_6^{(i)}$
1	*fe01*	*ff40*	*ff00*	*659a*	*c000*	*e001*
2	*fffd*	*8000*	*a000*	*cccc*	*0000*	*2000*
3	*a556*	*ffb0*	*ffc0*	*52ab*	*0010*	*0020*
4	*554b*	*ff90*	*e000*	*fe01*	*0800*	*1000*
5	*332d*	*c800*	*d000*	*fffd*	*0008*	*000c*
6	*4aab*	*ffe0*	*ffe4*	*c001*	*0010*	*0014*
7	*aa96*	*f000*	*f200*	*ff81*	*0800*	*0a00*
8	*4925*	*fc00*	*fff8*	*552b*	*0005*	*0006*
9	*0001*	*fffe*	*fffd*	*c001*	–	–

9.2.5. IDEA деректерді шифрлау

IDEA симметриялық блокты шифр және жоғарыда айтылғандай, шифрлау процессі керішифрлау процессімен бірдей. *IDEA* деректерді шифрлау (керішифрлау) алгоритмінің құрылымдық сұлбасы 9.1 - 9.4 суреттерінде көрсетілген.

Деректерді шифрлау (керішифрлау) процессі деген сегіз бірдей раунд, ал тоғызыншы - шығыс түрлендіру.

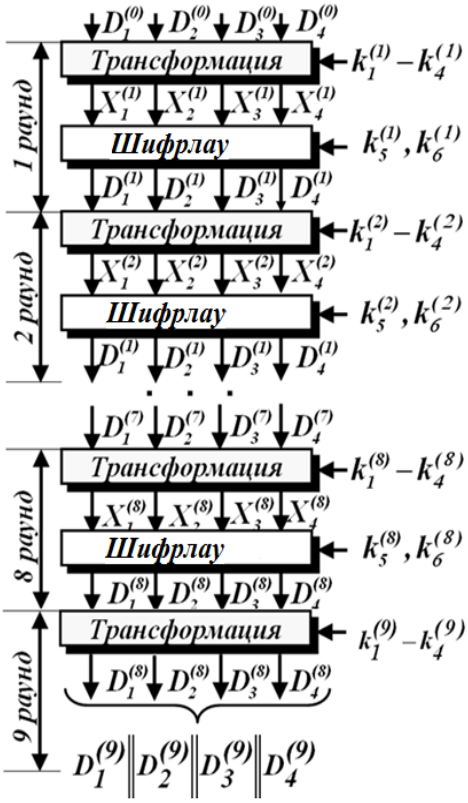
IDEA шифрлаудың (керішифрлаудың) жалпы сұлбасын қарастырамыз (9.6 сурет).

Кез келген алгоритмде сияқты бұл жерде екі кіріс бар: алғашқы деректер блогы және раундтық кілттер.

Өзгерту деген 9.7 суретте көрсетілген бір қатар операциялар, ал шифрлау 9.8 суретте көрсетілген бір қатар операциялар.

64-биттік деректер блогы төрт *16*-биттік ішкі блокқа бөлінеді (9.3 суретіне қараңыз):

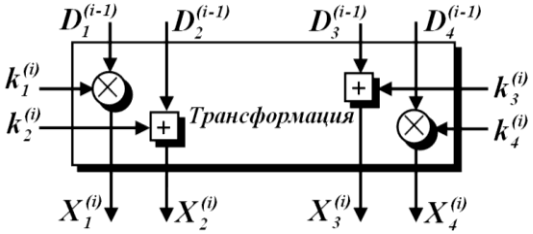
$D_1^{(0)},\ D_2^{(0)},\ D_3^{(0)},\ D_4^{(0)}$.



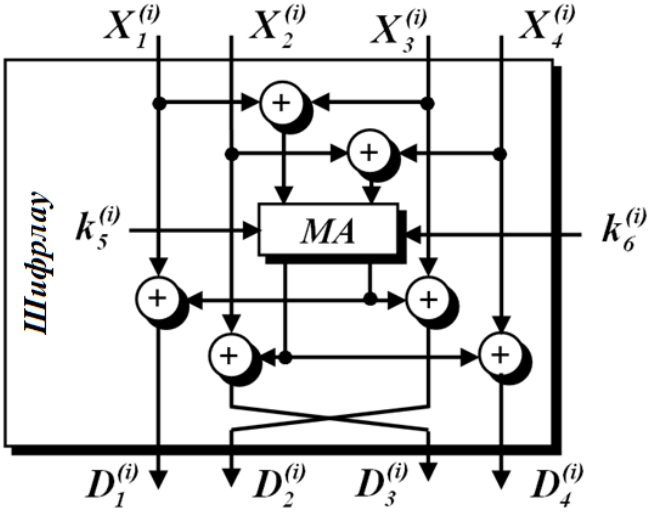
9.6 сурет - *IDEA* деректерді шифрлау алгоритмінің құрылымдық сұлбасы

Бұл төрт ішкі блок шифрлау алгоритмінің бірінші раундының кірісі болады. Барлығы сегіз раунд орындалады.

Раундтар арасында екінші және үшінші ішкі блоктар орын ауыстырады (9.3 суретті қараңыз).



9.7 сурет - *IDEA* шифрының раундында деректерді өзгерту процессінің құрылымы



9.8 сурет - *IDEA* шифрының раундында деректерді шифрлау процессінің құрылымы

Әр раундта (тоғызыншыдан басқа) келесі операциялар тізбегі орын алады (9.8 суретке қараңыз):

1. Ішкі блок $D_1^{(i-1)}$ және бірінші раундтық кілтті $k_1^{(i)}$ 2^{16} + *1* (*65537*) модулі бойынша көбейту:

$$X_1^{(i)} = D_1^{(i-1)} \otimes k_1^{(i)}.$$

2. Ішкі блок $D_2^{(i-1)}$ және екінші раундтық кілтті $k_2^{(i)}$ 2^{16} (*65536*) модулі бойынша қосу:

$$X_2^{(i)} = D_2^{(i-1)} \boxplus k_2^{(i)}.$$

3. Ішкі блок $D_3^{(i-1)}$ және үшінші раундтық кілтті $k_3^{(i)}$ 2^{16} (*65536*) модулі бойынша қосу:

$$X_3^{(i)} = D_3^{(i-1)} \boxplus k_3^{(i)}.$$

4. Ішкі блок $D_4^{(i-1)}$ және төртінші раундтық кілтті $k_4^{(i)}$ $2^{16} + 1$ (*65537*) модулі бойынша көбейту:

$$X_4^{(i)} = D_4^{(i-1)} \otimes k_4^{(i)}.$$

5. (1) және (3) қадамдардың нәтижелерін *2* модулі бойынша қосу:

$$A^{(i)} = X_1^{(i)} \oplus X_3^{(i)}.$$

6. (2) және (4) қадамдардың нәтижелерін *2* модулі бойынша қосу:

$$B^{(i)} = X_2^{(i)} \oplus X_4^{(i)}.$$

7. (5) қадамның нәтижесін және бесінші раундтық кілтті $k_5^{(i)}$ $2^{16} + 1$ (*65537*) модулі бойынша көбейту:

$$C^{(i)} = A^{(i)} \otimes k_5^{(i)}.$$

8. (6) және (7) қадамдардың нәтижелерін 2^{16} (*65536*) модулі бойынша қосу:

$$E^{(i)} = B^{(i)} \boxplus C^{(i)}.$$

9. (8) қадамның нәтижесін және алтыншы раундтық кілтті $k_6^{(i)}$ $2^{16} + 1$ (*65537*) модулі бойынша көбейту (9.2 суретті қараңыз):

$$F^{(i)} = E^{(i)} \otimes k_6^{(i)}.$$

10. (7) және (9) қадамдардың нәтижелерін 2^{16} (*65536*) модулі бойынша қосу:

$$G^{(i)} = C^{(i)} \boxplus F^{(i)}.$$

11. (1) және (9) қадамдардың нәтижелерін *2* модулі бойынша қосу:

$$D_1^{(i)} = X_1^{(i)} \oplus F^{(i)}.$$

12. (3) және (9) қадамдардың нәтижелерін *2* модулі бойынша қосу:

$$D_2^{(i)} = X_3^{(i)} \oplus F^{(i)}.$$

13. (2) және (10) қадамдардың нәтижелерін *2* модулі бойынша қосу:

$$D_3^{(i)} = X_2^{(i)} \oplus G^{(i)}.$$

14. (4) және (10) қадамдардың нәтижелерін *2* модулі бойынша қосу:

$$D_4^{(i)} = X_4^{(i)} \oplus G^{(i)}.$$

Раундтың шығысы 11–14 қадамдарды орындау нәтижелері ретінде алынатын төрт ішкі блок болады, нәтижесінде келесі раунд үшін кіріс қалыптастырылады.

Сегізінші раундтан кейін соңға түрлендіру орындалады (9.7 суретін қараңыз):

1. Ішкі блок $D_1^{(8)}$ және бірінші раундтық кілтті $k_1^{(9)}$ $2^{16} + 1$ (*65537*) модулі бойынша көбейту:

$$D_1^{(9)} = D_1^{(8)} \otimes k_1^{(9)}. \qquad (9.3)$$

2. Ішкі блок $D_3^{(8)}$ және екінші раундтық кілтті $k_2^{(9)}$ 2^{16} (*65536*) модулі бойнша қосу:

$$D_2^{(9)} = D_3^{(8)} \boxplus k_2^{(9)}. \qquad (9.4)$$

3. Ішкі блок $D_2^{(8)}$ және екінші раундтық кілтті $k_3^{(9)}$ 2^{16} (*65536*) модулі бойнша қосу:

$$D_3^{(9)} = D_2^{(8)} \boxplus k_3^{(9)}. \qquad (9.5)$$

4. Ішкі блок $D_4^{(8)}$ және бірінші раундтық кілтті $k_4^{(9)}$ $2^{16} + 1$ (*65537*) модулі бойынша көбейту:

$$D_4^{(9)} = D_4^{(8)} \otimes k_4^{(9)}. \quad (9.6)$$

Соңғы түрлендіруден кейін алынған бұл нәтижелі ішкі блоктар шифрланған деректерді алу үшін біріктіріледі

$$C = D_1^{(9)} || D_2^{(9)} || D_3^{(9)} || D_4^{(9)}.$$

Содан кейін алғашқы деректердің келесі *64*-биттік блогы алынады да және шифрлау алгоритмі қайталанады. Бұл ашық деректердің барлық *64*-биттік блоктары шифрланғанға дейін орындалады.

9.3 мысал. Деректерді шифрлау кілті К келесіге тең болсын

$$K = 0001\ 0002\ 0003\ 0004\ 0005\ 0006\ 0007\ 0008_{16.}$$

Шифрлау үшін алғашқы деректер

$$M = 0000\ 0001\ 0002\ 0003_{16.}$$

Берілген деректерді шифрлау нәтижесін алу қажет.

Шешім. IDEA алгоритмімен деректерді шифрлау процессі 9.6 кесте көмегімен түсіндіріледі

9.6 кесте

Шифрлаудың әр раунды үшін раундтық кілттер және деректердің ішкі блоктары

Раунд	*Раундтық кілттер*						*Деректер ішкі блоктарының мәндері*			
	$k_1^{(i)}$	$k_2^{(i)}$	$k_3^{(i)}$	$k_4^{(i)}$	$k_5^{(i)}$	$k_6^{(i)}$	$D_1^{(0)}$	$D_2^{(0)}$	$D_3^{(0)}$	$D_4^{(0)}$
							0000	*0001*	*0002*	*0003*
1	*0001*	*0002*	*0003*	*0004*	*0005*	*0006*	*00f0*	*00f5*	*010a*	*0105*
2	*0007*	*0008*	*0400*	*0600*	*0800*	*0a00*	*222f*	*21b5*	*f45e*	*e959*
3	*0c00*	*0e00*	*1000*	*0200*	*0010*	*0014*	*0f86*	*39be*	*8ee8*	*1173*
4	*0018*	*001c*	*0020*	*0004*	*0008*	*000c*	*57df*	*ac58*	*c65b*	*ba4d*
5	*2800*	*3000*	*3800*	*4000*	*0800*	*1000*	*8e81*	*ba9c*	*f77f*	*3a4a*
6	*1800*	*2000*	*0070*	*0080*	*0010*	*0020*	*6942*	*9409*	*e21b*	*1c64*
7	*0030*	*0040*	*0050*	*0060*	*0000*	*2000*	*99d0*	*c7f6*	*5331*	*620e*
8	*4000*	*6000*	*8000*	*a000*	*c000*	*e001*	*0a24*	*0098*	*ec6b*	*4925*
9	*0080*	*00c0*	*0100*	*0140*	-	-	*11fb*	*ed2b*	*0198*	*6de5*

Сонымен, алғашқы деректерді шифрлау нәтижесі келесіге тең болады:

$$C = 11fb\ ed2b\ 0198\ 6de5_{16}.$$

IDEA деректерді керішифрлау

IDEA – симметриялық блокты шифр және алдында айтылғандай, керішифрлау процессі шифрлау процессімен бірдей, сондықтан *IDEA* деректерді керішифрлау алгоритмінің құрылымдық сұлбасы шифрлау алгоритмінің сұлбасына сәйкес келеді (9.6 сурет).

Керішифрлау кезінде *IDEA* сол құрылымына кіріс ретінде шифрланған деректерді, бірақ шифрлауға қарағанда басқа раундтық кілттер жинағымен қолданады.

Сәйкес раундтық кілттермен керішифрлау алгоритмі дұрыс нәтиже беретінін дәлелдеу үшін шифрлау және керішифрлау процесстерін бір уақытта қарастырайық. Сегіз раундтың әрқайсысы, 9.6, 9.7 және 9.8 суреттерінде көрсетілгендей, екі түрлендіру кезеңіне бөлінген, біріншісі - өзгерту деп аталады, ал екіншісі - шифрлау.

9.7 суретінің тікбұрыштарында орындалатын түрлендірулерді қарастырамыз. Өзгерту шығысында шифрлау кезінде келесі арақатынастар (9.3)…(9.6) қолданылады.

Деректерді керішифрлау кезінде, 9.6 суретінде берілген жүйенің кірісіне келесі деректерді беру қажет:

$$D_1^{(0)} = D_1^{(9)},\ D_2^{(0)} = D_2^{(9)},\ D_3^{(0)} = D_3^{(9)},\ D_4^{(0)} = D_4^{(9)}.$$

9.6 суретінде берілген керішифрлау сұлбасының бірінші раундының бірінші сатысы келесі арақатынастарды қолдайды:

$$\begin{aligned} X_1^{(1)} &= D_1^{(9)} \otimes 1/k_1^{(9)}; \quad X_2^{(1)} = D_2^{(9)} \boxplus 1/k_2^{(9)}; \\ X_3^{(1)} &= D_3^{(9)} \boxplus 1/k_3^{(9)}; \quad X_4^{(1)} = D_4^{(9)} \otimes 1/k_4^{(9)}. \end{aligned} \tag{9.7}$$

(9.3), (9.4), (9.5) және (9.6) сәйкес мәндерін (9.7) қойып, келесіні аламыз:

$$
\begin{aligned}
X_1^{(1)} &= D_1^{(8)} \otimes 1/k_1^{(9)} \otimes k_1^{(9)} = D_1^{(8)};\\
X_2^{(1)} &= D_3^{(8)} \boxplus (-k_2^{(9)}) \boxplus k_2^{(9)} = D_3^{(8)};\\
X_3^{(1)} &= D_2^{(8)} \boxplus (-k_3^{(9)}) \boxplus k_3^{(9)} = D_2^{(8)};\\
X_4^{(1)} &= D_4^{(8)} \otimes 1/k_4^{(9)} \otimes k_4^{(9)} = D_4^{(8)}.
\end{aligned}
\tag{9.8}
$$

Сонымен, керішифрлау процессінің бірінші сатысының шығысы екінші және үшінші блоктарының кезектесуін аластамаумен шифрлау процессінің соңғы сатысының кірісіне эквивалентті.

Сегізінші раундтың шығысында ішкі блоктарының мәндері 9.3 және 9.6 суреттерінің сұлбасына сәйкес келесі арақатынастармен анықталады:

$$
\begin{aligned}
D_1^{(8)} &= X_1^{(8)} \oplus MA_R(X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)});\\
D_2^{(8)} &= X_3^{(8)} \oplus MA_R(X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)});\\
D_3^{(8)} &= X_2^{(8)} \oplus MA_L(X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)});\\
D_4^{(8)} &= X_4^{(8)} \oplus MA_L(X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)}),
\end{aligned}
\tag{9.9}
$$

бұл жерде $MA_R(X,Y)$ – X және Y кірістері бар MA құрылымының оң жақтағы кірісі; $MA_L(X,Y)$ – X және Y кірістері бар MA құрылымының сол жақтағы шығысы.

Бірінші раундтың шығысында 9.3 және 9.6 суреттерінің сұлбасына сәйкес ішкі блок $D_1^{(1)}$ мәні келесіге тең болады:

$$
\begin{aligned}
D_1^{(1)} &= X_1^{(1)} \oplus MA_R(X_1^{(1)} \oplus X_2^{(1)}, X_3^{(1)} \oplus X_4^{(1)}) =\\
&= D_1^{(8)} \oplus MA_R(D_1^{(8)} \oplus D_2^{(8)}, D_3^{(8)} \oplus D_4^{(8)}).
\end{aligned}
\tag{9.10}
$$

(9.9) мәндерін (9.10) қойып келесіні аламыз

$$D_1^{(1)} = X_1^{(8)} \oplus MA_R(\, X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)}\,) \oplus$$
$$\oplus\, MA_R(\, X_1^{(8)} \oplus MA_R(\, X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)}\,) \oplus$$
$$\oplus\, X_3^{(8)} \oplus MA_R(\, X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)}\,), \qquad (9.11)$$
$$X_2^{(8)} \oplus MA_L(\, X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)}\,) \oplus$$
$$\oplus\, X_4^{(8)} \oplus MA_L(\, X_1^{(8)} \oplus X_3^{(8)}, X_2^{(8)} \oplus X_4^{(8)}\,)).$$

(9.11) түрлендірулерді орындап қорытынында аламыз

$$D_1^{(1)} = X_1^{(8)}.$$

Тұра осылай бірінші раунд шығысындағы ішкі блоктарының басқа мәндерін алуға болады:

$$D_2^{(1)} = X_3^{(8)}, \quad D_3^{(1)} = X_2^{(8)}, \quad D_4^{(1)} = X_4^{(8)}.$$

Сонымен, керішифрлау процессінің екінші сатысының шығысы екінші және үшінші блоктарының кезектесуін аластамаумен шифрлау процессінің соңғыдан бұрынғы сатысының кірісіне эквивалентті.

Тұра осылай келесіні көрсетуге болады

$$D_1^{(8)} = X_1^{(1)}, \quad D_2^{(8)} = X_3^{(1)}, \quad D_3^{(8)} = X_2^{(1)}, \quad D_4^{(8)} = X_4^{(1)}.$$

Ақырында, керішифрлау процессінің өзгерту шығысы екінші және үшінші блоктарының кезектесуін аластамаумен шифрлау процессінің бірінші сатысына эквивалетті болғандықтан, керішифрлаудың барлық процессінің шығысы шифрлау процессінің кірісіне эквивалентті болады. Басқа сөзбен, екі шектес (жұпты және тақ) раундтарда деректердің ішкі блоктарының алмасуы екі рет орындалады. Сондықтан, сегізінші раунд шығысында деректердің ішкі блок мәндері орнықты ретке келеді. Тоғызыншы раундта деректердің ішкі блоктардың жүру реті екі рет өзгереді және сондықтан раунд шығысында жүру реті орнықты болады.

Сонымен, сәйкес кілттері бар керішифрлау алгоритмінің нәтижесі дұрыс болатыны дәлелденді.

9.4 мысал. Деректерді шифрлау кілті *K* келесіге тең болсын:

$$K = 0001\ 0002\ 0003\ 0004\ 0005\ 0006\ 0007\ 0008_{16}.$$

Шифрланған деректер тең:

$$C = 11fbed2b01986de5_{16}.$$

Керішифрлау нәтижесін алу қажет.

Шешім. IDEA алгоритмімен деректерді керішифрлау процессі 9.7 кесте көмегімен түсіндіріледі.

Сонымен, шифрланған деректерді керішифрлау нәтижесі келесіге тең болады:

$$M = 0000\ 0001\ 0002\ 0003_{16}.$$

Алынғын нәтижеден және 9.3 мысал нәтижесінен керішифрлау дұрыс орындалғаны көрініп тұр.

9.7 кесте

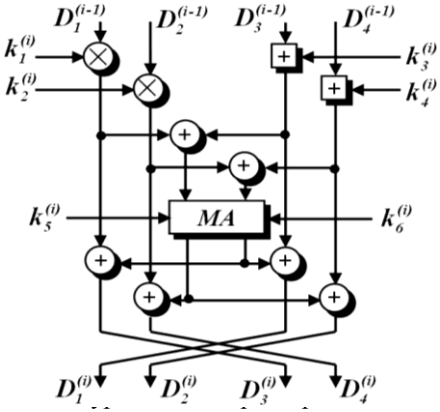
Шифрлаудың әр раунды үшін раундтық кілттер және деректердің ішкі блоктары

Раунд	*Раундтық кілттер*						*Деректердің ішкі блок мәндері*			
	$k_1^{(i)}$	$k_2^{(i)}$	$k_3^{(i)}$	$k_4^{(i)}$	$k_5^{(i)}$	$k_6^{(i)}$	$D_1^{(0)}$	$D_2^{(0)}$	$D_3^{(0)}$	$D_4^{(0)}$
							11fb	*ed2b*	*0198*	*6de5*
1	*fe01*	*ff40*	*ff00*	*659a*	*c000*	*e001*	*d98d*	*d331*	*27f6*	*82b8*
2	*fffd*	*8000*	*a000*	*cccc*	*0000*	*2000*	*bc4d*	*e26b*	*9449*	*a576*
3	*a556*	*ffb0*	*ffc0*	*52ab*	*0010*	*0020*	*0aa4*	*f7ef*	*da9c*	*24e3*
4	*554b*	*ff90*	*e000*	*fe01*	*0800*	*1000*	*ca46*	*fe5b*	*dc58*	*116d*
5	*332d*	*c800*	*d000*	*fffd*	*0008*	*000c*	*748f*	*8f08*	*39da*	*45cc*
6	*4aab*	*ffe0*	*ffe4*	*c001*	*0010*	*0014*	*3266*	*045e*	*2fb5*	*b02e*
7	*aa96*	*f000*	*f200*	*ff81*	*0800*	*0a00*	*0690*	*050a*	*00fd*	*1dfa*
8	*4925*	*fc00*	*fff8*	*552b*	*0005*	*0006*	*0000*	*0005*	*0003*	*000c*
9	*0001*	*fffe*	*fffd*	*c001*	-	-	*0000*	*0001*	*0002*	*0003*

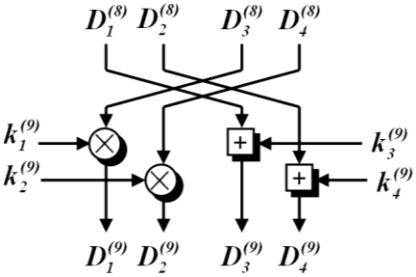
9.2.6. IDEA қауіпсіздігі

IDEA алгоритмін талдау

IDEA алгоритмі *PES* алгоритмін шамалы өзгерту нәтижесінде пайда болды. 9.9 суретінде *PES* алгоритммен деректерді шифрлаудың сегіз раундтың біреуінің құрылымы келтірілген, ал 9.10 суретінде - соңғы түрлендірудің.



9.9 сурет - *PES* алгоритммен деректерді үрлендірудің бір раунды



9.10 сурет - *PES* соңғы түрлендіру (9 раунд)

9.4 және 9.9 суреттерінде, 9.4 және 9.10 суреттерінде құрылымдарды салыстырулардан өзгерістер өте көп емес екені көрініп тұр.

PES алгоритміне қарағанда *IDEA* алгоритмінде келесі өзгерістер орындалған (барлық раундтарда, соның ішінде тоғызыншыда да):

- ішкі блокты $D_2^{(i)}$ екінші раундтық кілпен $k_2^{(i)}$ 2^{16} модулі бойынша көбейту оларды $2^{16}+1$ модулі бойынша қосумен ауыстырылған;

- ішкі блокты $D_4^{(i)}$ төртінші раундтық кілтпен $k_4^{(i)}$ $2^{16}+1$ модулі бойынша қосу оларды 2^{17} модулі бойынша көбейтумен ауыстырылған.

Осымен қоса, *PES* алгоритміне қарағанда *IDEA* алгоритмінде раунд соңында (тек бірінші сегіз раундта) деректер ішкі блоктарының жылжуы өзгерген.

Сонымен қатар, *PES* алгоритміне қарағанда *IDEA* алгоритмінде соңғы түрлендірудің басында деректердің ішкі блоктарын жылжыту өзгертілген.

Әлемде белгілі критологтарының бірі *Б. Шнайер* өзінің "*Прикладная криптография*" кітабінде: *"...осындай шамалы өзгерістер қалай сондай үлкен айырмашылықтарға әкелу мүмкін, ғажап"* деп жазды [47].

1996 жылы шыққан сол кітапта *Б. Шнайер IDEA* туралы келесіні жазды: *"Менің ойымша, бұл осы кезге дейін жарияланған ең жақсы және сенімді блокты шифр"*.

IDEA алгоритмінде *64*-биттік деректер блоктары пайдаланылады. Алғашқы хабардың статистикалық сипаттамаларын жасыру үшін блок ұзындығы жеткілікті болу қажет. Бірақ блок көлемін үлкейтумен криптографиялық алгоритмді жүзеге асыру қиындығы экспонентті жоғарлайды.

IDEA алгоритмінде әлсіз кілттер және кілттердегі әлсіздік

IDEA алгоритмінде *128*-биттік кілт қолданылады. Кілтті талдау мүмкіндігін болдырмау үшін кілт ұзындығы жеткілікті үлкен болу қажет. Ашық және оларға сәйкес келетін шифрланған деректер белгілі шартымен, *128*-биттік кілтті, кілттерді толық талдау арқылы ашу үшін 2^{128} (шамамен 10^{38}) шифрлау қажет болады. Бұндай кілт ұзындығымен *IDEA* жеткілікті қауіпсіз деп есептеледі. *IDEA* жоғары криптографиялық беріктілігі сонымен қатар келесі сипаттамалармен қамтамасыз етіледі:

- *шатастыру* – шифрлау кілттен қиын және шатасқан түрмен тәуелді;
- *шашырату* – алғашқы деректердің әр биты шифрланған деректердің әр битіне әсер етеді.

Сюэцзя Лай (*Xuejia Lai*) және *Джеймс Мэсси* (*James Massey*) *IDEA* дифференциалды криптографиялық талдауға криптографиялық беріктілігін анықтау мақсатында оған мұқият талдауды өткізді. Ол үшін олар марктік шифр түсінігін енгізді және дифференциалдық криптографиялық талдауға беріктілік модельдену және санды бағалану мүмкін екенін көрсетті. *IDEA* сызықты немесе алгебралық әлсіздіктер табылмады. *Бихам* (*Biham*) өткізген байланысқан кілттермен криптографиялық талдау көмегімен ашу талпыныс та нәтиже бермеді [20, 26, 44].

Аз раундтар саны (толық *IDEA* 8 толық және бір толық емес раунд) бар *IDEA* қолданатын сәтті шабуылдар бар. Егер шабуыл көмегімен шифрды ашудың операциялар саны кілттерді толық талдау кезіндегі операциялар санынан аз болса, онда ол сәтті деп есептеледі. *Вилли Майердің* (*Willi Meier*) ашу әдісі кілттерді толық талдау әдісінен тек екі раунды бар *IDEA* үшін тиімдірек болды. *"Ортада кездесу"* әдісімен 4,5 раунды бар *IDEA* ашылды. Ол үшін

кодтар сөздігінен барлық 2^{64} блоктарды білу талап етіледі және талдау қиындығы 2^{112} операцияны құрайды. 2007 жылға ең жақсы шабуыл барлық кілттерге пайдаланымды және *6* раунды бар *IDEA* бұза алады [7, 29].

IDEA әлсіз кілттерінің үлкен кластары бар. Олар әлсіз келесі мәнде: кілт осы класқа жататынын және содан кейін, кілттің өзін анықтайтын процедуралар бар. Кәзіргі уақытта келесі белгілі:

1. $2^{23} + 2^{35} + 2^{51}$ дифференциалдық криптографиялық талдауға әлсіз кілттер. Іріктелген ашық деректер көмегімен 2^{12} операциямен 2^{51} класына кіретінін есептеуге болады. Берілген шабуылдың авторлары *IDEA* алгоритмінің өзгертуін ұсынды. Бұл өзгерту раундтық кілттерді $k_j^{(i)}$ сәйкестерге ауыстыруда

$$k_j^{'(i)} = a \oplus k_j^{(i)},$$

бұл жерде i – шифрлау раунд нөмірі; j – раундтық кілт нөмірі.

A нақты мәні маңызды емес. Мысалы, $a = 0dae_{16}$ (он алтылық санақ жүйесінде) кезде берілген әлсіз кілттер аластатылады.

2. 2^{63} сызықты криптографиялық талдауға әлсіз кілттер. Бұл класқа жататындығы байланысқан кілттерде тест көмегімен анықталады.

3. $2^{53} + 2^{56} + 2^{64}$ әлсіз кілттер *Дэвид Вагнер* (*David Wagner*) ұсынған *бумеранг әдісі* (ағыл. *boomerang attack*) көмегімен табылған. Бұл класқа жататындықты анықтайтын тест 2^{16} операцияда орындалады және 2^{16} жады ұяшығын талап етеді.

Әлсіз кілттердің осындай үлкен кластарының болуы *IDEA* алгоритмінің тәжірибелік криптографиялық беріктілігіне әсер етпейді, себебі барлық мүмкін кілттер саны 2^{128} тең.

9.2.7. IDEA қолдану

IDEA алгоритмі сауда маркасы болып табылады және ол Австрияда, Францияда, Германияда, Италияда, Нидерландтарда, Испанияда, Швецияда, Швейцарияда, Англияда, АҚШ және Жапонияда патенттелген. Бүгінгі күнге лицензия *MediaCrypt* компаниясында және ол алгоритмді коммерциялық емес қосымшаларда қолдануға мүмкіндік береді. *MediaCrypt* 2005 жылдың мамыр айында ресми түрде жаңа *IDEA NXT* (алғашқы аты

FOX) шифрды көрсетті, *IDEA* мойындалған ауыстырғышы. *IDEA* типті қолдану аймақтары:

- кабельдік теледидар, бейнеконференциялар, қашықтықтан оқыту және басқа үшін аудио- және бейне- деректерді шифрлау;
- коньюктурлы тербелістерді бейнелейтін қаржылық және коммерциялық ақпаратты қорғау;
- модем, роутер немесе *ATM* (*Asynchronous Transfer Mode* – тасымалдаудың асинхронды әдісі) сызығы, *GSM* технология арқылы байланыс сызықтары;
- смарт-карталар;
- *OpenPGP*-де (опционалды) және *PGP v2.0* электронды поштаның конфиденциалды нұсқасының жалпықолжетімді пакеті.

IDEA алгоритмі раундтың инвариантымен жалпы типы бар шифрлайтын *SP*-желілерді қолданады. Раундтың инварианты шифрланатын блоктың үлкен және кіші бөлімдерінің *2* модулі бойынша биттік қосындысы. Бұл алгоритмнің ерекшеліктері:

- саны аз болса да раундтың қиындығы (сегіз толық раунд және соңғы түрлендіру);
- аддитивті және мультипликативті операцияларды қолданумен шифрлау функциясы;
- кілттік элементтерді қолданумен раундтар арасында шифрланатын блокты түзу өзгерту;
- ширек блоктармен (бүтін *16*-биттік) жұмыс істеу;
- биттік алмастыру мен кестелік ауыстырулардың болмауы.

Жоғарыда көрсетілгендерге сәйкес *IDEA* алгоритмі тұтас *"арифметикалық"* алгоритм. *IDEA* алгоритмі жылдам көбейту командасымен *16*-разрядтық процессорларға оңтайландырылған. Шифрлау кілтінен раундтық кілттерді өндірудің өте қарапайым сұлбасы.

IDEA алгоритмі күшті алмастыру эффектісі бар $2^{16}+1$ модулі бойынша көбейту операциясын қолдану арқасында сызықты криптографиялық талдауға қарсы жеткілікті берік. Бұл алгоритмнің криптографиялық талдаудың дифференциалды әдісіне беріктілігі айқын емес.

IDEA ендірілген аппаратты көбейткішін қолданумен бағдарламалық немесе аппараттық жүзеге асыруға бағытталған. Бірақ, бұл жағдайда да $2^{16}+1$ модулі бойынша көбейту бағдарламалық түрде қосуға қарағанда баяуырақ орындалады. Бұл *16*-биттік сандарды көбейтуден басқа қосымша операцияларды

қолдану қажеттілігімен шарттанады. Арнайы шаралардың жоқ кезде бағдарламалық жүзеге асыруда *IDEA* шифрлау үшін машиналық тактар саны кілт және шифрланатын деректер түрінен тәуелді. Сондықтан әр блокты шифрлау уақытын нақты өлшеу кілт туралы қосымша ақпаратты алуға мүмкіндік береді. Бұл жағдай *IDEA* беріктілігін айтарлықтай төмендету мүмкін. Осыған қоса, бұл алгоритм үшін әлсіз кілттер класы бар.

9.3. CAMELLIA ДЕРЕКТЕРДІ ШИФРЛАУДЫҢ БЛОКТЫ АЛГОРИТМІ

9.3.1. Camellia алгоритмін құру тарихы

Camellia блокты симметриялық шифрлау алгоритмі *Mitsubishi Electric* компаниясымен тағы бір белгілі жапондық корпорациямен - *Nippon Telegraph and Telephone (NTT)* ынтымақтастықпен құрылған. Соңғысы үшін бұл халықаралық криптографиялық байқауларға қатысудың бірінші тәжірибесі емес - *AES* байқауына *NTT* корпорациясы *E2* алгоритмін ұсынды, ол байқаудың финалына өтпеді [38].

Camellia құрастырушыларының ішінде *Kanda M. – E2* блокты криптографиялық алгоритмінің құрушысы. *Matsui M. – DES* блокты симметриялық алгоритмінің беріктілігіне арналған бірқатар жұмыстарының авторы және *Aoki K. - Misty1* блокты криптографиялық алгоритмінің авторы. Сондықтан *Camellia, E2* және *Misty1* салыстыру кезінде көп ұқсастықты табуға болады. Құрастырушылар бұл алгоритмдерден ең жақсыларды алды және сол құрылымдық шешімдер негізінде жаңа шифрды құрды.

9.3.2. Camellia алгоритмінің құрылымы

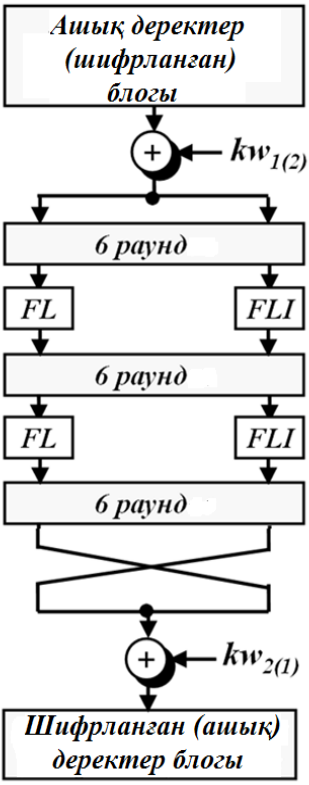
Camellia – бұл *128*-биттік блокты симметриялық алгоритм, шифрлау кілттің ұзындығы *128, 192* және *256* бит. Криптографиялық алгоритм негізінде жылдармен пайдаланған, *DES* криптографиялық алгоритмінде өзін жақсы көрсеткен *Фейстель* желісі жатыр [29]. *Фейстель* желісін қолдану алгоритмді құру және жүзеге асыру кезінде қателерді болдырмауға мүмкіндік береді, бұл оның *Rijndael* алгоритмінен сапалық артықшылығы. Сонымен қатар, осындай құрылымды қолдану шифрлауды және

керішифрлауды биективті жасауға мүмкіндік береді және айырмашылық тек кілттерді кері ретпен беруде болады.

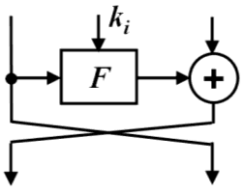
Шифрлау кілтінің көлеміне тәуелді *Camellia* алгоритмінде әртүрлі раунд саны *R* қарастырылған [29]:

- *18 – 128*-биттік шифрлау кілті үшін;
- *24* – шифрлау кілтінің басқа көлемдері үшін.

Шифрлаудың *128*-биттік кілті бар нұсқасы үшін шифрлау алгоритмінің құрылымы 9.11 суретінде келтірілген.



9.11 сурет - *Camellia* шифрлау алгоритмінің құрылымы



9.12 сурет - *Camellia* шифрлаудың бір раунд құрылымы

Шифрлаудың бірінші раунд алдында деректердің кіріс ағартуы орындалады - ашық деректер блогына *xor* операциясымен кеңейтілген кілттің *128*-биттік кесіндісі kw_1 қойылады. Содан кейін *128*-биттік деректер блогы *64*-биттік екі субблокқа бөлінеді, содан кейін субблоктар шифрлау раундтарынан өтеді. Әр алты раунд арасында сол жақтағы субблок *FL* функциясымен өңделеді, ал оң жақтағы субблок - *FLI* функциясымен (бұл функциялар ары қарай толық талқыланады).

Соңғы раундты орындағаннан кейін субблоктар орындарымен ауысады.

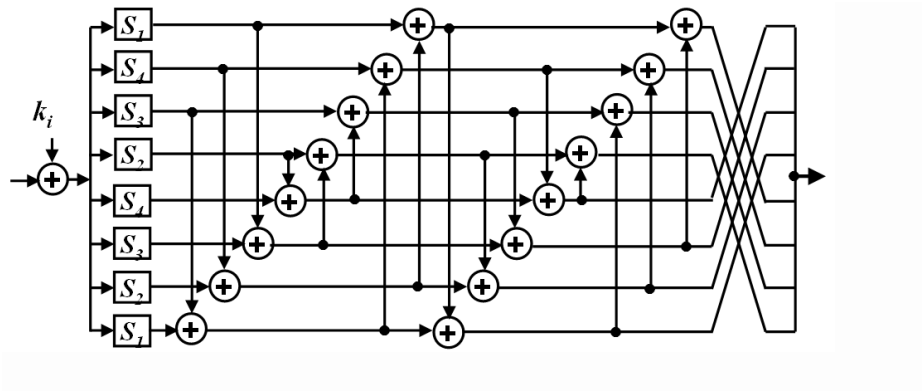
Содан кейін деректердің шығыс ағартуы орындалады; бұл жерде қеңейтілген кілттің тағы бір *128*-биттік кесіндісі kw_2 қолданылады.

Деректерді керішифрлау кезінде қеңейтілген кілттің *128*-биттік кесіндідері кері ретпен қолданылады.

Әр раундта сол жақтағы субблок k_i (*i* – раунд нөмірі) кілтінің *64*-биттік кесіндісін қолданатын *F* функциясымен өңделеді және *xor* операциясымен оң жақ субблокқа қойылады (9.12 суретін қараңыз). Раунд аяғында субблоктар орындарымен ауысады.

Camellia шифрлау алгоритмінің *F* функциясының құрылымы 9.13 суретінде келтірілген.

9.13 суретінде S_i деген ауыстыру блоктары.



9.13 сурет - *Camellia* шифрлау алгоритмінің *F* функциясы

F функцияның әрекеттерін тізбектейміз:

1. Ең алдымен өңделетін деректер субблогына *xor* операциясымен кілт кесіндісін k_i қойю орындалады.

2. Содан кейін алдынғы операцияның *64*-биттік нәтижесі *8* биттен *8* үзіндіге бөлінеді, оның әрқайсысы кестелік ауыстырудан (ауыстырудың *S*-блоктары) өтеді. Бұл операция ары қарай толық көрсетіледі.

3. Содан кейін байттық үзінділердің *xor* операциясы көмегімен бір біріне қоюы келесі ережемен орындалады:

$$y_1 = x_1 \oplus x_3 \oplus x_4 \oplus x_6 \oplus x_7 \oplus x_8;$$

$$y_2 = x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_7 \oplus x_8;$$

$$y_3 = x_1 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_8;$$

$$y_4 = x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7;$$

$$y_5 = x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_8;$$

$$y_6 = x_2 \oplus x_3 \oplus x_5 \oplus x_7 \oplus x_8;$$

$$y_7 = x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_8;$$

$$y_8 = x_1 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7;$$

бұл жерде $x_1 \ldots x_8$ және $y_1 \ldots y_8$ – өңделетін үзінділерінің сәйкес кіріс және шығыс мәндері.

4. Соңында үзінділерді қарапайым алмастыру орындалады: 1...4 байттары 5...8 байттарымен орын ауыстырады (9.13 суретіне қараңыз), содан кейін нәтиже қайтадан 64-биттік деректер субблогына біріктіріледі.

F функция жұмысының бағдарламасының псевдокоды төменде келтірілген.

```
x  = F_IN ^ K_E;
t1 =  x >> 56;
t2 = (x >> 48) & MASK8;
t3 = (x >> 40) & MASK8;
t4 = (x >> 32) & MASK8;
t5 = (x >> 24) & MASK8;
t6 = (x >> 16) & MASK8;
t7 = (x >>  8) & MASK8;
t8 =  x & MASK8;
t1 = S1[t1];
t2 = S2[t2];
t3 = S3[t3];
t4 = S4[t4];
t5 = S2[t5];
t6 = S3[t6];
t7 = S4[t7];
t8 = S1[t8];
y1 = t1 ^ t3 ^ t4 ^ t6 ^ t7 ^ t8;
y2 = t1 ^ t2 ^ t4 ^ t5 ^ t7 ^ t8;
y3 = t1 ^ t2 ^ t3 ^ t5 ^ t6 ^ t8;
y4 = t2 ^ t3 ^ t4 ^ t5 ^ t6 ^ t7;
y5 = t1 ^ t2 ^ t6 ^ t7 ^ t8;
```

$y_6 = t_2 \wedge t_3 \wedge t_5 \wedge t_7 \wedge t_8$;
$y_7 = t_3 \wedge t_4 \wedge t_5 \wedge t_6 \wedge t_8$;
$y_8 = t_1 \wedge t_4 \wedge t_5 \wedge t_6 \wedge t_7$;
$F\_OUT = (y_1 << 56) | (y_2 << 48) | (y_3 << 40) | (y_4 << 32) | (y_5 << 24) | (y_6 << 16) | (y_7 << 8) | y_8$.

Жоғарыда келтірілген бағдарламаның псевдокодында ^ символымен біттік аластамалы *немесе* (*xor*) операциясы белгіленген. S_1, S_2, S_3 және S_4 – ауыстыру блоктары, олар төменде сипатталады.

$MASK_8$, $MASK_{32}$, $MASK_{64}$ және $MASK_{128}$ тұрақтылары 9.8 кестесінде келтірілген (он алтылық мәндер берілген).

9.8 кесте

$MASK_8$, $MASK_{32}$, $MASK_{64}$ және $MASK_{128}$ тұрақтылардың мәндері

Тұрақты	*Мәндер*
$MASK_8$	*0xff*
$MASK_{32}$	*0xffffffff*
$MASK_{64}$	*0xffffffffffffffff*
$MASK_{128}$	*0xffffffffffffffffffffffffffffffff*

9.3.3. Camellia кілтті кеңейту процедурасы

Кілтті кеңейту мақсаты 128-, 192- немесе 256-биттік алғашқы шифрлау кілтінен K қажетті саны бар кеңейтілген кілттер үзінділерін (жоғарыда көрсетілген операциялар үшін) қалыптастыруда. Берілген процедура бірнеше кезеңдерден тұрады.

Ең алдымен *KL* және *KR* айнымалылардың инициализациясы келесідей орындалады:

- 128-биттік кілт үшін *KL* = *K*, *KR* = 0;

- 192-биттік кілт *K* 64 биттен үш үзіндіге бөлінеді, олардың бірінші екеуі *KL* қалыптастырады; үшінші үзінді және оның биттік комплементі *KR* қалыптастырады;

- 256-биттік кілт *K* екіге бөлінеді: *KL* және *KR*.

9.5 мысал. 128 бит ұзындығы бар шифрлау кілті K: 0123456789abcdeffedcba987654321016. *KL* және *KR* айнымалыларды анықтау қажет.

Шешім. Шифрлау кілтінің *K* ұзындығы 128 бит болғандықтан

$$K_L = K = 0123456789abcdeffedcba9876543210_{16},$$

ал

$$K_R = 00000000000000000000000000000000_{16}.$$

9.6 мысал. *256* бит ұзындығы бар шифрлау кілті *K*: *0123456789abcdeffedcba9876543210abcdef0123456789fedcba987654210*$_{16}$. K_L және K_R айнымалыларды анықтау қажет.

Шешім. Шифрлау кілтінің *K* ұзындығы *256* бит болғандықтан, ол екіге бөлінеді (сол жақ $K_Л$ және оң жақ $K_П$): $K_Л$ = *0123456789abcdeffedcba9876543210*$_{16}$; $K_П$ = *abcdef0123456789fedcba9876543210*$_{16}$. K_L айнымалысы келесі мәнді алады:

$$K_L = K_Л = 0123456789abcdeffedcba9876543210_{16},$$

ал K_R айнымалысы:

$$K_R = K_П = abcdef0123456789fedcba9876543210_{16}.$$

9.7 мысал. *192* бит ұзындығы бар шифрлау кілті *K*: *0123456789abcdeffedcba9876543210abcdef0123456789*$_{16}$. K_L және K_R айнымалыларды анықтау қажет.

Шешім. Шифрлау кілтінің *K* ұзындығы *192* бит болғандықтан, ол үшке бөлінеді (сол жақ $K_Л$, ортасы K_C және оң жақ $K_П$): $K_Л$ = = *0123456789abcdef*$_{16}$; K_C = *fedcba9876543210*$_{16}$; $K_П$ = *abcdef0123456789*$_{16}$. K_L айнымалысы келесі мәнді алады:

$$K_L = K_Л \,||\, K_C = 0123456789abcdeffedcba9876543210_{16},$$

ал K_R айнымалысы:

$$K_L = K_П \,||\, \neg K_П = abcdef0123456789543210fedcba9876_{16},$$

бұл жерде ¬ – санның инверсиясы операциясы.

K_L және K_R айнымалыларды анықтағаннан кейін екі *128*-биттік кілттік айнымалылар K_A және K_B есептеледі. Бұл жоғарыда сипатталған *F* функциясын қолданумен орындалады (9.14 сурет):

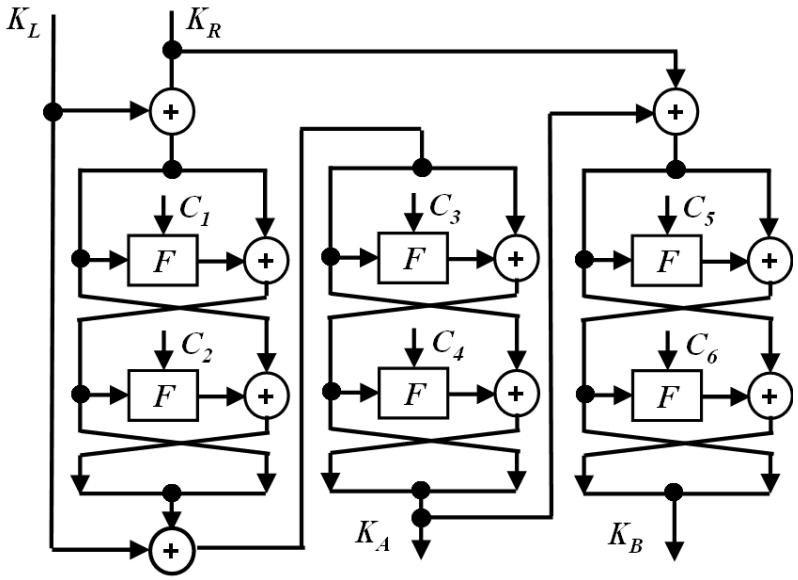
1. $K_L \oplus K_R$ операцияның нәтижесі *F* функциясымен екі рет өңделеді (9.16 сурет), оның кілт үзінділері ретінде C_1 және C_2 тұрақтылары қолданылады.

2. Алдынғы операцияның нәтижесіне *xor* операциясымен K_L қойылады.

3. Кілт үзінділері ретінде C_3 және C_4 тұрақтыларды қолданумен F функциясы тағы екі рет қолданылады. Нәтижесінде K_A айнымалысын алады.

4. Егер *192*- немесе *256*-биттік кілттерді қолданса, сонымен қатар K_B айнымалысын есептеу қажет. Ол үшін K_A *xor* операциясы көмегімен K_R қойылады, нәтижесі C_5 және C_6 тұрақтыларды қолданумен екі рет F функциясымен өңделеді.

$C_1...C_6$ тұрақтылары 9.9 кестесінде келтірілген (он алтылық мәндер көрсетілген).



9.14 сурет - *Camellia* алгоритмінің кілт кеңейту процедурасының кезеңі

9.9 кесте

$C_1...C_6$ тұрақтылардың мәндері

Тұрақты	Мән	Тұрақты	Мән
C_1	$a09e667f3bcc908b_{16}$	C_4	$54ff53a5f1d36f1c_{16}$
C_2	$в67ae8584caa73в2_{16}$	C_5	$10e527fade682d1d_{16}$
C_3	$c6ef372fe94f82be_{16}$	C_6	$b05688c2b3e6c1fd_{16}$

Кілттер кеңейтудің бастапқы кезеңі бағдарламасының псевдокоды төменде келтірілген.

$D_1 = (K_L \wedge K_R) >> 64;$
$D_2 = (K_L \wedge K_R) \,\&\, MASK_{64};$
$D_2 = D_2 \wedge F(D_1, C_1);$
$D_1 = D_1 \wedge F(D_2, C_2);$
$D_1 = D_1 \wedge (K_L >> 64);$
$D_2 = D_2 \wedge (K_L \,\&\, MASK_{64});$
$D_2 = D_2 \wedge F(D_1, C_3);$
$D_1 = D_1 \wedge F(D_2, C_4);$
$K_A = (D_1 << 64) \mid D_2;$
$D_1 = (K_A \wedge K_R) >> 64;$
$D_2 = (K_A \wedge K_R) \,\&\, MASK_{64};$
$D_2 = D_2 \wedge F(D_1, C_5);$
$D_1 = D_1 \wedge F(D_2, C_6);$
$K_B = (D_1 << 64) \mid D_2;$

Келесі кезеңде есептеледі: көмекші *128*-биттік кілттер kw_1 және kw_2; шифрлау кілтінің көлеміне тәуелді көмекші *64*-биттік кілттер ke_1, ..., ke_6 және раундтық кілттер k_1, ..., k_{24}. Кілтті кеңейту бағдарламаның псевдокодтары төменде келтірілген.

128 бит

$kw_1 = K_L;$
$k_1 = (K_A <<< 0) >>> 64;$
$k_2 = (K_A <<< 0) \,\&\, MASK_{64};$
$k_3 = (K_L <<< 15) >> 64;$
$k_4 = (K_L <<< 15) \,\&\, MASK_{64};$
$k_5 = (K_A <<< 15) >> 64;$
$k_6 = (K_A <<< 15) \,\&\, MASK_{64};$
$ke_1 = (K_A <<< 30) >> 64;$
$ke_2 = (K_A <<< 30) \,\&\, MASK_{64};$
$k_7 = (K_L <<< 45) >> 64;$
$k_8 = (K_L <<< 45) \,\&\, MASK_{64};$
$k_9 = (K_A <<< 45) >> 64;$
$k_{10} = (K_L <<< 60) \,\&\, MASK_{64};$
$k_{11} = (K_A <<< 60) >> 64;$
$k_{12} = (K_A <<< 60) \,\&\, MASK_{64};$
$ke_3 = (K_L <<< 77) >> 64;$
$ke_4 = (K_L <<< 77) \,\&\, MASK_{64};$
$k_{13} = (K_L <<< 94) >> 64;$
$k_{14} = (K_L <<< 94) \,\&\, MASK_{64};$
$k_{15} = (K_A <<< 94) >> 64;$

$k_{16} = (K_A <<< 94) \& MASK_{64};$
$k_{17} = (K_L <<< 111) >> 64;$
$k_{18} = (K_L <<< 111) \& MASK_{64};$
$kw_2 = (K_A <<< 111).$

Көмекші *128*-биттік кілттер шифрлаудың бірінші раунд алдында деректерді ағарту үшін қолданады kw_1, ал kw_2 - шифрлаудан кейін. Керішифрлау кезінде олар кері ретпен қолданылады.

64-биттік раундтық кілттер $k_1, ..., k_{18}$ деректерді шифрлау раундтарында қолданылады. Керішифрлау кезінде олар кері ретпен қолданылады.

Шифрлау кезінде *64*-биттік көмекші кілттер ke_1 және ke_3 *FL* функцияларында қолданылады, ал ke_2 және ke_4 – *FLI* функцияларында. Керішифрлау кезінде *64*-биттік көмекші кілттер ke_4 және ke_2 *FL* функцияларында қолданылады, ал ke_3 және ke_1 – *FLI* функцияларында.

192 және 256 бит

$kw_1 = K_L;$
$k_1 = (K_B <<< 0) >> 64;$
$k_2 = (K_B <<< 0) \& MASK_{64};$
$k_3 = (K_R <<< 15) >> 64;$
$k_4 = (K_R <<< 15) \& MASK_{64};$
$k_5 = (K_A <<< 15) >> 64;$
$k_6 = (K_A <<< 15) \& MASK_{64};$
$ke_1 = (K_R <<< 30) >> 64;$
$ke_2 = (K_R <<< 30) \& MASK_{64};$
$k_7 = (K_B <<< 30) >> 64;$
$k_8 = (K_B <<< 30) \& MASK_{64};$
$k_9 = (K_L <<< 45) >> 64;$
$k_{10} = (K_L <<< 45) \& MASK_{64};$
$k_{11} = (K_A <<< 45) >> 64;$
$k_{12} = (K_A <<< 45) \& MASK_{64};$
$ke_3 = (K_L <<< 60) >> 64;$
$ke_4 = (K_L <<< 60) \& MASK_{64};$
$k_{13} = (K_R <<< 60) >> 64;$
$k_{14} = (K_R <<< 60) \& MASK_{64};$
$k_{15} = (K_B <<< 60) >> 64;$
$k_{16} = (K_B <<< 60) \& MASK_{64};$
$k_{17} = (K_L <<< 77) >> 64;$
$k_{18} = (K_L <<< 77) \& MASK_{64};$

$ke_5 = (K_A <<< 77) >> 64;$
$ke_6 = (K_A <<< 77) \& MASK_{64};$
$k_{19} = (K_R <<< 94) >> 64;$
$k_{20} = (K_R <<< 94) \& MASK_{64};$
$k_{21} = (K_A <<< 94) >> 64;$
$k_{22} = (K_A <<< 94) \& MASK_{64};$
$k_{23} = (K_L <<< 111) >> 64;$
$k_{24} = (K_L <<< 111) \& MASK_{64};$
$kw_2 = (K_B <<< 111).$

Көмекші *128*-биттік кілттер шифрлаудың бірінші раунд алдында деректерді ағарту үшін қолданады kw_1, ал kw_2 - шифрлаудан кейін. Керішифрлау кезінде олар кері ретпен қолданылады.

64-биттік раундтық кілттер k_1, ..., k_{18} деректерді шифрлау раундтарында қолданылады. Керішифрлау кезінде олар кері ретпен қолданылады.

Шифрлау кезінде *64*-биттік көмекші кілттер ke_1, ke_3 және ke_5 *FL* функцияларында қолданылады, ал ke_2, ke_4 және ke_6 – *FLI* функцияларында. Керішифрлау кезінде *64*-биттік көмекші кілттер ke_6, ke_4 және ke_2 *FL* функцияларында қолданылады, ал ke_5, ke_3 және ke_1 – *FLI* функцияларында.

9.10 кесте

128-биттік шифрлау кілті үшін K_L, K_R, K_A *және* K_B *айнымалыларын кенейтілген кілт үзінділері ретінде қолдану*

k_{W1}	K_L
k_1 және k_2	K_A оң жақ және сол жақ жартылары
k_3 және k_4	Оң жақ және сол жақ жартылары (K_L<<<*15*)
k_5 және k_6	Оң жақ және сол жақ жартылары (K_A<<<*15*)
ke_1 және ke_2	Оң жақ және сол жақ жартылары (K_A<<<*30*)
k_7 және k_8	Оң жақ және сол жақ жартылары (K_L<<<*45*)
k_9	Сол жақ жартысы (K_A<<<*45*)
k_{10}	Оң жақ жартысы (K_L<<<*60*)
k_{11} және k_{12}	Оң жақ және сол жақ жартылары (K_A<<<*60*)
ke_3 және ke_4	Оң жақ және сол жақ жартылары (K_L<<<*77*)
k_{13} және k_{14}	Оң жақ және сол жақ жартылары (K_L<<<*94*)
k_{15} және k_{16}	Оң жақ және сол жақ жартылары (K_A<<<*94*)
k_{17} және k_{18}	Оң жақ және сол жақ жартылары (K_L<<<*111*)
k_{W2}	K_A<<<*111*

Шифрлау процессінде K_L, K_R, K_A және K_B айнымалылар кеңейтілген кілт үзінділері ретінде ары қарай келтірілген кестелер бойынша (үзінділерді қолдану ретімен) қолданылады.

128-биттік кілт үшін – 9.10 кестесін қараңыз, ал кілттің басқа көлемдері үшін – 9.11 кестені.

9.11 кесте

192- және 256-биттік шифрлау кілтері үшін K_L, K_R, K_A *және* K_B *айнымалыларын кенейтілген кілт үзінділері ретінде қолдану*

k_{W1}	K_L
k_1 және k_2	Оң жақ және сол жақ жартылары K_B
k_3 және k_4	Оң жақ және сол жақ жартылары (K_R<<<*15*)
k_5 және k_6	Оң жақ және сол жақ жартылары (K_A<<<*15*)
ke_1 және ke_2	Оң жақ және сол жақ жартылары (K_R<<<*30*)
k_7 және k_8	Оң жақ және сол жақ жартылары (K_B<<<*30*)
k_9 және k_{10}	Оң жақ және сол жақ жартылары (K_L<<<*45*)
k_{11} және k_{12}	Оң жақ және сол жақ жартылары (K_A<<<*45*)
ke_3 және ke_4	Оң жақ және сол жақ жартылары (K_L<<<*60*)
k_{13} және k_{14}	Оң жақ және сол жақ жартылары (K_R<<<*60*)
k_{15} және k_{16}	Оң жақ және сол жақ жартылары (K_B<<<*60*)
k_{17} және k_{18}	Оң жақ және сол жақ жартылары (K_L<<<*77*)
ke_5 және ke_6	Оң жақ және сол жақ жартылары (K_A<<<*77*)
k_{19} және k_{20}	Оң жақ және сол жақ жартылары (K_R<<<*94*)
k_{21} және k_{22}	Оң жақ және сол жақ жартылары (K_A<<<*94*)
k_{23} және k_{24}	Оң жақ және сол жақ жартылары (K_L<<<*111*)
k_{W2}	K_A<<<*111*

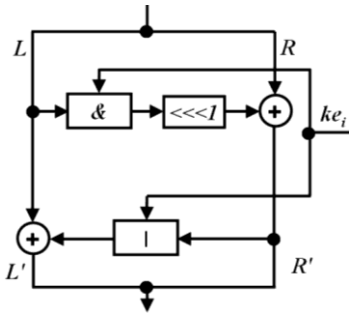
9.3.4. Camellia деректерді шифрлау

Camellia алгоритмінің FL функциясы

Әр *6* раундтан кейін орындалатын *FL* функциясы келесі әрекеттерді орындайды (9.15 сурет):

$$R' = R \oplus ((L \,\&\, ke_L) <<< 1);\ L' = L \oplus (R' \mid ke_R),$$

бұл жерде L және R – өңделетін субблоктың кіріс мәнінің сол жақ және оң жақ бөлімдері (әрқайсысының ұзындығы *32* бит); L' және R' – шығыс мәнінің сол жақ және оң жақ бөлімдері (әрқайсысының ұзындығы *32* бит);



9.15 сурет - *Camellia* алгоритмінің *FL* функциясы

ke_L және ke_R – FL функциясы үшін *64*-разрядты кілттің ke_i сол жақ және оң жақ бөлімдері (раунд санына тәуелді *i = 1, 3* және *5*); *<<< 1* – сол жаққа бір битке циклды жылжыту операциясы;& және | – "*және*" (*and* – көбейту) және "*немесе*" (*or* – қосу) биттік логикалық операциялар.

FL функциясының жұмыс бағдарламасының псевдокоды төменде көрсетілген.

var x_1, x_2 as 32-bit unsigned integer;
var k_1, k_2 as 32-bit unsigned integer;
x_1 = FL_IN >> 32;
x_2 = FL_IN & $MASK_{32}$;
k_1 = ke_L;
k_2 = ke_R;
x_2 = x_2 ^ ((x_1 & k_1) <<< 1);
x_1 = x_1 ^ (x_2 | k_2);
FL_OUT = (x_1 << 32) | x_2.

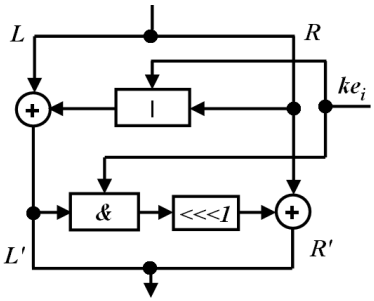
Camellia алгоритмінің FLI функциясы

FLI функциясы келесідей анықталған (9.16 сурет):

$$L' = L \oplus (R' | ke_L);\ R' = R \oplus ((L \& ke_R) <<< 1),$$

бұл жерде L және R – өңделетін субблоктың кіріс мәнінің сол жақ және оң жақ бөлімдері (әрқайсысының ұзындығы *32* бит); L' және R' – шығыс мәнінің сол жақ және оң жақ бөлімдері (әрқайсысының ұзындығы *32* бит); ke_L және ke_R – FLI функциясы үшін *64*-разрядты кілттің ke_i сол жақ және оң жақ бөлімдері (раунд санына тәуелді *i = 2, 4* және *6*).

FLI функциясының жұмыс бағдарламасының псевдокоды төменде көрсетілген.



var y_1, y_2 as 32-bit unsigned integer;
var k_1, k_2 as 32-bit unsigned integer;
y_1 = FLI_IN >> 32;
y_2 = FLI_IN & $MASK_{32}$;
k_1 = ke_L;
k_2 = ke_R;
y_1 = y_1 ^ (y_2 | k_2);
y_2 = y_2 ^ ((y_1 & k_1) <<< 1);
FLI_OUT = (y_1 << 32) | y_2.

9.16 сурет - *Camellia* алгоритмінің *FLI* функциясы

Camellia алгоритмінің ауыстыру блоктары

Ауыстырудың *S*-блогы ретінде (9.13 сурет) *Галуа* ($GF(2^8)$) өрісінде қайтымды элементті есептеу функциясына аффинды-эквивалентті функциялары алынған. $GF(2^8)$ функциялардың максималды дифференциалды және сызықты ықтималдықтары үшін тексерілген минимум – 2^{-6} екені жақсы белгілі [29]. Ауыстырудың *S*-блогы үшін авторлар функцияның осы типын таңдады. Осыған қоса, ауыстырудың *S*-блогынан әр шығыс биттың буль пониномдарының жоғары деңгейі жоғары дәрежесі бар дифференциалдармен шабуылды орындауды қиындатады. Осы *S*-блоктарының әр шығыс битының буль полиномдарының дәрежесі шынайы *7* тең деп авторлар айтады. Интерполяциялық шабуылды қиындату үшін *S*-блоктардың кірісінде және шығысында қосымша тағы екі аффинды функцияны қолданады, бұл $GF(2^8)$ ішінде *S*-блоктарының өзін табуды қиындатады.

Алгоритмде төрт ауыстыру блогы қолданылады: $S_1 \ldots S_4$. Бұл блоктардың бір бірінен шамалы айырмашылығы бар, олар негізгі *S*-блоктан математикалық түрлендіру арқылы алынады, бұл Камелия крипторгафиялық алгоритмін смарт-карталарда жүзеге асыру кезінде жадыны үнемдейді. Төрт *S*-блокты қолдану қысқартылған дифференциалды криптографиялық талдауға қарсы қауіпсіздікті жоғарлатады.

Ауыстыру блоктар $x_1 \ldots x_8$ байттық үзінділерін келесі ретпен өңдейді:

- S_1 ауыстыру блогы x_1 және x_8 үзінділерін өңдейді;

- S_2 ауыстыру блогы x_4 и x_7 өңдейді;
- S_3 ауыстыру блогы x_3 и x_6 өңдейді;
- S_4 ауыстыру блогы x_2 и x_5 өңдейді.

Camellia алгоритмінде ауыстыру блоктары (шифрлаторды жүзеге асыруға қойылатын талаптарына тәуелді) тікелей кесте көмегімен жүзеге асырылу мүмкін, сонымен қатар келесі есептер арқылы жүзеге асырылу мүмкін:

- ауыстыру блогының S_1 кестесі:

$$y = h(g(f(c5 \oplus x))) \oplus 6e, \tag{9.12}$$

бұл жерде *c5* және *6e* – он алтылық тұрақтылар; *x және y* – кіріс және шығыс мәндері; *f, g* және *h* ары қарай жазбаланады;

- ауыстыру блогының S_2 кестесі:

$$y = S_1(x) <<< 1; \tag{9.13}$$

- ауыстыру блогының S_3 кестесі:

$$y = S_1(x) >>> 1; \tag{9.14}$$

бұл жерде >>>1 – бір битке оң жаққа циклды жылжыту операциясы;

- ауыстыру блогының S_4 кестесі:

$$y = S_1(x <<< 1). \tag{9.15}$$

Функция *f* деректер байтын келесідей түрлендіреді:

$$b_1 = a_6 \oplus a_2; \quad b_2 = a_7 \oplus a_1; \quad b_3 = a_8 \oplus a_5 \oplus a_3; \quad b_4 = a_8 \oplus a_3;$$

$$b_5 = a_7 \oplus a_4; \quad b_6 = a_5 \oplus a_2; \quad b_7 = a_8 \oplus a_1; \quad b_8 = a_6 \oplus a_4,$$

бұл жерде a_1 ... a_8 және b_1 ... b_8 – кіріс және шығыс мәндерінің биттері.

Функция *g* шығыс мәннің биттерін келесідей анықтайды:

$$(b_8 + b_7\alpha + b_6\alpha^2 + b_5\alpha^3) + (b_4 + b_3\alpha + b_2\alpha^2 + b_1\alpha^3)\beta =$$

$$= \frac{1}{(a_8 + a_7\alpha + a_6\alpha^2 + a_5\alpha^3) + (a_4 + a_3\alpha + a_2\alpha^2 + a_1\alpha^3)\beta}.$$

Ауыстыру блоктары S_1, S_2, S_3 және S_4 $GF(2^8)$ ішінде инверсия функциясына аффинды эквивалентті. Есептеулер шегі бар $GF(2^8)$ өрісінде орындалады, 0 қайтымды мәні 0 болады, β – келесі шартқа қанағаттандырылатын шегі бар $GF(2^8)$ өрістің элементі:

$$\beta^8 + \beta^6 + \beta^5 + \beta^3 + 1 = 0,$$

ал

$$\alpha = \beta^{238} = \beta^6 + \beta^5 + \beta^3 + \beta^2$$

$\alpha^4 + \alpha + 1 = 0$ теңдікке қанағаттандырылатын элемент $GF(2^4)$ өрісіндегі элемент.

Функция h келесі түрлендірулерді орындайды:

$$b_1 = a_5 \oplus a_6 \oplus a_2; \quad b_2 = a_6 \oplus a_2; \quad b_3 = a_7 \oplus a_4; \quad b_4 = a_8 \oplus a_2;$$

$$b_5 = a_7 \oplus a_3; \quad b_6 = a_8 \oplus a_1; \quad b_7 = a_5 \oplus a_1; \quad b_8 = a_6 \oplus a_1.$$

S_1 ауыстыру блокгының мәндерін x 0 ден 255 дейін болған кезде өрнекпен (9.12) есептеп, 9.12 кестесінде келтірілген 256 мәнді алуға болады [29].

9.12 кестені келесідей түсінеді: 00_{16} кіріс мәні 70_{16} ауыстырылады, $01_{16} - e6_{16}$ және ары қарай шығыс мәні ff_{16} болғанша, ол $9e_6$ ауыстырылады.

S_2 ауыстыру блогының мәндерін (9.13) өрнегімен, S_3 (9.14) өрнегімен және S_4 (9.15) өрнегімен x мәні 00_{16} ден ff_{16} дейін болғанда есептеп, ауыстыру кестелерін алуға болады (9.13…9.15 кестелері).

9.12 кесте

Ауыстыру блогы S_1 мәндері

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	70	82	2c	ec	b3	27	c0	e5	e4	85	57	35	ea	0c	ae	41
1	21	ef	6b	93	45	19	a5	21	ed	0e	4f	4e	1d	65	92	bd
2	86	b8	af	8f	7c	eb	1f	ce	3e	30	dc	5f	5e	c5	0b	1a
3	a6	e1	39	ca	d5	47	5d	3d	d9	01	5a	d6	51	56	6c	4d
4	8b	0d	9a	66	fb	cc	b0	2d	74	12	2b	20	f0	b1	84	99
5	d9	4c	cb	c2	34	7e	76	05	6d	b7	a9	31	d1	17	04	d7
6	14	58	3a	61	de	1b	11	1c	32	0f	9c	16	53	18	f2	22
7	fe	44	cf	b2	c3	b5	7a	91	24	08	e8	a8	60	fc	69	50
8	aa	d0	a0	7d	a1	89	62	97	54	5b	1e	95	e0	ff	64	d2
9	10	c4	00	48	a3	f7	75	db	8a	03	e6	da	09	3f	dd	94
a	87	5c	83	02	cd	4a	90	33	73	67	f6	f3	9d	7f	bf	e2
b	52	9b	d8	26	c8	37	c6	3b	81	96	6f	4b	13	be	63	2e
c	e9	79	a7	8c	9f	6e	bc	8e	29	f5	f9	b6	2f	fd	b4	59
d	78	98	06	6a	e7	46	71	ba	d4	25	ab	42	88	a2	8d	fa
e	72	07	b9	55	f8	ee	ac	0a	36	49	2a	68	3c	38	f1	a4
f	40	28	d3	7b	bb	c9	43	c1	15	e3	ad	f4	77	c7	80	9e

9.13 кесте

Ауыстыру блогы S_2 мәндері

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	e0	05	58	d9	67	4e	81	cb	c9	0b	ae	6a	d5	18	5d	82
1	46	df	d6	27	8a	32	4b	42	db	1c	9e	9c	3a	ca	25	7b
2	0d	71	5f	1f	f8	d7	3e	9d	7c	60	b9	be	bc	8b	16	34
3	4d	c3	72	95	ab	8e	ba	7a	b3	02	b4	ad	a2	ac	d8	9a
4	17	1a	35	cc	f7	99	61	5a	e8	24	56	40	e1	63	09	33
5	bf	98	97	85	68	fc	ec	0a	da	6f	53	62	a3	2e	08	af
6	28	b0	74	c2	bd	36	22	38	64	1e	39	2c	a6	30	e5	44
7	fd	88	9f	65	87	6b	f4	23	48	10	d1	51	c0	f9	d2	a0
8	55	a1	41	fa	43	13	c4	2f	a8	b6	3c	2b	c1	ff	c8	a5
9	20	89	00	90	47	ef	ea	b7	15	06	cd	b5	12	7e	bb	29

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
a	0f	b8	07	04	9b	94	21	66	e6	ce	ed	e7	3b	fe	7f	c5
b	a4	37	b1	4c	91	6e	8d	76	03	2d	de	96	26	7d	c6	5c
c	d3	f2	4f	19	3f	dc	79	1d	52	eb	f3	6d	5e	fb	69	b2
d	f0	31	0c	d4	cf	8c	e2	75	a9	4	57	84	11	45	1b	f5
e	e4	0e	73	aa	f1	dd	59	14	6c	92	54	d0	78	70	e3	49
f	80	50	a7	f6	77	93	86	83	2a	c7	5b	e9	ee	8f	01	3d

9.14 кесте

Ауыстыру блогы S_3 мәндері

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	38	41	16	76	d9	93	60	f2	72	c2	ab	9a	75	06	57	a0
1	91	f7	b5	c9	a2	8c	d2	90	f6	07	a7	27	8e	b2	49	de
2	43	5c	d7	c7	3e	f5	8f	67	1f	18	6e	af	2f	e2	85	0d
3	53	f0	9c	65	ea	a3	ae	9e	ec	80	2d	6b	a8	2b	36	a6
4	c5	86	4d	33	fd	66	58	96	3a	09	95	10	78	d8	42	cc
5	ef	26	e5	61	1a	3f	3b	82	b6	db	d4	98	e8	8b	02	eb
6	0a	2c	1d	b0	6f	8d	88	0e	19	87	4e	0b	a9	0c	79	11
7	7f	22	e7	59	e1	da	3d	c8	12	04	74	54	30	7e	b4	28
8	55	68	50	be	d0	c4	31	cb	2a	ad	0f	ca	70	ff	32	69
9	08	62	00	24	d1	fb	ba	ed	45	81	73	6d	84	9f	ee	4a
a	c3	2e	c1	01	e6	25	48	99	b9	b3	7b	f9	ce	bf	df	71
b	29	cd	6c	13	64	9b	63	9d	c0	4b	b7	a5	89	5f	b1	17
c	f4	bc	d3	46	cf	37	5e	47	94	fa	fc	5b	97	fe	5a	ac
d	3c	4c	03	35	f3	23	b8	5d	6a	92	d5	21	44	51	c6	7d
e	39	83	dc	aa	7c	77	56	05	1b	a4	15	34	1e	1c	f8	52
f	20	14	e9	bd	dd	e4	a1	e0	8a	f1	d6	7a	bb	e3	40	4f

9.15 кесте

Ауыстыру блогы S_4 мәндері

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	70	2c	b3	c0	e4	57	ea	ae	23	6b	45	a5	ed	4f	1d	92
1	86	af	7c	1f	3e	dc	5e	0b	a6	39	d5	5d	d9	5a	51	6c

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
2	8b	9a	fb	b0	74	2b	f0	84	df	cb	34	76	6d	a9	d1	04
3	14	3a	de	11	32	9c	53	f2	fe	cf	c3	7a	24	e8	60	69
4	aa	a0	a1	62	54	1e	e0	64	10	00	a3	75	8a	e6	09	dd
5	87	83	cd	90	73	f6	9d	bf	52	d8	c8	c6	81	6f	13	63
6	e9	a7	9f	bc	29	f9	2f	b4	78	06	e7	71	d4	ab	88	8d
7	72	b9	f8	ac	36	2a	3c	f1	40	d3	bb	43	15	ad	77	80
8	82	ec	27	e5	85	35	0c	41	ef	93	19	21	0e	4e	65	bd
9	b8	8f	eb	ce	30	5f	c5	1a	e1	ca	47	3d	01	d6	56	4d
a	0d	66	cc	2d	12	20	b1	99	4c	c2	7e	05	b7	31	17	d7
b	58	61	1b	1c	0f	16	18	22	44	b2	b5	91	08	a8	fc	50
c	d0	7d	89	97	5b	95	ff	d2	c4	48	f7	db	03	da	3f	94
d	5c	02	4a	33	67	f3	7f	e2	9b	26	37	3b	96	4b	be	2e
e	79	8c	6e	8e	f5	b6	fd	59	98	6a	46	ba	25	42	a2	fa
f	07	55	ee	0a	49	68	38	a4	28	7b	c9	c1	e3	f4	c7	9e

Шифрлау *Фейстель* сұлбасымен *128*-биттік кілт үшін *18* раундпен, *192*- және *256*-биттік кілттер үшін *24* раундпен орындалады. Әр *6* раунд сайын *FL* және *FLI* функциялары қолданылады.

128-, *192*- және *256*-биттік кілттер үшін деректерді шифрлау бағдарламалардың псевдокодтары төменде келтірілген.

128 бит

$D_1 = M >> 64;$ // *Шифрланатың хабар екіге бөлінеді*
$D_2 = M \& MASK_{64};$
$D_1 = D_1 \wedge kw_1;$ // *Алдынғы ағарту*
$D_2 = D_2 \wedge kw_2;$
$D_2 = D_2 \wedge F(D_1, k_1);$
$D_1 = D_1 \wedge F(D_2, k_2);$
$D_2 = D_2 \wedge F(D_1, k_3);$
$D_1 = D_1 \wedge F(D_2, k_4);$
$D_2 = D_2 \wedge F(D_1, k_5);$
$D_1 = D_1 \wedge F(D_2, k_6);$
$D_1 = FL(D_1, ke_1);$ // *FL*
$D_2 = FLI(D_2, ke_2);$ // *FLI*
$D_2 = D_2 \wedge F(D_1, k_7);$

$D_1 = D_1 \wedge F(D_2, k_8);$
$D_2 = D_2 \wedge F(D_1, k_9);$
$D_1 = D_1 \wedge F(D_2, k_{10});$
$D_2 = D_2 \wedge F(D_1, k_{11});$
$D_1 = D_1 \wedge F(D_2, k_{12});$
$D_1 = FL(D_1, ke3);$ // *FL*
$D_2 = FLI(D_2, ke4);$ // *FLI*
$D_2 = D_2 \wedge F(D_1, k_{13});$
$D_1 = D_1 \wedge F(D_2, k_{14});$
$D_2 = D_2 \wedge F(D_1, k_{15});$
$D_1 = D_1 \wedge F(D_2, k_{16});$
$D_2 = D_2 \wedge F(D_1, k_{17});$
$D_1 = D_1 \wedge F(D_2, k_{18});$
$D_2 = D_2 \wedge kw3;$ // *Соңғы ағарту*
$D_1 = D_1 \wedge kw4;$
$C = (D_2 << 64) \mid D_1.$

192 және 256 бит

$D_1 = M >> 64;$ // *Шифрланатың хабар екіге бөлінеді*
$D_2 = M \ \& \ MASK_{64};$
$D_1 = D_1 \wedge kw_1;$ // *Алдынғы ағарту*
$D_2 = D_2 \wedge kw_2;$
$D_2 = D_2 \wedge F(D_1, k_1);$
$D_1 = D_1 \wedge F(D_2, k_2);$
$D_2 = D_2 \wedge F(D_1, k_3);$
$D_1 = D_1 \wedge F(D_2, k_4);$
$D_2 = D_2 \wedge F(D_1, k_5);$
$D_1 = D_1 \wedge F(D_2, k_6);$
$D_1 = FL(D_1, ke_1);$ // *FL*
$D_2 = FLI(D_2, ke_2);$ // *FLI*
$D_2 = D_2 \wedge F(D_1, k_7);$
$D_1 = D_1 \wedge F(D_2, k_8);$
$D_2 = D_2 \wedge F(D_1, k_9);$
$D_1 = D_1 \wedge F(D_2, k_{10});$
$D_2 = D_2 \wedge F(D_1, k_{11});$
$D_1 = D_1 \wedge F(D_2, k_{12});$
$D_1 = FL(D_1, ke3);$ // *FL*
$D_2 = FLI(D_2, ke4);$ // *FLI*
$D_2 = D_2 \wedge F(D_1, k_{13});$
$D_1 = D_1 \wedge F(D_2, k_{14});$
$D_2 = D_2 \wedge F(D_1, k_{15});$
$D_1 = D_1 \wedge F(D_2, k_{16});$

$D_2 = D_2 \wedge F(D_1, k_{17});$
$D_1 = D_1 \wedge F(D_2, k_{18});$
$D_1 = FL(D_1, ke5);$ // *FL*
$D_2 = FLI(D_2, ke6);$ // *FLI*
$D_2 = D_2 \wedge F(D_1, k_{19});$
$D_1 = D_1 \wedge F(D_2, k_{20});$
$D_2 = D_2 \wedge F(D_1, k_{21});$
$D_1 = D_1 \wedge F(D_2, k_{22});$
$D_2 = D_2 \wedge F(D_1, k_{23});$
$D_1 = D_1 \wedge F(D_2, k_{24});$
$D_2 = D_2 \wedge kw3;$ // *Соңғы ағарту*
$D_1 = D_1 \wedge kw4;$
$C = (D_2 << 64) | D_1.$

9.8 мысал. *128* бит ұзындығы бар шифрлау кілті *K*: $0123456789abcdeffedcba9876543210_{16}$. Шифрланатын хабар: $M = 0123456789abcdeffedcba9876543210_{16}$. Раундтық кілттерді және шифрланған хабарды анықтау.

Шешім: Раундтық кілттер:

$k_1 = ae71c3d55ba6bf1d_{16}$; $k_2 = 169240a795f89256_{16}$;
$k_3 = a2b3c4d5e6f7ff6e_{16}$; $k_4 = 5d4c3b2a19080091_{16}$;
$k_5 = e1eaadd35f8e8b49_{16}$; $k_6 = 2053cafc492b5738_{16}$;
$k_7 = 79bdffdb97530eca_{16}$; $k_8 = 8642002468acf135_{16}$;
$k_9 = d7e3a2d24814f2bf_{16}$; $k_{10} = 00123456789abcde_{16}$;
$k_{11} = d169240a795f83df5_{16}$; $k_{12} = 6ae71c3d55ba6bf1_{16}$;
$k_{13} = 1d950c840048d159_{16}$; $k_{14} = e26af37bffb72ea6_{16}$;
$k_{15} = e57e2495ab9c70f5_{16}$; $k_{16} = 56e9afc745a49029_{16}$;
$kw_1 = 0123456789abcdef_{16}$; $kw_2 = fedcba9876543210_{16}$;
$kw_3 = 492b5738e1eaadd3_{16}$; $kw_4 = 5f8e8b492053cafc_{16}$;
$ke_1 = 56e9afc745a49029_{16}$; $ke_2 = e57e2495ab9c70f5_{16}$;
$ke_3 = 97530eca86420024_{16}$; $ke_4 = 68acf13579bdffdb_{16}$.

Шифрланған хабар: $C = 676731385496697308570656 48\ eabe43_{16}$.

9.3.5. Camellia деректерді керішифрлау

Camellia алгоритмімен деректерді керішифрлау шифрлаумен бірдей орындалады, бірақ кеңейтілген кілт кесінділерін кері ретпен қолданумен, яғни:

- kw_2 ішкі кілті деректерді кіріс ағартуда қолданады, және керісінше kw_1 ішкі кілті деректерді соңғы ағартуда қолданылады;

- шифрлау раундтарында кілттер k_i кері ретпен қолданылады: k_N - k_1;

- FL және FLI операцияларында кеңейтілген кілттің жұп үзінділері сол жақ субблокты өңдеуде қолданылады, ал тақ үзінділер - оң жақ; екеуіде шифрлауға қарағанда кері ретпен алынады;

Шифрлау және керішифрлау кезінде раундтық және көмекші кілттерді қолданудағы айырмашылықтар 9.16 кестеде келтірілген.

9.16 кесте

Шифрлау және керішифрлау кезінде раундтық және көмекші кілттерді қолданудағы айырмашылықтар

Шифрлау кілтінің көлемі			
128 бит		*192 немесе 256 бит*	
Шифрлау	*Керішифрлау*	*Шифрлау*	*Керішифрлау*
kw_1	kw_2	kw_1	kw_2
kw_2	kw_1	kw_2	kw_1
k_1	k_{18}	k_1	k_{24}
k_2	k_{17}	k_2	k_{23}
k_3	k_{16}	k_3	k_{22}
k_4	k_{15}	k_4	k_{21}
k_5	k_{14}	k_5	k_{20}
k_6	k_{13}	k_6	k_{19}
k_7	k_{12}	k_7	k_{18}
k_8	k_{11}	k_8	k_{17}
k_9	k_{10}	k_9	k_{16}
k_{10}	k_9	k_{10}	k_{15}
k_{11}	k_8	k_{11}	k_{14}
k_{12}	k_7	k_{12}	k_{13}
k_{13}	k_6	k_{13}	k_{12}
k_{14}	k_5	k_{14}	k_{11}
k_{15}	k_4	k_{15}	k_{10}
k_{16}	k_3	k_{16}	k_9
k_{17}	k_2	k_{17}	k_8
k_{18}	k_1	k_{18}	k_7
		k_{19}	k_6
		k_{20}	k_5

		k_{21}	k_4
		k_{22}	k_3
		k_{23}	k_2
		k_{24}	k_1
ke_1	ke_3	ke_1	ke_6
ke_2	ke_4	ke_2	ke_5
		ke_3	Ke_4

9.3.6. Camellia қауіпсіздігі

Camellia алгоримінің бастапқы талдауы оның құрушыларымен орындалған. Алгоритмнің сызықты және дифференциалды крито-талдауына беріктілік [20] көрсетілген, сонымен қатар қысқартылған және мүмкінді емес дифференциалдарды қолдануға, бумеранг әдісіне, интерполяция әдісіне, жылжу шабуылдарына және бір қатар басқа шабуылдарға.

Camellia алгоритмі туралы белгілі сарапшылардың пікірлері оңтайлы болды. Әртүрлі жұмыстарда *Camellia* жоғары криптографиялық беріктілігі белгіленеді. *Ларс Кнудсен* келесіні бекітеді:

- *Camellia* кез келген тәжірибелік криптографиялық шабуылдар тек криптоталдау аймағындағы түбегейлі алға жылжудан кейін мүмкін болады;

- *Camellia* – ең берік заманауи шифрлау алгоритмдерінің бірі.

Camellia алгоритмінің ресурскөлемдігін және жылдамдығын бағалауда сарапшылар бір пікірде болмады:

- *Camellia* жылдамдықта *Rijndael* алгоритмінен жеңілетіні белгіленеді;

- *Camellia* жедел және энерготәуелді жадыға жеткілікті жоғары талаптарды қояды.

Көп зерттеулер *Camellia* алгоритмінің әлсізденген нұсқасына жүргізілді - раундтар саны аз, сонымен қатар кіріс/шығыс ағартуларсыз және FL/FLI *операцияларысыз*. Келесі жұмыстарды айтып кету қажет:

- *Camellia 9*-раундтық нұсқасына шабуыл жазбаланған, ол 2^{105} таңдалған ашық мәтіндер болса 2^{64} шифрлау операцияларымен орындалады; сол жерде алгоритмді ашу үшін сызықты криптографиялық талдау жарамды емес екені белгіленеді;

- *9*-раундты қысқартылған дифференциал табылған, оның көмегімен *Camellia 11*-раундтық нұсқасына шабуыл мүмкін;

- *Camellia 9*-раундтық нұсқасына коллизияларды іздеу әдісі ұсынылған, ол үшін $2^{113,6}$ таңдалған ашық мәтіндер және 2^{121} шифрлау операциялары (*128*-биттік кілт) қажет немесе 2^{13} таңдалған ашық мәтіндер және $2^{175,6}$ шифрлау операциялары (*192*-биттік кілт) қажет; сол жерде *256*-биттік кілт бар *10*-раундтық нұсқасына шабуыл ұсынылды, ол үшін 2^{14} таңдалған ашық мәтін және $2^{239,9}$ операция қажет;

- мүмкін емес дифференциалдар әдісі қолданылған, *8*-раундтық мүмкін емес дифференциал табылған, оның көмегімен *12*-раундтық нұсқаға шабуыл жасалды; шабуыл үшін 2^{120} ашық мәтін және 2^{181} операция қажет.

Camellia алғашқы нұсқасына жақын нұсқаларды талдайтын жұмыстар сирек кездеседі. *Camellia 9*-раундтық алгоритмін (алтыншы раундтан кейін FL/FLI *операцияларымен бірге, бірақ алғашқы және соңғы ағартусыз*) ашатын *Square*-шабуылы қолданылған жұмыс есте қалады, ол үшін $2^{60,5}$ашық мәтін және $2^{202,2}$ шифрлау операция қажет.

Camellia алгоритмі жанама арналармен ақпаратты ағуды қолданатын шабуылдар көз қарасынан да зерттелді. Бұнымен байланысты келесі жұмыстар белгілі:

- *Camellia* пайдаланатын құаттылық бойынша шабуылдардан қорғалғандық көз қарасынан *NESSIE* байқауының екінші раундында ұсынылған алгоритмдер ішінде ең жақсылардың бірі деп жұмыстарының бірінің авторлары айтады;

- *Camellia* алгоритмінің кілт кеңейту процедураласының ерекшеліктерін қолданатын пайдаланатын құаттылық бойынша шабуыл ұсынылды.

Camellia – NESSIE байқауының жеңімпаз алгоритмдерінің бірі. *NESSIE* байқауы кезінде өткізілген зерттеулер барысында берілген алгоритмнің криптографиялық беріктілігімен байланысты ешқандай проблемалар анықталған жоқ. Кәзіргі уақытта *Camellia* алгоритмінде ауыр осалдықтар табылған жоқ, бірақ бұл алгоритмнің белсенді талдауы ары қарай жалғасатыны белгілі.

Бақылау сұрақтары және тапсырмалар

1. *IDEA* өңделетін блок көлемі, шифр кілтінің көлемі және раунд кілттерінің көлемі қандай?

2. *IDEA* раундтар саны неше?

3. *IDEA* раундтарында қандай қарапайым операциялар қолданылады?

4. *IDEA* раундтарында деректерді өзгерту кезеңінде қандай қарапайым операциялар қолданылады?

5. *IDEA* раундтарында деректерді шифрлау кезеңінде қандай қарапайым операциялар қолданылады?

6. *IDEA* деректерді шифрлау кезінде раундтық кілттерді қалыптастыру алгоритмін түсіндіріңіз.

7. *IDEA* деректерді керішифрлау кезінде раундтық кілттерді қалыптастыру алгоритмін түсіндіріңіз.

8. *IDEA* алгоритмінің алғашқы (сеанстық) кілті - 128 бит ұзындығы бар тізбек, ол тең $K = 3f424cdc105ca00d7b3dbe8c96a2978e_{16}$. Шифрлаудың екінші раунды үшін раундтық кілттерді қалыптастыру.

9. *IDEA* алгоритмінің керішифрлаудың раундтық кілттерін $k_2^{(i)}$ ($i = 1...9$) анықтау, егер шифрлаудың раундтық кілттері тең: $k_2^{(1)} = 4cdc_{16}$, $k_2^{(2)} = 97be_{16}$, $k_2^{(3)} = 452f_{16}$, $k_2^{(4)} = 5a8a_{16}$, $k_2^{(5)} = 64b5_{16}$, $k_2^{(6)} = 6bd9_{16}$, $k_2^{(7)} = 00d7_{16}$, $k_2^{(8)} = 9401_{16}$, $k_2^{(9)} = 1728_{16}$, $k_3^{(1)} = 105c_{16}$, $k_3^{(2)} = b820_{16}$, $k_3^{(3)} = 1c7e_{16}$, $k_3^{(4)} = 5e38_{16}$, $k_3^{(5)} = 14bc_{16}$, $k_3^{(6)} = 6a29_{16}$, $k_3^{(7)} = b3db_{16}$, $k_3^{(8)} = af67_{16}$, $k_3^{(9)} = 035e_{16}$.

10. *IDEA* алгоритмінің керішифрлаудың раундтық кілттерін $k_3^{(i)}$ ($i = 1...9$) анықтау, егер шифрлаудың раундтық кілттері тең: $k_2^{(1)} = 4cdc_{16}$, $k_2^{(2)} = 97be_{16}$, $k_2^{(3)} = 452f_{16}$, $k_2^{(4)} = 5a8a_{16}$, $k_2^{(5)} = 64b5_{16}$, $k_2^{(6)} = 6bd9_{16}$, $k_2^{(7)} = 00d7_{16}$, $k_2^{(8)} = 9401_{16}$, $k_2^{(9)} = 1728_{16}$, $k_3^{(1)} = 105c_{16}$, $k_3^{(2)} = b820_{16}$, $k_3^{(3)} = 1c7e_{16}$, $k_3^{(4)} = 5e38_{16}$, $k_3^{(5)} = 14bc_{16}$, $k_3^{(6)} = 6a29_{16}$, $k_3^{(7)} = b3db_{16}$, $k_3^{(8)} = af67_{16}$, $k_3^{(9)} = 035e_{16}$.

11. *IDEA* алгоритмінің керішифрлаудың раундтық кілттерін $k_5^{(i)}$ ($i = 1...9$) анықтау, егер шифрлаудың раундтық кілттері тең: $k_5^{(1)} =$

$7b3d_{16}$, $k_5^{(2)} = 1af6_{16}$, $k_5^{(3)} = 8035_{16}$, $k_5^{(4)} = 3370_{16}$, $k_5^{(5)} = = 1266_{16}$, $k_5^{(6)} = f424_{16}$, $k_5^{(7)} = c7e8_{16}$, $k_5^{(8)} = 92d4_{16}$.

12. *IDEA* алгоритмінің керішифрлаудың раундтық кілттерін $k_1^{(i)}$ ($i = 1...9$) анықтау, егер шифрлаудың раундтық кілттері тең: $k_1^{(1)} = 3f42_{16}$, $k_1^{(2)} = 96a2_{16}$, $k_1^{(3)} = 192d_{16}$, $k_1^{(4)} = fa32_{16}$, $k_1^{(5)} = edf4_{16}$, $k_1^{(6)} = e500_{16}$, $k_1^{(7)} = 05ca_{16}$, $k_1^{(8)} = 820b_{16}$, $k_1^{(9)} = 3704_{16}$.

13. Егер кіріс деректер $M = 3d550f51d71ee0aa_{16}$ тең болса, *IDEA* алгоритмімен бірінші шифрлау раундының алмастыру (өзгерту) кезеңінің нәтижесін анықтау. Шифрлаудың раундтық кілттері тең: $k_1^{(1)} = 3f42_{16}$, $k_2^{(1)} = 4cdc_{16}$, $k_3^{(1)} = 105c_{16}$, $k_4^{(1)} = = a00d_{16}$.

14. *IDEA* алгоритмінің бірінші раундының *MA* құрылғысының нәтижесін анықтау, егер оның кірісіндегі деректер тең: $f_1^{(1)} = 4cb9_{16}$, $f_2^{(1)} = 4000_{16}$. Шифрлаудың раундтық кілттері тең: $k_5^{(1)} = 7b3d_{16}$, $k_6^{(1)} = be8c_{16}$.

15. *IDEA* алгоритмімен деректерді шифрлау нәтижесін анықтау, егер *8*-ші раундтаң кейін нәтиже тең: $D_1^{(8)} = e2fb_{16}$, $D_2^{(8)} = cd9d_{16}$, $D_3^{(8)} = cb10_{16}$, $D_4^{(8)} = 9936_{16}$. Шифрлаудың раундтық кілттері тең: $k_1^{(9)} = 3704_{16}$, $k_2^{(9)} = 1728_{16}$, $k_3^{(9)} = 035e_{16}$, $k_4^{(9)} = cf6f_{16}$.

16. *IDEA* алгоритмінің шифрлау кезеңінің екінші раундының *MA* құрылғысының кіріс деректерін анықтау, егер берілген раундтың алмастыру нәтижелері тең: $X_1^{(1)} = abc3_{16}$, $X_2^{(1)} = 5c2d_{16}$, $X_3^{(1)} = e77a_{16}$, $X_4^{(1)} = 1c2d_{16}$.

17. *Camellia* өңделетін блок көлемі, шифр кілтінің көлемі және раунд кілттерінің көлемі қандай?

18. *Camellia* раундтар саны неше?

19. *Camellia* раундтарында қандай операциялар қолданылады?

20. *Camellia* деректерді шифрлау кезінде раундтық кілттерді қалыптастыру алгоритмін түсіндіріңіз.

21. *Camellia* көмегімен деректерді шифрлау алгоритмін түсіндіріңіз.

22. *Camellia* көмегімен деректерді керішифрлау алгоритмін түсіндіріңіз.

10 тарау

АСИММЕТРИЯЛЫҚ КРИПТОГРАФИЯЛЫҚ ШИФРЛАУ ЖҮЙЕЛЕРІ

10.1. ТАРИХЫ ЖӘНЕ НЕГІЗГІ ИДЕЯЛАР

Ашық кілті бар криптографияның (асимметриялық криптографиялық жүйелер) идеяларын және әдістерін жақсы түсінуге көмектесетін үш есептің шешімін қарастырамыз. Осы есептердің барлығының мыңызды тәжірибелік мәні бар.

Бірінші есеп - құпиясөздерді компьютерде сақтау [10]. Желідегі әр пайдаланушының өзінің құпиясөзі бар екені белгілі. Желіге кіру кезінде пайдаланушы өзінің есімін (құпия емес) көрсетеді және содан кейін құпиясөз енгізеді. Проблема келесіде: егер құпиясөзді компьютердің дискінде сақтаса, онда қасқой оны оқу, ал содан кейін рұқсатсыз қатынас құру (егер қасқой сол желінің жүйелік әкімшісі болып жұмыс атқарса, онда оны өте оңай жасауға болады) үшін қолдану мүмкін. Сондықтан компьютерде құпиясөздерді сақтауды оны онай *"бұзу"* мүмкін болмайтындай ұйымдастыру қажет.

Екінші есеп әуе шабуылына қарсы қорғаныс (ӘҚҚ) жүйелесінің және радиолокаторлардың пайда болуымен пайда болды. Ұшақ шекараны қыйып өткен кезде радиолокатор құпиясөзді сұрайды. Егер құпиясөз дұрыс болса, онда ұшық *"өзінікі"*, басқа жағдайда - *"бөтен"*. Бұл жерде келесі проблема пайда болады: құпиясөз ашық арнамен (әуе орта) жіберілетін болғандықтан, қасқой барлық сөйлесулерді тыңдап дұрыс құпиясөзді біле алады. Содан кейін *"бөтен"* ұшақ сұраныс кезінде алдында жолай ұстаған *"дұрыс"* құпиясөзді локаторға жауап ретінде қайталап жібереді.

Үшінші есеп алдынғыға ұқсас және қашықтық қатынас құруы бар компьютерлік желілерде пайда болады, мысалы, банк және

клиент қатынас құру кезінде. Әдетте сеанс алдында банк клиенттен есімін сұрайды, ал содан кейін құпиясөзді, бірақ байланыс арнасы ашық болғандықтан, қасқой құпиясөзді біле алады.

Бүгінгі күні барлық осы проблемалар криптографиялық әдістерді қолданумен шешіледі. Осы есептердің шешімі маңызды түсінік *біржақты функцияда* (*one-way function*) негізделген.

10.1 анықтама. X (x ∈ X) шекті көптікте анықталған функция берілсін

$$y = f(x), \tag{10.1}$$

ол үшін қайтымды функция бар

$$x = f^{-1}(y). \tag{10.2}$$

Функция *біржақты* деп аталады, егер (10.1) өрнек бойынша есептеу – қарапайым есеп, ал (10.2) бойынша – үлкен есептеу ресурстарды қолдануды талап ететін күрделі есеп, мысалы, қуатты суперкомпьютердің 10^6–10^{10} жұмыс істеу жылын.

Берілген анықтама формалды емес. Біржақты функцияның қатаң анықтамасын [10, 18, 26] табуына болады, бірақ біздің мақсатымыз үшін жоғарыда келтірілгені жеткілікті.

Біржақты функция мысалы ретінде келесіні қарастырамыз:

$$y = a^x \bmod p, \tag{10.3}$$

бұл жерде p – кейбір жай сан (яғни қалдықсыз тек өзіне және бірге бөлінетін); $x – \{1,2,... p–1\}$ көптігінен бүтін сан.

Қайтымды функция

$$x = \log_a y \bmod p \tag{10.4}$$

белгіленеді және дискретті логарифм деп аталады.

Ең жақсы заманауи компьютерлерді қолдану кезінде (10.4) бойынша есептеуді қиындату үшін, кәзіргі уақытта *512* биттен астам көлемі бар сандар қолданылады. Тәжірибеде жиі басқада біржақты функцияларды қолданады, мысалы, *хэш-функцияларды,* олар *128–512* бит көлеміндегі қысқа сандармен жұмыс істейді.

Алдымен (10.3) бойынша есептеу жеткілікті тез орындалу мүмкін екенін көрсетеміз. $a^{16} \bmod p$ санын есептеу мысалынан бастаймыз. Жазуға болады:

$$a^{16} \bmod p = (((a^2)^2)^2)^2 \bmod p,$$

яғни берілген функцияның мәні *«аңқау»* нұсқадағы он бес $a \cdot a \cdot \ldots \cdot a$ операцияның орнына төрт көбейту операциясымен есептеледі. Осыда жалпы алгоритм негізделген.

Алгоритмді сипаттау үшін $t = [\log_2 x]$ шамасын енгіземіз - $\log_2 x$ (ары қарай барлық логарифмдер екілік болады, сондықтан логарифм негізін *2* көрсетпейміз) бүтін бөлімін. Сандар қатарын есептейміз:

$$a, a^2, a^4, a^8, \ldots, a^{2^t} \pmod{p}. \qquad (10.5)$$

Қатарда (10.5) әр сан алдынғы санды өзіне *p* модулі бойынша көбейтумен алынады. Дәреженің *x* көрсеткішін екілік санақ жүйесінде жазамыз:

$$x = (x_t x_{t-1} \ldots x_1 x_o)_2.$$

Сонда $y = a^x \bmod p$ саны келесідей есептелу мүмкін

$$y = \prod_{i=0}^{t} a^{x_i \cdot 2^i} \bmod p. \qquad (10.6)$$

Барлық есептеулер *p* модулі бойынша орындалады.

10.1 мысал. $3^{100} \bmod 7$ есептеу қажет болсын.

$t = [\log 100] = 6$ бар. Қатар (10.5) сандарын есептейміз:

a	a^2	a^4	a^8	a^{16}	a^{32}	a^{64}
3	*2*	*4*	*2*	*4*	*2*	*4*

(10.7)

Жалпы жағдайда келесі әділ.

10.1 мақұлдау ((10.3) есептеулердің қиындығы туралы). Сипатталған әдіс бойынша (10.3) есептеу кезінде көбейту операциялар саны $2 \cdot \log x$ аспайды.

Дәлелдеу. (10.5) қатар сандарын есептеу үшін *t* көбейту қажет, (10.6) бойынша *y* есептеу үшін *t* көбейтуден көп емес (10.1 мысалын қараңыз). $t = [\log x]$ шартынан, $[\log x] \le \log x$ екенін есепке алып, дәлелденетін мақұлдау әділ екені туралы қорытынды жасаймыз.

Ескерту. Ары қарай көрсетілетіндей, *p* модулі бойынша дәреже есептеу кезінде тек $x < p$ тек көрсеткіштерін қолданудың мәні бар. Бұл жағдайда (10.3) есептеу кезінде көбейту операциялардың саны $2 \cdot \log p$ аспайтынын айтуға болады.

Қайтымды функцияны (10.4) есептеудің осындай тиімді алгоритмдері белгісіз екенін айтып кету өте маңызды. (10.4) есептеудің *«сәбі қадамы, дәу қадамы»* деп аталатын әдістерінің бірі 10.2 тарауда толық сипатталады. Бұл әдіс шамамен $2\sqrt{p}$ операцияны талап етеді.

Үлкен p кезінде (10.3) функциясы расында біржақты екенін көрсетеміз, егер қайтымды функцияны есептеу үшін *«сәбі қадамы, дәу қадамы»* әдісі қолданылса. Келесі нәтижені аламыз (10.1 кесте).

10.1 кесте

Түзу және қайтымды функцияны есептеу үшін көбейту саны

p жазуында ондық белгі саны	(10.3) есептеу($2 \cdot \log p$ көбейту)	(10.4) есептеу($2 \cdot \sqrt{p}$ көбейту)
12	$2 \cdot 40 = 80$	$2 \cdot 10^{6}$
60	$2 \cdot 200 = 400$	$2 \cdot 10^{30}$
90	$2 \cdot 300 = 600$	$2 \cdot 10^{45}$

10.1 кестесінен көрініп тұр, егер *50–100* ондық сандардан тұратын модульдерді қолданса, онда *«түзу»* функциясы тез есептеледі, ал қайтымды – есептелмейді. Мысалы, екі *90*-белгілік санды 10^{-14} секундта көбейтетін суперкомпьютерді қарастырамыз (заманауи компьютерлер үшін бұл жеткізілімді емес). Сондай компьютерге (10.3) есептеу үшін:

$$T_{ВЫЧ.ПР.} = 600 \cdot 10^{-14} = 6 \cdot 10^{-12} \text{ сек.},$$

қажет, ал (10.4) есептеу үшін

$$T_{ВЫЧ.ОБР.} = 10^{45} \cdot 10^{-14} = 10^{31} \text{ сек.},$$

яғни 10^{22} жылдан астам.

Бұдан шығады, қайтымды функцияларды сандардың ұзындығы шамамен *90* ондық сан болса мүмкін емес, және қатарлас есептеулерді және компьютерлік желілерді қолдану жағдайды айтарлықтай өзгертпейді. Қарастырған мысалда қайтымды функция $2 \cdot \sqrt{p}$ операциямен есептеледі деп ойланған. Кәзіргі уақытта дискретті логарифмдерді оданда *«жылдам»* есептеу әдістері белгілі, бірақ жалпы жағдай өзгермеген – олар талап етітін

операциялар саны $2 \cdot \log p$ көп үлкен. Сонымен, (10.3) функциясы шынымен біржақты екенін айтуға болады, бірақ ескертумен. Қайтымды функция (10.4) *«түзу»* сияқты тез есептелмейтіні ешкіммен дәлелденбеген.

Осы тараудың басында сипатталған барлық үш есепті шешу үшін (10.3) біржақты функциясын қолданамыз, бірақ тұра осылай кез келген басқа біржақты функцияны қолдануға болатынын ұмытуға болмайды.

Компьютер жадысында құпиясөздерді сақтаудан бастаймыз. Есептің шешімі құпиясөздер мүлдем сақталмайтында негізделген! Нақтырақ, желіде тіркелген кезде пайдаланушы өзінің есімі мен құпиясөзді береді: мысалы, оның есімі *"жеміс"*, ал құпиясөзі - *"өрік"*. Компьютер *"өрік"* сөзін x санның екілік жазбасы ретінде қарастырады және (10.3) есептейді, бұл жерде a және p - екі құпия емес сан, олар бәріне белгілі болу мүмкін. Содан кейін компьютер жадысында (*есім, y*) жұп пайда болады, бұл жерде x=*құпиясөз* болған кезде y (10.3) бойынша есептелген. Пайдаланушының барлық болашақ кірістері кезде (*"жеміс"* - *"өрік"*) жұбын енгізген соң, компьютер x=*"өрік"* пен (10.3) бойынша $y_{НОВ}$ жаңа мәнін есептейді және алдында есептелген компьютер жадысында сақталған y салыстырады. Егер $y_{НОВ}$ осы есімге сәйкес жадыда сақталған y тең болса, онда бұл заңды пайдаланушы. Басқа жағдайда бұл қасқой.

Қасқой y бойынша x іздеп көру мүмкін. Бірақ *90*-белгісі бар сандар кезінде ол үшін 10^{22} астам жыл қажет.

Сонымен, көрсетілген құпиясөзді сақтау жүйесі өте сенімді және кәзіргі уақытта көптеген операциялық жүйелерде қолданылады.

Екінші есептің шешімін қарастырамыз (ӘҚҚ және ұшақ). Келесі әдісті қолдануға болады. Әр *"өз"* ұшағына құпия есім меншіктеледі, тек ӘҚҚ және ұшақтың борт компьютеріне белгілі. Мысалы, ұшақтың біріне *СҰҢҚАР* құпия есімі меншіктелсін, және бұл ұшақ шекараға *2014 жылдың 1 ақпанның 12 сағат 45 минутында* жақындасын. Онда шекараға жақындаудың алдында ұшақтың борт компьютері сөзді қалыптастырады:

СҰҢҚАР	*2014*	*02*	*01*	*12*	*45*
есім	*жыл*	*ай*	*күн*	*сағат*	*минут*

Басқа сөзбен, ұшақтағы компьютер және *ӘҚҚ* бекеті құпия сөзге ағымдағы уақыт туралы ақпаратты қосады және, алынған сөзді *x* саны ретінде қарастырып, $y = a^x \bmod p$ есептейді, бұл жерде *a* және *p* құпия емес. Содан кейін ұшақ *y* санын ӘҚҚ бекетіне хабарлайды. Бекет өзі есептеген *y* ұшақтан алынғанмен салыстырады. Егер есептелген және алынған мәндер тең болса, онда ұшақ *"өзінікі"* деп танылады.

Қасқой бұл жүйені бұза алмайды. Шынында, бір жағынан, ол *СҰНҚАР* құпиясөзін білмейді және оны *y* бойынша таба алмайды, себебі *x* мәнін *y* бойынша табу 10^{22} жылды алады деп айтайық. Басқа жағынан, ол жай *y* көшіріп және оны болашақта жауап ретінде қолдана алмайды, себебі шекараны өту уақыты ешқашан қайталанбайды және *y* келесі мәндерінің біріншіден айырмашылықтары болады.

ӘҚҚ есебінің қарастырылған шешу нұсқасы ұшақта және локаторда сағатты нақты синхрондауды талап етеді. Бұл проблема жеткілікті жеңіл шешіледі. Мысалы, навигация қызметі үнемі уақыт белгілерін ашық түрде (уақыт құпия емес) жібереді, және барлық ұшақтар мен локаторлар бұл белгілерді өз сағаттарын синхрондау үшін қолданады. Бірақ одан жіңішке проблемалар бар. Уақыт белгісі *x* сөзіне *y* барлық есептелетін мәндері әртүрлі болу үшін және қасқой оларды қайта қолданбауға мүмкіндік бермеу үшін қосылады. Бірақ қасқой *y* сол сәтте тез сол минут ішінде қайталау мүмкін. Бұл мүмкіндікті қалай болдырмауға болады? Бұл бірінші сұрақ. Басқа қиындық ұшық *y* санын *45*-ші минуттың соңында жіберген, ал локатор оны *46*-ші минуттың басында қабылдаған жағдайда болады. Бұл проблемаларды шешуді оқырманның өздеріне қалдырамыз.

"ӘҚҚ есебін" шешудің басқа тәсілі мүмкін, егер ұшақтан локаторға деректерді жіберудің қосымша ашық арнасын қолданса. Жоғарыдай, әр *"өзіміздің"* ұшағымыз және локатор ауыстырылмайтын құпия сөзді (*СҰҢҚАР* сияқты) біледі. Мақсатты тауып, локатор оған кездейсоқ генерацияланған санды *a* (*"шақыру"*) жібереді. Ұшақ $y = a^x \bmod p$ есептейді, және *y* санын локаторға хабарлайды, бұл жерде *x* – құпиясөз (*СҰНҚАР*). Локатор сол есептерді орындайды және есептелген *y* қабылданғанмен салыстырады. Бұл сұлбада сағаттарды синхрондау қажет емес, бірақ, бұрынғыдай қасқой *y* санын қайтала алмайды, себебі локатор

әртүрлі шақыруларды (*a*) жібереді. Қызығы, бұл есеп біржақты функцияны қолданумен тарихи бірінші болды.

Үшінші есеп тұра осылай шешіледі, және қарастырылған құпиясөзді қалыптастырудың екі әдісі де реалды желілік хаттамаларда қолданылады.

10.2. АШЫҚ КІЛТПЕН БІРІНШІ КРИПТОГРАФИЯЛЫҚ ЖҮЙЕ – ДИФФИ-ХЕЛЛМАН ЖҮЙЕСІ

Бұл криптожүйе *70*-ші жылдардың ортасында *У. Диффи* (*Whitfield Diffie*) және *М. Хеллман* (*Martin Hellman*) американдық ғалымдарымен ашылды және крипторафия мен оның тәжірибелік қолдануларында кәдімгі төңкеріске алып келді. Бұл қорғалған арналармен жіберілетін құпия кілттерді қолданусыз ақпаратты қорғауға мүмкіндік берген бірінші жүйе. Осындай жүйелерді пайдалану сұлбаларының бірін көрсету үшін N пайдаланушылармен байланыс желісін қарастырамыз, бұл жерде N – үлкен сан. Олардың әр жұбы үшін құпия байланысын ұйымдастыру қажет. Егер құпия кілттерді таратудың әдеттегі жүйесін қолдансақ, онда әр абоненттер жұбында өзінің құпия кілті болу қажет, яғни барлығы

$$C_N^2 = \frac{N(N-1)}{2} \approx \frac{N^2}{2}$$

кілт қажет.

Егер *100* абонент болса, онда *5000* кілт қажет, егер абоненттер саны 10^4 болса, онда $5 \cdot 10^7$ кілт қажет. Абоненттер саны үлкен болған кезде, оларды құпия кілттермен қамтамасыз ету жүйесі өте үлкен және қымбат болатыны көрініп тұр.

Диффи және *Хеллман* бұл проблеманы кілттерді ашық тарату және есептеу арқылы шешті.

A, *B*, *C*, ... абоненттері үшін байланыс жүйесі құрылсын. Әр абонентте өзінің құпия және ашық ақпараты бар. Бұл жүйені ұйымдастыру үшін үлкен жай сан p және кейбір сан g ($1<g<p-1$) таңдалады, $\{1,2,..., p-1\}$ көптігінен барлық сандар $g \bmod p$ (сондай g сандарын табудың әртүрлі әдістері белгілі, олардың бірі төменде көрсетіледі) әртүрлі деңгейлері ретінде көрсетілу мүмкін.

Абоненттер d_A, d_B, d_C үлкен сандарын таңдайды, оларды құпия сақтайды (әдетте бұндай таңдауды кездейсоқ, кездейсоқ сандар

датчиктерін қолдану арқылы жасауды ұсынады). Әр абонент сәйкес y санын есептейді, ол басқа абоненттерге ашық жіберіледі,

$$\begin{cases} e_A = g^{d_A} \bmod p, \\ e_B = g^{d_B} \bmod p, \\ e_C = g^{d_C} \bmod p. \end{cases} \qquad (10.9)$$

Нәтижесінде келесі кестені аламыз.

10.2 кесте

Диффи–Хеллман жүйесінде пайдаланушылардың кілттері

Абонент	Құпия кілт	Ашық кілт
A	d_A	e_A
B	d_B	e_B
C	d_C	e_C

A абоненті B байланыс сеансын құрамын деп шешсін, екі абоненттерге де 10.2 кестесінен ашық ақпарат қол жетімді. A абоненті B ашық арнамен оған хабар жіберу туралы хабарлайды. Содан кейін A абоненті келесі шаманы есептейді:

$$K_{AB} = (e_B)^{d_A} \bmod p. \qquad (10.10)$$

A басқа ешкім бұны жасай алмайды, себебі x_A саны құпия.

Өз ретімен B абоненті келесі санды есептейді:

$$K_{BA} = (e_A)^{d_B} \bmod p. \qquad (10.11)$$

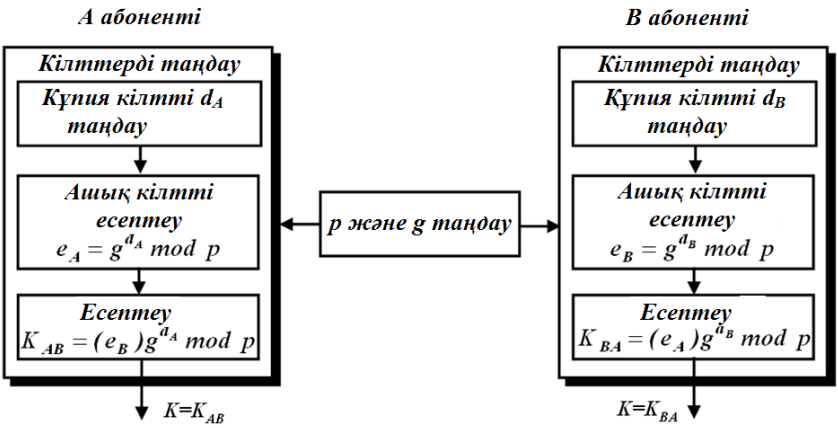
Диффи–Хеллманның криптографиялық жүйе сұлбасы 10.1 суретінде көрсетілген.

10.2 мақұлдау. $K_{AB} = K_{BA}$.

Дәлелдеу. Шынында,

$$K_{AB} = (e_B)^{d_A} \bmod p = (g^{d_B})^{d_A} \bmod p = \\ = g^{d_A d_B} \bmod p = (e_A)^{d_B} = K_{BA}.$$

Бұл жерде бірінші теңдік (10.10), екінші және төртінші – (10.9), соңғысы – (10.11) шығады.



10.1 сурет - *Диффи–Хеллманның* криптографиялық жүйе сұлбасы

Жүйенің басты қасиеттері:

1) *A* және *B* абоненттері ашық байланыс сызығы арқылы жіберілмеген бір санды алды $K = K_{AB} = K_{BA}$;

2) қасқой d_A және d_B құпия сандарын білмейді, сондықтан K_{AB} немесе K_{BA} санын есептей алмайды (ол e_A бойынша d_A құпия санын табуға ұмтылу мүмкін ((10.9) қараңыз), бірақ үлкен *p* болған кезде бұл мүмкін емес (миллиондаған жылдар қажет)).

A және *B* абоненттері $K = K_{AB} = K_{BA}$ деректерді шифрлау және керішифрлау үшін құпия кілт ретінде қолдану мүмкін. Тұра солай абоненттердің кез келген жұбы тек соларға белгілі құпия кілтті есептеу мүмкін.

Енді жоғарыда айтылған *g* санын таңдау есебін қарастырамыз. Кез келген берілген *p* кезде ол *p–1* санын жай көбейткіштерге жіктеумен байланысты қиын есеп болу мүмкін. Қарастырылған жүйенің жоғары беріктілігін қамтамасыз ету үшін *p–1* санында міндетті түрде жай үлкен көбейткіш болу қажет (қарсы жағдайда, мысалы, [7, 14, 20] сипатталған *Полиг-Хеллман алгоритмі* дискретті логарифмды тез есептейді). Сондықтан жиі келесі әдісті қолдануды ұсынылады. Жай сан *p* келесі теңдік орындалатындай таңдалады

$$p = 2 \cdot q + 1,$$

бұл жерде *q* – жай сан, онда *g* ретінде келесі өрнектер орындалатын кез келген санды алуға болады

$$1 < g < p - 1 \quad \text{және} \quad g^q \bmod p \neq 1.$$

10.2 мысал. Келесі болсын

$$p = 23 = 2 \cdot 11 + 1 \quad (q = 11).$$

Параметр g таңдаймыз. Аламыз $g = 3$. Тексереміз:

$$g^q \bmod p = 3^{11} \bmod 23 = 1,$$

және бұл осындай g келмейтінін көрсетеді. Аламыз $g = 5$. Тексереміз:

$$g^q \bmod p = 5^{11} \bmod 23 = 22 \neq 1.$$

Сонымен, біз параметрлерді қабылдадық $p = 23$, $g = 5$.

Енді әр абонент құпия санды таңдайды және оған сәйкес келетін ашық санды есептейді. $d_A = 7$, $d_B = 13$ таңдалсын. Есептейміз

$$e_A = g^{d_A} \bmod p = 5^7 \bmod 23 = 17;$$

$$e_B = g^{d_B} \bmod p = 5^{13} \bmod 23 = 21.$$

A және B жалпы құпия кілтті қалыптастырғысы келсін. Ол үшін A есептейді

$$K_{AB} = (e_B)^{d_A} \bmod p = 21^7 \bmod 23 = 10,$$

ал B есептейді

$$K_{BA} = (e_A)^{d_B} \bmod p = 17^{13} \bmod 23 = 10.$$

Енді оларда арнамен жіберілмеген жалпы кілті бар $K = K_{AB} = K_{BA} = 10$.

10.3. САНДАР ТЕОРИЯСЫНЫҢ ЭЛЕМЕНТТЕРІ

Көптеген криптографиялық алгоритмдер классикалық сандар теориясына негізделеді. Бұл теориядан қажетті минимумды қарастырамыз. Сандар теориясынан классикалық *Ферма*, *Эйлер теоремалары* және бір қатар басқа нәтижелер дәлелдеулерсіз беріледі, оларды сандар теориясы бойынша кез келген оқулықтан табуға болады (мысалы, [14]).

10.2 анықтама. Бүтін оң сан *жай* деп аталады, егер ол өзіне және бірге ғана бөлінсе.

10.3 мысал. 11 және *23* сандары - жай сандар; *27* және *33* – жікті сандар (*27* саны *3* және *9* бөлінеді, *33* саны *3* және *11* бөлінеді).

10.3 теорема (арифметиканың негізгі теоремасы). Кез келген бүтін оң сан жай сандар көбейтіндісі ретінде көрсетілу мүмкін, тек бір түрде.

10.4 мысал. 27 = 3·3·3, 33 = 3·11.

10.3 анықтама. Екі сан *өзара жай сан* деп аталады, егер олардың бірден басқа ортақ бөлгіштері болмаса.

10.5 мысал. 27 және *28* өзара жай сандар (оларда бірден басқа ортақ бөлгіштері жоқ), *27* және *33* өзара жай сандар емес (оларда ортақ бөлгіш бар *3*).

10.4 анықтама (*Эйлер функциясы*). $n \geq 1$ бүтін сан берілсін. *Эйлер* функциясының мәні $\varphi(n)$ *1, 2, 3, ... , n–1* қатарындағы *n* өзара жай болатын сандар санына тең.

10.6 мысал.

$\varphi(10)$	$\varphi(12)$
1, 2, 3, 4, 5, 6, 7, 8, 9	*1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11*
$\varphi(10) = 4$	$\varphi(12) = 4$

Бұл жерде екі сызықпен *n* өзара жай сан болмайтын сандар көрсетілген.

10.3 мақұлдама. Егер *p* – жай сан болса, онда $\varphi(p) = p–1$.

Дәлелдеу. 1, 2, 3, ... , p–1 қатарында барлық сандар *p* өзара жай сандар, себебі *p* - жай сан және анықтама бойынша басқа сандарға бөлінбейді.

10.4 мақұлдама. p және *q* – екі әртүрлі жай сан болсын ($p \neq q$). Онда

$$\varphi(p \cdot q) = (p–1) \cdot (q–1).$$

Дәлелдеу. 1, 2, ... , p·g–1 қатарында $p \cdot q$ өзара жай сандар болмайтын келесі сандар

$$p, 2 \cdot p, 3 \cdot p, \ldots, (q–1) \cdot p \quad \text{и} \quad q, 2 \cdot q, 3 \cdot q, \ldots, (p–1) \cdot q.$$

Бұндай сандардың барлығы $(q–1)+(p–1)$ болады. Сондықтан, $p \cdot q$ өзара жай сандардың саны келесі болады [14]

$$p \cdot q - 1 - (p-1) - (q-1) = p \cdot q - q - p + 1 = (p-1) \cdot (q-1).$$

10.6 теорема (Ферма теоремасы). p – жай сан болсын және $0<a<p$. Онда

$$a^{p-1} \bmod p = 1.$$

10.7 мысалдар.

$p = 13, a = 2;$

$$2^{12} \bmod 13 = (2^2)^2 \cdot ((2^2)^2)^2 \bmod 13 = 3 \cdot 9 \bmod 13 = 1,$$

$p = 11, a = 10;$

$$10^{10} \bmod 11 = 10^2 \cdot ((10^2)^2)^2 \bmod 11 = 1 \cdot 1 \bmod 11 = 1.$$

$p = 10, a = 10;$

$$10^9 \bmod 11 = 10 \cdot ((10^2)^2)^2 \bmod 11 = 10 \cdot 1 \bmod 11 = 10 \neq 1.$$

10.7 теорема (Эйлер теоремасы). a және b – өзара жай сандар болсын. Онда

$$a^{\varphi(b)} \bmod b = 1.$$

Ферма теоремасы b – жай сан болған кезде *Эйлер* теоремасының дара оқиғасы болады.

10.8 мысал.

$a = 5$ және $b = 12$ болсын. Алдында (10.6 мысалында) келесі анықталған болатын

$$\varphi(b) = \varphi(12) = 4.$$

Онда

$$a^{\varphi(b)} \bmod b = 5^4 \bmod 12 = (5^2)^2 \bmod 12 = (-1)^2 \bmod 12 = 1.$$

$a = 2$ және $b = 21$ болсын. $\varphi(b)$ есептеп, келесіні аламыз

$$\varphi(b) = \varphi(21) = 12,$$

$$a^{\varphi(b)} \bmod b = 2^{12} \bmod 21 = 2^4 \cdot (2^4)^2 \bmod 21 = 16 \cdot 4 \bmod 21 = 1.$$

Бізге *Эйлер* теоремасына жақын тағы бір теорема қажет болады.

10.8 теорема. Егер p және q – жай сандар, $p \neq q$ және k – кез келген бүтін сан болса, онда

$$a^{k\cdot\varphi(p\cdot q)+1} \bmod (p\cdot q) = a. \qquad (10.12)$$

10.9 мысал. $p = 5$, $q = 7$ деп аламыз. Онда $p\cdot q = 35$, ал *Эйлер* функциясы – $\varphi(35) = 4\cdot 6 = 24$. Келесі жағдайды қарастырамыз $k = 2$, яғни сандардың дәрежелерін аламыз $k\cdot\varphi(p\cdot q)+1 = 2\cdot 24 + 1 = 49$.

$a = 9$ болсын. Нәтижесінде аламыз

$$a^{k\cdot\varphi(p\cdot q)+1} \bmod (p\cdot q) = 9^{49} \bmod 35 = 9.$$

$a = 23$ болсын. Нәтижесінде аламыз

$$a^{k\cdot\varphi(p\cdot q)+1} \bmod (p\cdot q) = 23^{49} \bmod 35 = 23.$$

Бұл ғажап емес, себебі 9 және 23 сандарының әрқайсысы 35 модулімен өзара жай сандар, және *Эйлер* теоремасы бойынша $9^{24} \bmod 35 = 1$, $23^{24} \bmod 35 = 1$. Бірақ 10.8 теоремасы келесі сандар үшін де дұрыс болып қалады:

$$10^{49} \bmod 35 = 10, \quad 28^{49} \bmod 35 = 28,$$

сол уақытта *Эйлер* теоремасы олар үшін қолданымсыз (10 және 28 сандардың әрқайсысы 35 модулімен өзара жай сан емес және $10^{24} \bmod 35 = 15$, $28^{24} \bmod 35 = 21$).

10.5 анықтама. a және b – екі бүтін оң сан болсын. a және b сандарының *ең үлкен ортақ бөлгіші* a және b бөлетін ең үлкен сан c:

$$c = gcd(a, b).$$

Ең үлкен ортақ бөлгіш үшін *gcd* белгілеу *greatest common divisor* ағылшын сөздерінен шыққан және заманауи әдебиетте қабылданған.

10.10 мысал.

$$gcd(10,15) = 5; \quad gcd(8,28) = 4.$$

Бұл мысалда a санының мәні 10 (8), b санының – 15 (28), ал c санының – 5 (4).

Ең үлкен ортақ бөлгішті табу үшін келесі алгоритмді қолдануға болады - *Евклид* алгоритмі ретінде белгілі.

10.1 алгоритм. *Евклид* алгоритмі (псевдокод қолданылады):

КІРІС: Бүтін оң сандар a, b, $a \geq b$.

1. *while (b != 0)*
2. *{*

$r = a \bmod b;\ a = b;\ b = r;$

3. *}*

ШЫҒЫС: Ең үлкен ортақ бөлгіш $c = gcd(a, b)$.

10.11 мысал. Евклид алгоритмі көмегімен *gcd(28,8)* қалай есептелетінін көрсетеміз:

a:	*28*	*8*	*4*
b:	*8*	*4*	*0*
r:	*4*	*0*	

Бұл жерде әр баған алгоритмнің кезекті түсіндіруі. Процесс *b* нөлге тең болғанша қайталанады. Сонда *a* айнымалысының мәнінде жауап болады (*4*).

Көптеген криптографиялық жүйелер үшін жалпылама *Евклид* алгоритмі актуалды, онымен келесі теорема байланысты.

10.9 теорема. a және *b* – екі бүтін оң сан болсын. Онда *x* және *y* бүтін (оң болу міндетті емес) сандар бар, олар үшін

$$a \cdot x + b \cdot y = gcd(a, b). \qquad (10.13)$$

Жалпылама *Евклид* алгоритмі *gcd(a,b)* және (10.13) сәйкес келетін *x, y* табу үшін қызмет етеді. Үш жол енгіземіз $U = (u_1, u_2, u_3)$, $V = (v_1, v_2, v_3)$ және $T = (t_1, t_2, t_3)$.

Онда алгоритм келесідей жазылады.

10.2 алгоритм. Евклидтың жалпылама алгоритмі (псевдокод қолданылады):

КІРІС: Бүтін оң сандар *a, b*, $a \geq b$.

1. $U = \{ a, 1, 0 \};\ V = \{ b, 0, 1 \}$.
2. *while* $(v_1 != 0)$
3. *{*

$q = u_1 \ div\ v_1;$

$T = \{ u_1 \ div\ v_1;\ u_2 - q \cdot v_2\ ;\ u_3 - q \cdot v_3 \};$

$U = V;\ V = T;$

4. *}*
5. $U = \{ gcd(a, b);\ x;\ y \}$

ШЫҒЫС: (10.13) сәйкес келетін $gcd(a, b);\ x;\ y$.

Нәтиже *U* жолында.

Алгоритмдегі *div* операциясы – бұл бүтін санды бөлу

$$a \; div \; b = [\, a / b \,].$$

10.2 алгоритмнің дұрыстығын дәлелдеуді [2, 14] табуға болады.

10.12 мысал. $a = 28$, $b = 19$ болсын. (10.13) сәйкес келетін x және y сандарын табамыз.

U				28	1	0	
V	U			19	0	1	
T	V	U		9	1	-1	$q = 1$
	T	V	U	1	-2	3	$q = 2$
		T	V	0	19	-28	$q = 9$

Берілген сұлбаны түсіндіреміз. Алдымен U жолына $(28,1,0)$ сандары жазылады, ал V жолына - $(19,0,1)$ сандары (бұл сұлбадағы бірінші екі жол). T жолы есептеледі (сұлбадағы үшінші жол). Содан кейін U жолы ретінде сұлбадағы екінші жол алынады, ал V ретінде - үшінші жол, және қайтадан T жолы есептеледі (сұлбадағы төртінші жол). Бұл процесс V жолының бірінші элементі нөлге тең болғанша қайталанады. Сонда сұлбаның соңының алдындағы жолда жауап болады. Біздің жағдайда $gcd(28,19) = 1$, $x = -2$, $y = 3$. Тексеруді орындаймыз: $28 \cdot (-2) + 19 \cdot 3 = 1$.

Жалпылама Евклид алгоритмінің тағы бір маңызды қолдануын қарастырамыз. Криптографияның көптеген есептерінде берілген c және m сандары үшін $d < m$ санын табу қажет болады, келесі орындалатын

$$c \cdot d \bmod m = 1. \qquad (10.14)$$

Осындай d бар болады тек егер c және m сандары өзара жай сандар болса.

10.6 анықтама. (10.14) сәйкес келетін d саны, c мультипликативті инверсиясы болады m модулі бойынша және жиі $c^{-1} \bmod m$ деп белгіленеді.

Берілген белгілеу инверсия үшін табиғи, себебі енді біз (10.14) келесі түрге қайта жазуымызға болады

$$c \cdot c^{-1} \bmod m = 1.$$

Модуль m бойынша есептеулерде c^{-1} көбейту c бөлуге сәйкес келеді. Осылай m модулі бойынша есептеулерде кез келген теріс дәрежелерді енгізуге болады:

$$c^{-e} \bmod m = (c^{e})^{-1} \bmod m = (c^{-1})^{e} \pmod m.$$

10.13 мысал. $3 \cdot 4 \bmod 11 = 1$, сондықтан *4* саны – бұл *3* санының *11* модулі бойынша мультипликативті инверсиясы. $3^{-1} \bmod 11 = 4$ жазуға болады. $5^{-2} \bmod 11$ екі тәсілмен табуға болады:

$$5^{-2} \bmod 11 = (5^{2} \bmod 11)^{-1} \bmod 11 = 3^{-1} \bmod 11 = 4,$$

$$5^{-2} \bmod 11 = (5^{-1} \bmod 11)^{2} \bmod 11 = 9^{2} \bmod 11 = 4.$$

Екінші тәсілмен есептеу кезінде бір $5^{-1} \bmod 11 = 9$ қолдандық. Шынымменде, $5 \cdot 9 \bmod 11 = 45 \bmod 11 = 1$.

Евклидтің жалпылама алгоритмі көмегімен инверсияны қалай есептеуге болатынын көрсетеміз. (10.14) теңдік кейбір бүтін *k* үшін

$$c \cdot d - k \cdot m = 1. \qquad (10.15)$$

c және *m* өзара жай сан екенін есептке алып, (10.15) келесі түрде жазамыз

$$m \cdot (-k) + c \cdot d = gcd(m, c) = 1, \qquad (10.16)$$

бұл толық (10.13) сәйкес келеді, тек айнымалылар басқаша белгіленген. Сондықтан, $c^{-1} \bmod m$ есептеу үшін, яғни *d* табу үшін, (10.16) теңдікті шешу үшін жалпылама Евклид алгоритмін қолдану қажет. Бізді *k* айнымалының мәндері қызықтырмайды, сондықтан *U, V, T* жолдарының екінші элементтерін есептемеуге болады. Сонымен қатар, егер *d* саны теріс болса, онда оған *m* қосу қажет, себебі анықтама бойынша *a mod m* саны *{0,1, ..., m–1}* көптігінен алынады.

10.14 мысал. $7^{-1} \bmod 11$ есептейміз. 10.12 мысалында көрсетілгендей есептеулерді жазу сұлбасын қолданамыз:

U					*11*	*0*	
V	*U*				*7*	*1*	
T	*V*	*U*			*4*	*-1*	$q = 1$
	T	*V*	*U*		*3*	*2*	$q = 1$
		T	*V*	*U*	*1*	*-3*	$q = 1$
			T	*V*	*0*	*11*	$q = 3$

$d = -3$ және $d \bmod 11 = (11 - 3) \bmod 11 = 8$ алады, яғни $7^{-1} \bmod 11 = 8$. Нәтижені тексереміз: $(7 \cdot 8) \bmod 11 = 56 \bmod 11 = 1$.

Ашық кілтті криптографияда маңызды операциялардың бірі модуль бойынша дәрежесін шығару. Дәрежесін шығарудың тиімді алгоритмді құру идеясы алдында (10.5) және (10.6) көрсетілген болатын. Қарастырылған алгоритмді жадыда сандар қатарын (10.5) сақтамау арқылы да жүзеге асыруға болады. Бұл алгоритмнің сипаттамасын бағдарламалық жүзеге асыруға жарайтын түрінде көрсетеміз. Алгоритмнің атында келесі факт көрсетілген: дәреже көрсеткіштің биттері оң жақтан сол жаққа қарастырылады, яғни кішіден үлкенге дейін.

10.3 алгоритм. Дәрежесін шығару (оң жақтан сол жаққа)

КІРІС: Бүтін сандар a, $x = (x_t x_{t-1}, ..., x_0)_2$, p.
ШЫҒЫС: Сан $y = a^x \bmod p$.
1. $y \leftarrow 1$; $s \leftarrow a$
2. FOR $i = 0,1,...,t$ DO
3. IF $x_i = 1$ THEN $y \leftarrow y \cdot s \bmod p$;
4. $s \leftarrow s \cdot s \bmod p$
5. RETURN y

Берілген алгоритм бойынша (10.6) шынымен y есептелінетінін көрсету үшін циклдың әр итерациясынан кейін айнымалылар дәрежелерін жазамыз. 10.1 мысалындағыдай $x = 100 = (1100100)_2$ болсын, онда:

i:	0	1	2	3	4	5	6
x_i:	0	0	1	0	0	1	1
y:	1	1	a^4	a^4	a^4	a^{36}	a^{100}
s:	a^2	a^4	a^8	a^{16}	a^{32}	a^{64}	a^{128}.

Кейбір жағдайларда келесі алгоритм тиімдірек болады, ода дәреже көрсеткіштерінің биттері сол жақтан оң жаққа қарастырылады, яғни үлкеннен кішігедейін.

10.4 алгоритм. Модуль бойынша дәрежесін шығару (сол жақтан оң жаққа):

КІРІС: Бүтін сандар a, $x = (x_t x_{t-1}, ..., x_0)_2$, p.
ВЫХОД: Сан $y = a^x \bmod p$.
1. $y \leftarrow 1$;
2. FOR $i = t,t-1,...,0$ DO
3. $y \leftarrow y \cdot y \bmod p$;

4. *IF* $x_i = 1$ *THEN* $y \leftarrow y \cdot a \bmod p$;
5. *RETURN* y.

10.4 алгоритмі 10.3 алгоритмнің есептегенін есептейтіне көз жеткізу үшін, $x = 100$ үшін әр цикл итерациядан кейін y айнымалылардың дәрежелерін жазамыз:

i:	6	5	4	3	2	1	0
x_i:	1	1	0	0	1	0	0
y:	a	a^3	a^6	a^{12}	a^{25}	a^{50}	a^{100}.

Осы тарауда келтірілген сандар теориясынан мәліметтер негізгі криптографиялық алгоритмдерді және әдістерді сипаттау үшін жеткілікті.

10.4. ШАМИРДЫҢ КРИПТОГРАФИЯЛЫҚ ЖҮЙЕСІ

А. Шамир (*Adi Shamir*) ұсынылған бұл криптографиялық жүйе ешқандай қорғалған арналары және құпия кілттері жоқ және мүмкін бір бірін көрмеген адамдар үшін ашық байланыс сызығымен құпия хабарлармен алмасуды ұйымдастыруға мүмкіндік беретін бірінші жүйе болды. *Диффи-Хеллманның* криптографиялық жүйесі тек құпия сөзді қалыптастыруға мүмкіндік береді, ал хабарды жіберу бұл сөз кілт ретінде қолданатын кейбір криптожүйені қолдануды талап етеді.

Жүйені сипаттаймыз. Екі абонент A және B байланыс сызығымен жалғансын. A абоненті M хабарын B абонентіне оның мазмұнын ешкім білмейтіндей жібергісі келеді. A кездейсоқ үлкен жай сан p таңдайды және оны ашық B жібереді. Содан кейін A екі санды e_A және d_A таңдайды, олар келесідей болу қажет

$$e_A \cdot d_A \bmod (p-1) = 1. \qquad (10.17)$$

Бұл сандарды A құпияда сақтайды және жібермейді. B да екі санды e_B және d_B таңдайды, олар келесідей болу қажет

$$e_B \cdot d_B \bmod (p-1) = 1, \qquad (10.18)$$

және оларды құпияда сақтайды.

Содан кейін A өзінің қысқа хабарын M үшқадамды хаттаманы қолданып жібереді. Егер $M < p$ (M сан ретінде қарастырылады), онда

M хабары бірден жіберіледі, егер $M > p$, онда хабар m_1, m_2, …, m_t түріне аударылады, бұл жерде барлық $m_i < p$, және содан кейін ретпен жіберіледі m_1, m_2, …, m_t. Бұл жерде әр m_i шифрлау үшін кездейсоқ жаңа (e_A,d_A) және (e_B,d_B) жұптарын таңдаған жөн - қарсы жағдайда жүйенің сенімділігі төмендейді. Кәзіргі уақытта осындай криптографиялық жүйе, ереже сияқты, сандарды жіберу үшін қолданылады, мысалы, мәндері p төмен құпия кілттерді жіберу үшін. Сонымен, біз тек $M < p$ жағдайды қарастырамыз. Хаттаманың сипаттамасын береміз.

1 қадам. *A* келесі санды есептейді

$$c_1 = M^{e_A} \bmod p, \tag{10.19}$$

бұл жерде M – алғашқы мәтін, және c_1 *B* жібереді.

2 қадам. *B*, c_1 алып, келесі санды есептейді

$$c_2 = c_1^{e_B} \bmod p \tag{10.20}$$

және c_2 *A* жібереді.

3 қадам. *A* санды есептейді

$$c_3 = c_2^{d_A} \bmod p \tag{10.21}$$

және оны *B* жібереді.

4 қадам. *B*, c_3 алып, келесі санды есептейді

$$c_4 = c_3^{d_B} \bmod p. \tag{10.22}$$

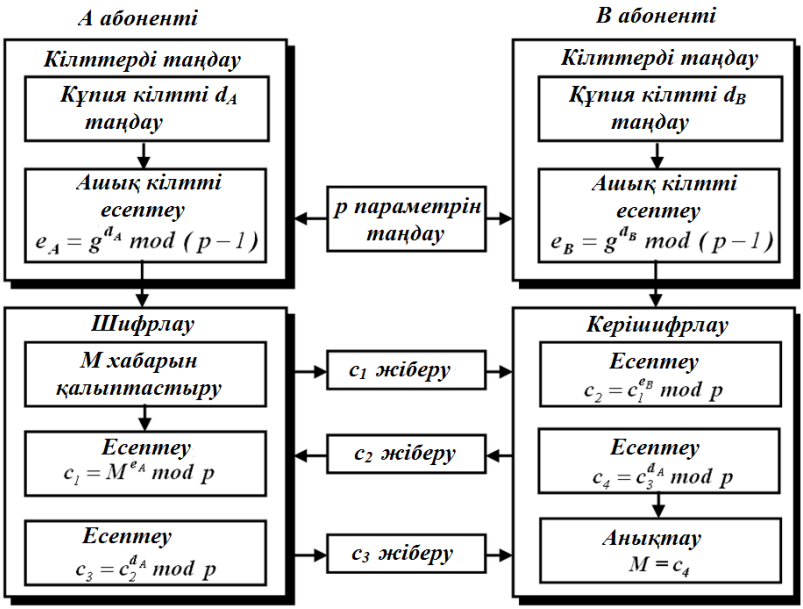
A абонентінен *B* абонентіне жіберу кезінде деректерді шифрлау процесстерін көрсететін *Шамирдің* криптографиялық жүйесінің сұлбасы 10.2 суретінде көрсетілген.

B абонентінен *A* абонентіне жіберу кезінде деректерді шифрлау процесстерін көрсететін *Шамирдің* криптографиялық жүйесінің сұлбасы алдында көрсетілгендей, тек бұл жағдайда хабарды жиберудің инициаторы *B* абоненті болады.

10.5 мақұлдау. (*Шамир* хаттамасының қасиеттері):

1) $c_4 = M$, яғни хаттаманы жүзеге асыру нәтижесінде *A* дан *B* шыныменде алғашқы хабар жіберіледі;

2) қасқой қандай хабар жіберілгенін біле алмайды.



10.2 сурет - *Шамирдің* криптографиялық жүйесінің сұлбасы

Дәлелдеу. Алдымен, кез келген бүтін санды $n \geq 0$ келесі түрде көрсетілу мүмкін екенін айта кетеміз

$$n = k \cdot (p-1) + r,$$

бұл жерде $r = n \bmod (p-1)$.

Сондықтан *Ферма* теоремасы негізінде ($x^{p-1} \bmod p = 1$)

$$x^n \bmod p = x^{k \cdot (p-1)+r} \bmod p = (1^k \cdot x^r) \bmod p = \\ = x^{n \bmod (p-1)} \bmod p. \qquad (10.23)$$

Мақұлдаудың бірінші тармағының дұрыстығы келесі теңдеулерден шығады:

$$c_4 = c_3^{d_B} \bmod p = (c_2^{d_A})^{d_B} \bmod p = (c_1^{e_B})^{d_A \cdot d_B} \bmod p =$$

$$= (M^{e_A})^{e_B \cdot d_A \cdot d_B} \bmod p = M^{e_A \cdot e_B \cdot d_A \cdot d_B} \bmod p =$$

$$= M^{(e_A \cdot e_B \cdot d_A \cdot d_B) \bmod (p-1)} \bmod p = M.$$

Соңының алдындағы теңдік (10.23) шығады, ал соңғысы орындалады келесі (10.17) және (10.18) арқасында.

Екінші тармақтың дәлелдеуі келесі ойда негізделген: *M* анықтамысы келетін қасқой үшін келесі стратегиядан тиімдірек стратегия жоқ. Алдымен ол (10.20) e_B есептейді, содан кейін d_B табады, және (10.22) бойынша $c_4 = M$ есептейді. Бірақ бұл стратегияны жүзеге асыру үшін қасқой дискретті логарифмдеу (10.20) есебін шешу қажет, ал ол үлкен *p* кезде мүмкін емес.

(10.17) және (10.18) сәйкес келетін e_A, d_A және e_B, d_B жұптарын табу әдісін сипаттаймыз. Тек *A* абоненті үшін әрекеттерді сипаттау жеткілікті, себебі *B* үшін әрекеттер тұра сондай. Санды e_A кездейсоқ, *p–1* (*p–1* жұп сан болғандықтан, тақ сандар ішінен іздеу дұрыс) өзаражай сан болатындай таңдаймыз. Содан кейін Евклидтың жалпылама алгоритмі көмегімен d_A есептейміз.

10.15 мысал. A абоненті *B M = 10* хабарын жібергісі келеді. *A* $p = 23$, $e_A = 7$ ($gcd(7, 22) = 1$) таңдайды және $d_A = 19$ есептейді. Тұра солай, *B* $e_B = 5$ (*22* өзаражай сан) және $d_B = 9$ параметрлерін таңдайды. Шамир хаттамасына көшеміз.

1 қадам. c_1 (10.19) сәйкес $c_1 = 10^7 \bmod 23 = 14$.

2 қадам. c_2 (10.20) сәйкес $c_2 = 14^5 \bmod 23 = 15$.

3 қадам. c_3 (10.21) сәйкес $c_3 = 15^{19} \bmod 23 = 19$.

4 қадам. c_4 (10.22) сәйкес $c_4 = 19^9 \bmod 23 = 10$.

Сонымен, *B* жіберілетін хабарды алады *M = 10*.

10.5. ЭЛЬ-ГАМАЛЬДІҢ КРИПТОГРАФИЯЛЫҚ ЖҮЙЕСІ

A, *B*, *C*, ..., абоненттері бар болсын, олар ешқандай қорғалған арналарды қолданбай бір біріне шифрланған хабарларды жібергісі келеді. Бұл ішкітарауда бұл есепті шешетін *Эль-Гамаль* (*Taher ElGamal*) ұсынған криптографиялық жүйені қарастырамыз, *Шамирдің* криптографиялық жүйесіне қарағанда ол хабарды тек бір жіберуін қолданады. Шынында бұл жерде бір біріне хабар жіберетін екі абонент үшін жалпы құпия кілтті қалыптастыру үшін *Диффи-Хеллман* сұлбасы қолданылады, және содан кейін хабар оны осы кілтке көбейту арқылы шифрланады. Әр келесі хабар үшін құпия кілт қайтадан есептеледі. Криптографиялық жүйенің нақты сипатамасына көшеміз.

Абоненттер тобы үшін кейбір үлкен жай сан *p* және *g* саны таңдалады, *g* әртүрлі дәрежелері - *p* модулі бойынша әртүрлі сандар

(10.2 ішкі тарауды қараңыз). Сандар p және g абоненттерге ашық түрде жіберіледі (оларды желінің барлық абоненттері қолдана алады).

Содан кейін топтың әр абоненті өзінің құпия санын d_i таңдайды, ол келесі талапқа сәйкес болу керек

$$1 < d_i < p–1$$

және оған сәйкес келетін ашық санды e_i есептейді

$$e_i = g^{d_i} \bmod p. \qquad (10.24)$$

Нәижесінде 10.3 кестесін аламыз.

10.3 кесте

Эль-Гамаль жүйесінде пайдаланушылар кілттері

Абонент	Құпия кілт	Ашық кілт
A	d_A	e_A
B	d_B	e_B
C	d_C	e_C

Енді қалай *A* абоненті *M* хабарын *B* абонентіне жіберетінін көрсетеміз. *Шамирдің* криптографиялық жүйесін сипаттаған кездегідей хабар сан түрінде $M < p$ берілді деп есептейміз.

1 қадам. A кездейсоқ сан k қалыптастырады, $1 < k < p–2$, сандарды есептейді

$$r = g^k \bmod p; \qquad (10.25)$$

$$c = (M \cdot e_B^{\,k}) \bmod p \qquad (10.26)$$

және (r, c) сандар жұбын *B* абонентіне жібереді.

2 қадам. B, (r, c) алып, есептейді

$$M' = (c \cdot r^{p-1-d_B}) \bmod p. \qquad (10.27)$$

A абонентінен *B* абонентіне деректерді жіберу кезінде оларды шифрлау процесстерін көрсететін *Эль-Гамаль* криптографиялық жүйесінің сұлбасы 10.3 суретінде көрсетілген.

B абонентінен *A* абонентіне жіберу кезінде деректерді шифрлау процесстерін көрсететін *Эль-Гамальдің* криптографиялық жүйесінің

сұлбасы алдында көрсетілгендей, тек бұл жағдайда хабарды жиберудің инициаторы *B* абоненті болады.

10.6 мақұлдау (*Эль-Гамаль* криптографиялық жүйесінің қасиеттері):

1) *B* абоненті хабарды алды, яғни $M' = M$;

2) қасқой, p, g, e_B, r және c біліп, M есептей алмайды.

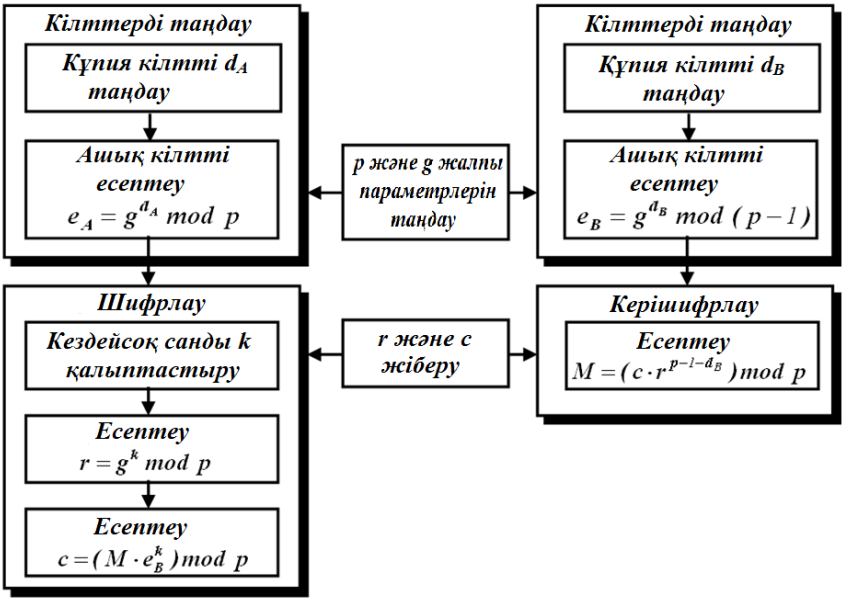
Дәлелдеу. (10.27) өрнегіне (10.26) шыққан c мәнін қоямыз:

$$M' = (M \cdot e_B^{\,k} \cdot r^{p-1-d_B}) \bmod p.$$

Енді r орнына (10.25) қоямыз, ал e_B орнына – (10.24):

$$M' = (M(g^{d_B})^k (g^k)^{p-1-d_B}) \bmod p =$$

$$= (M \cdot g^{d_B \cdot k + k(p-1) - k \cdot d_B}) \bmod p = (M \cdot g^{k(p-1)}) \bmod p.$$



10.3 сурет - *Эль-Гамаль* криптографиялық жүйесінің сұлбасы

Ферма теоремасы бойынша

$$g^{k(p-1)} \bmod p = 1^k \bmod p = 1,$$

және, осылай, мақұлдаудың бірінші бөлімін аламыз.

Екінші бөлімді дәлелдеу үшін қасқой (10.25) теңдікте k есептей алмайды, себебі бұл дискретті логарифмдеу есебі. Сондықтан, ол (10.26) теңдігінде M есептей алмайды, себебі M оған белгісіз санға көбейтілген болатын. Қасқой хабардың заңды қабылдаушысының әрекеттерін де қалпына келтіре алмайды (B абонентінің), себебі оған d_B құпия саны да белгісіз (d_B санын (10.24) негізінде есептеу – дискретті лографмдеу есебі).

10.16 мысал. $M = 15$ хабарын A абонентінен B абонентіне жібереміз. 10.2 мысалында жасалғанға ұқсас параметрлерді таңдаймыз. Аламыз $p = 23$, $g = 5$. B абоненті өзі үшін құпия санды $d_B = 13$ таңдады және (10.24) бойынша есептеді

$$e_B = g^{d_B} \bmod p = 5^{13} \bmod 23 = 21.$$

A абоненті k кездейсоқ санын таңдайды, мысалы $k = 7$, және (10.25), (10.26) бойынша есептейді:

$$r = g^k \bmod p = 5^7 \bmod 23 = 17,$$

$$c = (M \cdot e_B^{\,k}) \bmod p = (15 \cdot 21^7) \bmod 23 = (15 \cdot 10) \bmod 23 = 12.$$

Енді A абоненті B абонентіне шифрланған хабарды $(r, c) = (17, 12)$ сандар жұбы түрінде жібереді.

B абоненті (10.27) бойынша есептейді

$$M' = (c \cdot r^{p-1-d_B}) \bmod p = (12 \cdot 17^{23-1-13}) \bmod 23 =$$

$$= (12 \cdot 17^9) \bmod 23 = (12 \cdot 7) \bmod 23 = 15.$$

Бұдан, B жіберілген хабарды керішифрлағаны көрініп тұр.

Бұндай сұлба арқылы желідегі барлық абоненттер хабарды жібере алатыны түсінікті. B абонентінің ашық кілтін білетін кез келген абонент e_B ашық кілт көмегімен шифрланған хабарларды оған жібере алады. Бірақ, тек B абоненті тек оған белгілі құпия кілтті d_B қолданып сол хабарларды керішифрлай алады. Шифрланған мәтіннің көлемі хабардың көлемінен екі есе үлкен, бірақ тек бір жіберуді талап етеді (тек егер ашық кілттері бар кесте алдын ала барлық абоненттерге белгілі болса).

10.6. RSA КРИПТОГРАФИЯЛЫҚ ЖҮЙЕСІ

RSA криптографиялық жүйесі оны құрушылардың атымен аталған *Ривест*, *Шамир* және *Адлеман*. Бұл криптографиялық жүйе кәзірге дейін ең кең қолданыстағы жүйелердің бірі [17, 18, 24, 35].

Жоғарыда көрсетілгендей, *Шамирдің* криптографиялық жүйесі абоненттер тек ашық байланыс арналарын колданатын жағдайда оқуға жабық хабарлармен алмасу есебін толық шешеді. Бірақ хабар бір абоненттен екіншіге үш рет жіберіледі, бұл кемшілік болады. *Эль-Гамальдің* криптографиялық жүйесі сол есепті деректерді бір рет жіберу арқылы шешеді, бірақ жіберілетін шифрланған деректер көлемі жіберілетін ашық деректер көлемінен екі есе үлкен. *RSA* жүйесінде бұндай кемшіліктер жоқ. Ол басқа біржақты функцияға негізделгені қызықты. Сонымен қатар, *RSA* криптографиялық жүйесінде заманауи криптографияның өнертабысы пайда болады – *«құпысы» бар біржақты функция* (*trapdoor function*).

Бұл жүйе сандар теориясының келесі екі фактісіне негізделген:

- санды жайлыққа тексеру есебі жеңіл деп есептеледі;

- $n = p \cdot q$ (p және q – жай сандар) түрі бар сандарды көбейткіштерге бөлу есебі өте қиын, егер біз тек n білсек, ал p және q – үлкен сандар болса (бұл *факторизациялау есебі* деп аталады).

RSA криптографиялық жүйесінде *A*, *B*, *C*, ... абоненттері болсын. Әр абонент кездейсоқ екі үлкен жай сан p және q таңдайды. Содан кейін ол келесі санды есептейді

$$n = p \cdot q. \tag{10.28}$$

Бұл сан n ашық ақпарат болады, ол барлық абоненттер үшін қолжетімді.

Содан кейін абонент $\varphi(n) = (p-1)\cdot(q-1)$ санын есептейді және кейбір санды таңдайды $e < \varphi(n)$, ол $\varphi(n)$ өзара жай сан болу қажет, және *Евклидтың* жалпылама алгоритмі бойынша d санын табады, ол

$$e \cdot d \bmod \varphi(n) = 1. \tag{10.29}$$

Абоненттермен байланысты және олардың ашық және құпия кілттер болатын барлық ақпарат 10.4 кестесінде көрсетілген.

RSA хаттамасын сипаттаймыз. *A* абоненті *M* хабарын *B* абонентіне жібергісі келсін, *M* хабары $M < n_B$ сәйкес келетін сан

ретінде қарастырылады (ары қарай *B* индексі сәйкес келетін параметрлер *B* абонентіне қатысты екенін көрсетеді).

10.4 кесте

RSA жүйесінде пайдаланушылардың кілттері

Абонент	Ашық кілт	Құпия кілт
A	e_A, n_A	d_A
B	e_B, n_B	d_B
C	e_C, n_C	d_C

1 қадам. A абоненті *B* абонентінің ашық параметрлерін қолданып, хабарды келесі өрнек бойынша шифрлайды

$$C = (M^{e_B}) \bmod n_B, \qquad (10.30)$$

және *e* ашық арнамен жібереді.

2 қадам. Шифрланған хабарды алған *B* абоненті есептейді

$$M' = (C^{d_B}) \bmod n_B. \qquad (10.31)$$

A абонентінен *B* абонентіне деректерді жіберу кезінде оларды шифрлау процесстерін көрсететін *RSA* криптографиялық жүйесінің сұлбасы 10.4 суретінде көрсетілген.

B абонентінен *A* абонентіне жіберу кезінде деректерді шифрлау процесстерін көрсететін *RSA* криптографиялық жүйесінің сұлбасы алдында көрсетілгендей, тек бұл жағдайда хабарды жиберудің инициаторы *B* абоненті болады.

10.7 мақұлдау. Сипатталған хаттама үшін $M' = M$, яғни *B* абоненті *A* абонентінен шыққан хабарды алады.

Дәлелдеу. Хаттаманы құру бойынша

$$M' = (C^{d_B}) \bmod n_B = (M^{e_B \cdot d_B}) \bmod n_B.$$

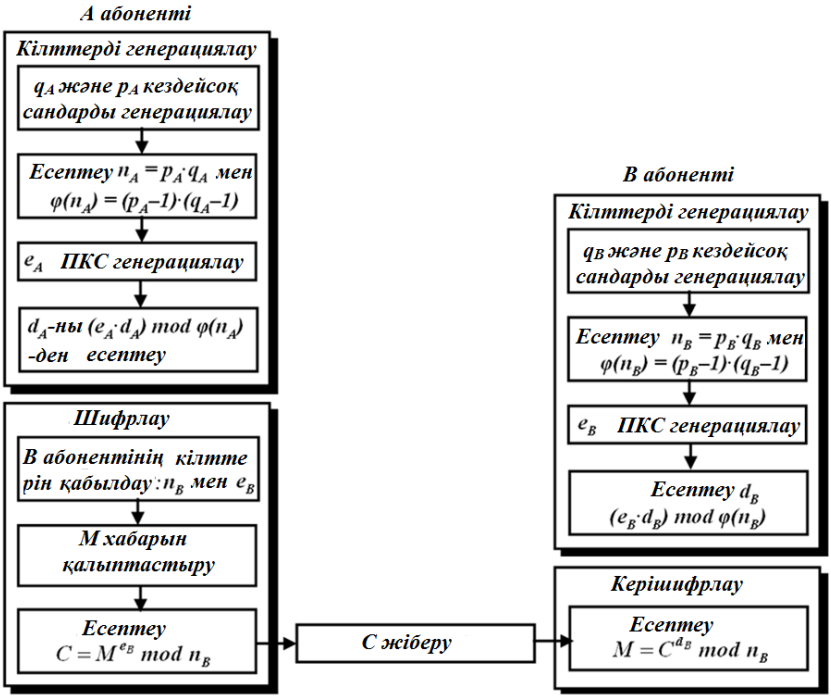
(10.29) теңдігі білдіреді кейбір *k* үшін

$$e_B \cdot d_B = k \cdot \varphi_B + 1.$$

10.5 мақұлдауға сәйкес

$$\varphi_B = (p_B - 1) \cdot (q_B - 1) = \varphi(n_B),$$

бұл жерде $\varphi(n_B)$ – *Эйлер* функциясы.



10.4 сурет - *RSA* криптографиялық жүйесінің сұлбасы

Осыдан және 10.8 теоремасынан шығады

$$M' = (M^{e_B \cdot d_B}) \bmod n_B = (M^{k \cdot \varphi(n_B)+1}) \bmod n_B = M .$$

10.8 мақұлдау (*RSA* хаттаманың қасиеттері).

1) хаттама ақпаратты дұрыс шифрлауды және керішифрлауды қамтамасыз етеді;

2) барлық ашық хабарды білетін және барлық хабарды жолай ұстайтын қасқой үлкен p және q кезінде алғашқы хабарды таба алмайды.

Дәлелдеу. Хаттаманың бірінші қасиеті 10.8 мақұлдаудан шығады. Екінші қасиетті дәлелдеу үшін, қасқой тек ашық параметрлерді n және e білетінін еске сақтаймыз. d табу үшін ол

$\varphi(n) = (p–1)\cdot(q–1)$ білу керек, ал ол үшін оған p және q білу қажет. Жалпы айтқанда, ол p және q таба алады, ол үшін n көбейткіштерге бөлу қажет, бірақ бұл қиын есеп (10.2 ішкі тарауды қараңыз). Үлкен кездейсоқ p және q қажетті уақытта табу мүмкін, себебі 1 тармақ әділет.

RSA жүйесінде қолданатын, біржақты функциясында $y = x^d \bmod n$, егер n жай көбейткіштерге бөлу белгілі болса, кері функцияны $x = \sqrt[d]{y} \cdot \bmod n$ жеңіл есептеуге мүмкіндік беретін *«құпысы»* бар. Шынында, $\varphi(n) = (p–1)\cdot(q–1)$, ал содан кейін $d = e^{-1} \bmod \varphi(n)$ есептеу жеңіл. Егер p және q белгісіз болса, онда кері функция мәнін есептеу мүмкін емес, ал n бойынша p және q табу өте қиын, яғни p және q табу – *«құпыс»* немесе *«құпия жол»*. Осындай құпысы бар біржақты функциялар криптографияның басқа тарауларында да қолданыс табады.

RSA сұлбасы үшін келесі өте маңызды: әр абонент жеке p және q жай сандар жұбын таңдау керек, яғни барлық модулдер $n_A, n_B, n_C, \ldots$ әртүрлі болу қажет (қарсы жағдайда бір абонент басқа абонентке арналған шифрланған хабарларды оқуға мүмкіндік алатын еді). Бұл екінші ашық параметр e талап етілмейді. Параметр e барлық абоненттерде бірдей болу мүмкін. Жиі $e = 3$ таңдауды ұсынылады (p және q сәйкес таңдауында, [7, 17] қараңыз). Сонда шифрлау максималды тез орындалады, барлығы екі көбейтумен.

9.17 мысал. A абоненті *B* абонентіне $M = 15$ хабарды жібергісі келсін. *B* абоненті келесі параметрлерді таңдасын:

$$p_B = 3,\ q_B = 11,\ n_B = 33,\ e_B = 3,$$

e_B өзаражай сан $\varphi(33) = 20$.

Евклидтің жалпылама алгоритмі көмегімен d_B табамыз:

B абонентінің құпия кілті – Евклидтің жалпылама алгоритмі көмегімен $d_B = 7$ болады.

Тексереміз: $e_B \cdot d_B \bmod \varphi(n) = 3\cdot 7 \bmod 20 = 1$.

M (10.30) өрнегі бойынша шифрлаймыз:

$$C = M^{e_B} \bmod n_B = 15^3 \bmod 33 = 15^2 \cdot 15 \bmod 33 =$$

$$= 27\cdot 15 \bmod 33 = 9.$$

A абоненті *C = 9* санын *B* абонентіне ашық байланыс арнасымен жібереді. Тек *B* абоненті $d_B = 7$ біледі, сондықтан ол қабылданған хабарды (10.31) қолданып, керішифрлайды:

$$M' = C^{d_B} \bmod n_B = 9^7 \bmod 33 = (9^2)^2 \cdot 9^2 \cdot 9 \bmod 33 =$$

$$= 15^2 \cdot 15 \cdot 9 \bmod 33 = 15.$$

Сонымен, *B* абоненті *A* абоненттің хабарын керішифрлады.

Қарастырылған сұлба үлкен *p* және *q* үшін ашылмайды, бірақ оның келесі кемшілігі бар: *A* абоненті *B* абонентіне хабарды *B* абонентінің ашық ақпаратын (n_B және c_B сандарын) қолданып жібереді. Қасқой *B* арналған хабарларды оқый алмайды, бірақ ол *A* атынан *B* хабар жіберу мүмкін. Бұны күрделірек хаттамаларды қолдану арқылы болдырмауға болады, мысалы, келесіні.

A абоненті *B* абонентіне *M* хабарын жібергісі келеді. Алдымен *A* абоненті $C = M^{d_A} \bmod n_A$ санын есептейді. Қасқой оны жасай алмайды, себебі d_A құпия. Содан кейін *A* $F = C^{e_B} \bmod n_B$ санын есептейді және *B* жібереді. *B F* қабылдайды және ретімен сандарды есептейді

$$U = F^{d_B} \bmod n_B \text{ және } W = U^{e_A} \bmod n_A.$$

Нәтижесінде *B* абоненті *W* хабарын алады. *RSA* алғашқы сұлбасындағыдай, қасқой жіберілген хабарды оқый алмайды, бірақ бұл жерде ол *A* атынан хабар жібере алмайды (себебі құпия d_A білмейді).

Бұл жерде жаңа жағдай пайда болады. *B* хабар *A* келгенін біледі, яғни *A* хабарды өз құпия d_A шифрлап, *«қол қойған»* сияқты болады. Бұл электронды немесе цифрлық қолтаңба деп аталатынның мысалы. Бұл – заманауи криптографияның тәжірибеде кең қолданатын өнертабыстардың біреуі.

10.7. РАБИННЫҢ КРИПТОГРАФИЯЛЫҚ ЖҮЙЕСІ

Рабинның криптографиялық жүйесінің қауіпсіздігі құрамды санның модулі бойынша шаршы түбірді табудың қиындығына сүйенеді. Сандар теориясы позициясынан – бұл есеп модульді факторизациялау есебіне ұқсас. Жүйенің жүзеге асырудың біреуін

қарастырамыз. *A* және *B* пайдаланушылары бір бірімен шифрланған хабарлармен алмасқысы келді. Онда *Рабин* сұлбасы бойынша:

1. *A* және *B* пайдаланушылары өздері үшін ашық және жабық кілттерді генерациялайды, ол үшін:

- пайдаланушылардың әрқайсысы екі үлкен жай санды таңдайды p_A (p_B) және q_A (q_B) (ары қарай керішифрлау алгоритмі ерекше оңай болады, егер осы сандар *3* санымен *4* модулі бойынша салыстырмалы болса, яғни $p_A \bmod 4 = 3$, $q_A \bmod 4 = 3$, $p_B \bmod 4 = 3$ және $q_B \bmod 4 = 3$). Сандар p_A (p_B) және q_A (q_B) жүйенің жабық кілттері болады (*A* және *B* пайдаланушылардың);
- әр пайдаланушы өзінің ашық кілттерін анықтайды: $n_A = p_A \cdot q_A$ және $n_B = p_B \cdot q_B$;
- *A* және *B* пайдаланушылары ашық кілттермен алмасады.

2. Хабарларды шифрлаудың алдында абоненттер оларды m_i блоктарына бөледі, олардың ұзындықтары ашық кілттен n_A (егер хабар *B* пайдаланушысынан *A* пайдаланушысына жіберілсе) немесе n_B (егер хабар *A* пайдаланушысынан *B* пайдаланушысына жіберілсе) кіші болу қажет. $M < n_A$ немесе $M < n_B$ деп қабылдаймыз.

3. Хабарларды шифрлау үшін *A* немесе *B* абоненттері деректерді шифрлау теңдігіне сәйкес түрлендіреді

$$C = E_n(M) = M^2 \bmod n \qquad (10.32)$$

және содан кейін *C* басқа абоненттке *A* немесе *B* жібереді.

4. (10.32) өрнегінен шығады $M - C$ санынан *n* модулі бойынша шаршы түбір екені. Егер $gcd(C,n) = 1$ (*C* және *n* сандары өзара жай сандар), онда әр шаршылық қалдықта қалдықтардың мультипликативті тобында Z_n^* тұра төрт әртүрлі түбірі болады [7, 24]. Қабылдаушы *n* санның көбейткіштерін *p* және *q* білгендіктен, ол екі салыстыруды шешеді. Ең алдымен ол есептейді

$$\begin{aligned} r_1 &= C^{(p+1)/4} \bmod p; \quad r_2 = -C^{(p+1)/4} \bmod p; \\ r_3 &= C^{(q+1)/4} \bmod q; \quad r_4 = -C^{(q+1)/4} \bmod q; \end{aligned} \qquad (10.33)$$

$$a = q \cdot (q^{-1} \bmod p); \quad b = p \cdot (p^{-1} \bmod q). \qquad (10.34)$$

C санынан *n* модулі бойынша төрт мүмкін түбір – бұлар

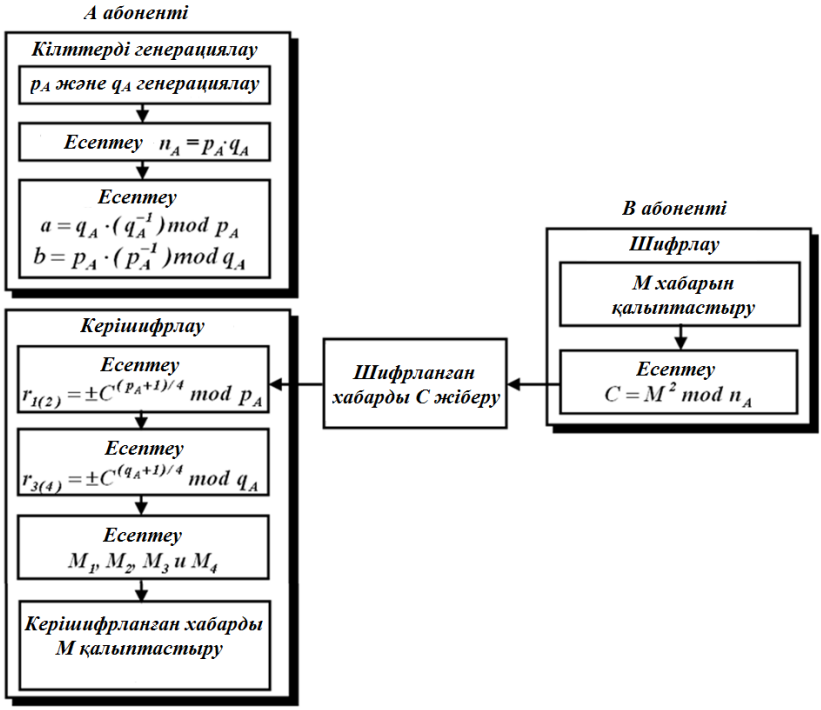
$$M_1 = (a \cdot r_1 + b \cdot r_3) \bmod n; \quad M_2 = (a \cdot r_2 + b \cdot r_4) \bmod n; \qquad (10.35)$$

$$M_3 = (a \cdot r_1 + b \cdot r_4) \bmod n; \quad M_4 = (a \cdot r_2 + b \cdot r_3) \bmod n.$$

A абонентінен *B* абонентіне деректерді жіберу кезінде оларды шифрлау процесстерін көрсететін *Рабин* криптографиялық жүйесінің сұлбасы 10.5 суретінде көрсетілген.

A абонентінен *B* абонентіне жіберу кезінде деректерді шифрлау процесстерін көрсететін *RSA* криптографиялық жүйесінің сұлбасы алдында көрсетілгендей, тек бұл жағдайда хабарды жиберудің инициаторы *A* абоненті болады.

Рабинның криптографиялық жүйесінде шифрлайтын айқындау инъективті емес (өзара бірмәнді емес) екені көрініп тұр. Барлық төрт түбірді есептегеннен кейін олардың ішінен жіберілетін хабардың сандық эквиваленті болатын сан таңдалады. Егер хабар әдеттегі тілмен жазылса, онда дұрыс *M* таңдау қиын емес. Егер кездейсоқ биттер жинағы шифрланса, онда дұрыс *M* анықтау тәсілі жоқ.



10.5 сурет – *Рабинның* криптографиялық жүйесінің сұлбасы

10.18 мысал. *B* пайдаланушыдан *A* пайдаланушыға жіберу үшін кілттерді генерациялау және "*протокол*" хабарын *Рабинның* криптографиялық жүйесі көмегімен шифрлау.

Шешім. *A* пайдаланушының жабық кілттерін $p_A = 79$ және $q_A = 83$ жай сандар жұбы құрсын, онда ашық кілт - $n_A = p_A \cdot q_A = 6557$ саны. Ашық хабардың әр әріпін оның орыс алфавитіндегі нөмірімен ауыстырамыз

$$\text{"}протокол\text{"} => 15\ \ 16\ \ 14\ \ 18\ \ 14\ \ 10\ \ 14\ \ 11.$$

Келтірілген сандық тізбек – цифрлық түрде жазылған ашық хабар. Тізбекті төрт блокқа m_i бөлеміз, оның әрқайсысы – кейбір натуралды сан, ол $n_A = 6557$ санынан кіші болу қажет:

$$M => 1516 - 1418 - 1410 - 1411.$$

Ашық мәтін *M* (10.32) өрнегі бойынша шифрлаймыз:

$$C_1 = E_{n_A}(m_1) \bmod n_A = 1516^2 \bmod 6557 = 3306;$$

$$C_2 = E_{n_A}(m_2) \bmod n_A = 1418^2 \bmod 6557 = 4282;$$

$$C_3 = E_{n_A}(m_3) \bmod n_A = 1410^2 \bmod 6557 = 1329;$$

$$C_4 = E_{n_A}(m_4) \bmod n_A = 1411^2 \bmod 6557 = 4150.$$

Нәтижесінде шифрланған хабарды аламыз

$$C => 3306 - 4282 - 1329 - 4150.$$

10.19 мысал. *A* пайдаланушысы *Рабинның* криптографиялық жүйесінде өзінің кілттер жұбын генерациялады: жабық кілт - $p_A = 59$ және $q_A = 67$ жай сандар, оларды құпияда сақтайды, және ашық кілт – жалпы қолжетімді сан $n_A = p_A \cdot q_A = 3953$. *B* пайдаланушысынан алған $C => 1108 - 1756$ хабарын керішифрлау.

Шешім. Керішифрлау – модуль $n_A = 3953$ бойынша $C_1 = 1108$ және $C_2 = 1756$ сандардан шаршы түбірді алу. Ол үшін жай модульдермен $p_A = 59$ және $q_A = 67$ түбірлерді алу қажет.

Есептейміз

$$r_1 = C_1^{(p_A+1)/4} \bmod p_A = 1108^{(59+1)/4} \bmod 59 = 20;$$

$$r_2 = -C_1^{(p_A+1)/4} \bmod p_A = -1108^{(59+1)/4} \bmod 59 = -20 \bmod 79 = 39;$$

$$r_3 = C_1^{(q_A+1)/4} \bmod q_A = 1108^{(67+1)/4} \bmod 67 = 6;$$

$$r_4 = -C_1^{(q_A+1)/4} \bmod q_A = -1108^{(67+1)/4} \bmod 67 = -6 \bmod 79 = 61.$$

Алынған шешімдерді өзара жүйеге құрамдастырамыз және қалдықтар туралы қытай теоремасы бойынша шаршы түбірлер мәндерін анықтаймыз. $gcd(p_A,q_A) = 1$ және *Евклидтың* кеңейтілген теоремасы бойынша:

$$1 = 37 \cdot 67 + 25 \cdot 59,$$

яғни $(q_A^{-1} \cdot q_A) \bmod p_A = 1$, бұл жерден $q_A^{-1} = 37$; $(p_A^{-1} \cdot p_A) \bmod q_A = 1$, одан $p_A^{-1} = 25$.

q_A, q_A^{-1}, p_A және p_A^{-1} мәндеріне сәйкес:

$$a = q_A(q_A^{-1} \bmod p_A) = 67 \cdot 37 = 2479;$$

$$b = p_A(p_A^{-1} \bmod q_A) = 59 \cdot 25 = 1475.$$

M бірінші түбірін (10.33), (10.34) және (10.35) табамыз. Одан шығады:

$$M_1 = (a \cdot r_1 + b \cdot r_3) \bmod n_A = 3088.$$

Екінші түбір табылған бірінші түбірге қарсы $M_1 = 3088$, оны (10.33), (10.34) және (10.35) табамыз. Одан шығады:

$$M_2 = (a \cdot r_2 + b \cdot r_4) \bmod n_A = 0865.$$

M_1 және M_2 түбірлері қарсы екеніне көз жеткізуге болады, яғни $(M_1 + M_2) \bmod n_A = 0$:

$$(M_1 + M_2) \bmod n_A = (3088 + 0865) \bmod 3953 = 3953 \bmod 3953 = 0.$$

Үшінші түбірді M (10.33) және (10.34) табамыз. Одан шығады:

$$M_3 = (a \cdot r_1 + b \cdot r_4) \bmod n_A = 1200.$$

Төртінші түбір табылған үшінші түбірге қарсы $M_3 = 1200$, оны (10.33) және (10.34) табамыз. Одан шығады:

$$M_4 = (a \cdot r_2 + b \cdot r_3) \bmod n_A = 2753.$$

M_3 және M_4 түбірлері қарсы екеніне көз жеткізуге болады, яғни $(M_3 + M_4) \bmod n_A = 0$:

$$(M_3 + M_4) \bmod n_A = (1200 + 2753) \bmod 3953 = 3953 \bmod 3953 = 0.$$

$C_1 = 1108$ санынан 3953 модулі бойынша төрт түбір алынды: $3088, 0865, 1200$ және 2753. Тұра солай шифрланған хабардың екінші саннан $C_2 = 1756$ төрт түбірді аламыз. Олар: $1215, 2738, 2336, 1617$. Бұл жағдайда $1200, 1215$ және 1617 түбірлері – 32 әріпі бар орыс алфавиттағы хабардың сандық эквиваленттері.

Орыс алфавиттың 12 санына "*м*" әріпі сәйкес келетіндіктен, 00 – "*а*", 15 – "*п*", 16 – "*р*", 17 – "*с*", онда керішифрланған хабардың бірінші екі әріпі "*ма*", ал екінші екі әріпі *мп*", немесе "*рс*" болады. Сондықтан керішифрланған хабар немесе "*мамп*", немесе "*марс*" сәйкес келеді. Орыс тілінде мәні бар тек "*марс*" сөзінде.

10.8. ДИСКРЕТТІК ЛОГАРИФМДЕУГЕ НЕГІЗДЕЛГЕН КРИПТОГРАФИЯЛЫҚ ЖҮЙЕЛЕРДІ БҰЗУ ӘДІСТЕРІ

10.8.1. Есепті қою

Сенімді криптографиялық жүйені құру үшін қасқой қолдана алатын бұзу әдістерін назарға алу керек және бұл әдістер жүзеге асырылмайтын болатындай криптографиялық жүйенің параметрлерін (мысалы, сандар көлемі) таңдау қажет. Берілген тарауда сондай екі әдісті қарастырамыз.

Жоғарыда айтылғандай, көптеген криптографиялық жүйелер біржақты функцияға негізделеді

$$y = a^x \bmod p. \qquad (10.36)$$

Егер a, x және p берілсе, $2 \cdot \log x$ операция ішінде (10.1 мақұлдау) y есептеуге болады. Бірақ белгілі a, y және p бойынша x табу, яғни дискретті логарифмды есептеу, - өте қиын есеп.

Шамирдың криптожүйесін қарастыру кезінде көрсетілгендей ((10.23) қараңыз), *Ферма* теоремасы негізінде қарапайым p модулі

бойынша дәрежені шығару кезінде дәреже көрсеткіштері *p–1* модулі бойынша келтіріледі. Сондықтан тек $0 \le x \le p-1$ сәйкес келетін *x* көрсеткіштерін қарастыру жеткілікті.

(10.1) *a*, *x* және *p* бойынша *y* есептеу үшін қажетті көбейту операциялардың санын t_y деп белгілейміз және қысқалық үшін t_y есептеу уақыты деп атаймыз. 10.1 тараудағы алгоритмдер бойынша дәрежеге шығару уақыты $2 \cdot \log x$ көп емес, бұл жерде $x < p$. Осыдан

$$t_y \le 2 \cdot \log x \qquad (10.37)$$

x дәреженің кез келген көрсеткішінде.

Енді берілген *a*, *y* және *p* бойынша (10.36) *x* табу есебіне көшеміз. Алдымен түзу талдау қиындығын бағалаймыз. Ол үшін алдымен a^1 есептеп және $a^1 \bmod p = y$ дұрыстығын тексереміз. Егер жоқ болса, онда $a^2 \bmod p = y$ тексереміз, егер жок болса, онда $a^3 \bmod p = y$ тексереміз және ары қарай $a^{p-1} \bmod p = y$ дейін. Орташа есеппен $(p-1)/2$ рет *a* көбейту және теңдікті тексеру қажет болады. Сонымен, түзу талдау уақыты

$$t_{п.п.} \approx p/2.$$

Төменде қарастырылатын *«сәби қадамы, дәу қадамы»* әдісінде *x* табу уақыты шамамен азырақ

$$t_{ш.м\ ш.в.} \approx 2 \cdot \sqrt{p}\ ,$$

ал дәрежені шығару әдісінде бұл уақыт оданда аз:

$$t_{и.п.} \approx c_1 \cdot 2^{c_2 \sqrt{\log p \cdot \log(\log p)}}\ ,$$

бұл жерде c_1, c_2 – кейбір оң тұрақтылар.

Салыстыруды көрнектірек жасау үшін есептеу уақытын (10.36) *p* санының ұзындығы арқылы көрсетеміз. Бұл ұзындықты биттермен *n* деп белгілейміз. *p* модулі бойынша есептеулерде $n \approx \log p$ болады. Сондықтан алгоритмдердің еңбеккөлемділігі (операциялар саны мәнінде) келесі болады:

$$t_y \approx n, \quad t_{п.п.} \approx 2^{n-1}, \quad t_{ш.м\ ш.в.} \approx 2^{n/2}, \quad t_{и.п.} \approx 2^{c_2 \sqrt{n \cdot \log n}}\ ,$$

бұл жерде ≈ “*пропорционалды*” дегенді білдіреді.

Көріп отырғандай, дәрежеге шығару кезінде операциялар саны *n* санының ұзындығымен сызықты өседі, ал қайтымды есепті әртүрлі

әдістермен шешу экспоненциалды немесе субэкспоненциалды (дәрежені санау әдісі үшін.) өседі. Дискретті логарифмдерді есептеу, және де криптографиялық талдауда пайда болатын басқа қайтымды есептерді шығарудың жылдам алгоритмдердің бар болуы туралы сұрақ ашық қалды.

10.8.2. "Сәби қадамы, дәу қадамы" әдісі

Ашық әдебиетте бұл әдіс бірінші рет *Д. Шенкпен* (*Daniel Shanks*) сипатталған; оған деген сілтемелер 1973 жылдан белгілі [20]. Бұл әдіс дискретті логарифмды есептеу есебі талдау әдісіне қарағанда тез шешілу мүмкін екенін көрсеткен бірінші әдістердің бірі. (10.36) *x* табудың осы әдісін сипаттаймыз.

1 қадам. Алдымен келесі өрнек орындалатын екі бүтін санды *m* және *k* аламыз

$$m \cdot k > p. \tag{10.38}$$

2 қадам. Екі сандар қатарын есептейміз

$$y,\ a \cdot y,\ a^2 \cdot y,\ \ldots,\ a^{m-1} \cdot y \bmod p; \tag{10.39}$$

$$a^m,\ a^{2 \cdot m},\ \ldots,\ a^{k \cdot m} \bmod p. \tag{10.40}$$

Барлық есептеулер *p* модулі бойынша орындалады.

3 қадам. Келесі теңдік орындалатын *i* және *j* табамыз

$$a^{i \cdot m} = a^j \cdot y. \tag{10.41}$$

10.9 мақұлдау. Сан

$$x = i \cdot m - j \tag{10.42}$$

(10.36) теңдігінің шешімі болады. Осыған қоса, (10.41) сәйкес келетін бүтін сандар *i* және *j* бар.

Дәлелдеу. (10.42) әділеттігі төменде келтірілген теңдеулер тізімінен шығады, бұл жерде барлық есептеулер *p* модулімен берілген, ал бөлу қайтымды элементке көбейтуге сәйкес келеді:

$$a^x = a^{i\ m-j} = \frac{a^{i\ m}}{a^j} = \frac{a^{i\ m}y}{a^j y} = \frac{a^{i\ m}y}{a^{i\ m}} = y\,.$$

Енді (10.41) сәйкес келетін i және j сандары бар екенін дәлелдейміз. Ол үшін (10.42) түрі бар барлық сандарды 10.5 кестесіне жазамыз.

Кестеде 1 ден $k \cdot m$ дейін барлық сандар бар екенін көреміз. Сондықтан, (10.42) кестеде 1 ден p дейін барлық сандар болатыны шығады. Сонымен, дәреженің кез келген көрсеткіші $x < p$ кестеде болады, яғни (10.36) сәйкес келетін x саны (10.42) түрінде көрсетілу мүмкін және кестеде әрқашан болады, сондықтан (10.41) теңдігінде әрқашан шешім болады.

10.5 кесте

$i \cdot m - j$ түрі бар сандарды тарату

	$j = 0$	$j = 1$	$j = 2$	...	$j = m - 1$
$i = 1$	m	$m - 1$	$m - 2$	...	1
$i = 2$	$2 \cdot m$	$2 \cdot m - 1$	$2 \cdot m - 2$	...	$m + 1$
...	...	...	...	...	...
$i = k$	$k \cdot m$	$k \cdot m - 1$	$k \cdot m - 2$	...	$(k - 1) \cdot m + 1$

10.20 мысал. "*Сәби қадамы, дәу қадамы*" әдісін қолданып, $2^x \bmod 23 = 9$ теңдіктін шешімін табамыз.

m және k таңдаймыз. $m = 6$ және $k = 4$ болсын. (10.38) орындалатынын көріп тұрмыз. (10.39) және (10.40) сандарын есептейміз.

$9, 18, 13, 3, 6, 12$; 18.

Әрі қарай есептеуді жүргізбейміз, себебі $i = 1$, $j = 1$ кезінде (10.39) және (10.40) бірдей сандар табылды. (10.42) бойынша аламыз

$$x = i \cdot m - j = 1 \cdot 6 - 1 = 5.$$

Тексереміз: $2^5 \bmod 23 = 9$. Шыныменде, $x = 5$ шешім болады.

Қарастырған әдістің атының пайда болуын түсіндіреміз. Криптографияда p – үлкен жай сан екені белгілі, сондықтан m және k үлкен. (10.39) қатарында дәреже 1 үлкейеді (сәбі қадамы), ал (10.40) қатарында дәреже m үлкейеді (дәу қадамы). Бұл әдістің қиындығын бағалаймыз.

10.10 мақұлдау. Берілген әдіс бойынша үлкен p кезінде есептеулер уақыты келесіге сәйкес келеді [34]

$$t_{ш.м.ш.в.} \le const \cdot \sqrt{p} \cdot log^2\ p\,. \qquad (10.43)$$

Бұл жерде толық есептеу уақыты, көбейту саны емес.

Дәлелдеу. Алуға болады

$$k = m = \left\lfloor \sqrt{p} \right\rfloor + 1\,, \qquad (10.44)$$

сонда (10.38) орындалады. Онда (10.39) және (10.40) $2 \cdot \sqrt{p}$ көп емес көбейту операциялары қажет болады. *Әдетті (мектептегі)* көбейту және бөлу әдістері үшін екі *r*-белгісі бар сан үшін нәтижені есептеу уақыты r^2 пропорционалды екені белгілі. Бізде барлық сандар *{ 1, ... ,p }* көптігінен алынады, демек, $r < log\ p$, және есептеу уақыты $log^2 p$ пропорционалды. Осыдан (10.39) және (10.40) қатарларын есептеуге кеткен уақытты аламыз. Бірақ алгоритмнің барлық кезеңдері және осы қатарларда бірдей сандарды шешу үшін қажетті уақыт есепке алынбаған. Үлкен *k* және *m* үшін бұл есеп оңай емес. Ол келесідей шешілу мүмкін: алдымен әр санға қатардағы нөмірін береміз және тағы бір битты, оған ол қай қатарға жатқанын жазамыз (10.39) немесе (10.40), содан кейін екі тізбекті тізімге түрлендіреміз және сұрыптаймыз (шама бойынша реттейміз). Жалпы қатардың ұзындығы $k + m \approx 2 \cdot \sqrt{p}$. Ең жақсы сұрыптау әдістері үшін $S \cdot log\ S$ салыстыру операциялары қажет, бұл жерде *S* – тізімдегі элементтер саны (мысалы, [34] қараңыз). Біздің жағдайда $S = 2 \cdot \sqrt{p}$ және

$$2\sqrt{p} \cdot log(\,2 \cdot \sqrt{p}\,) \approx \sqrt{p} \cdot log\ p$$

салыстыру операциялары қажет $log\ p$ бит ұзындығы бар сөздер үшін, яғни барлығы шамамен $\sqrt{p} \cdot log^2\ p$ операция қажет. Біріктірілген қатарды сұрыптаудан кейін әртүрлі қатарлардан (10.39), (10.40) биттік белгіні қолданып екі бірдей санды табу қажет. Сонымен, барлық кезеңдерде есептеу уақытын қосып, (10.43) аламыз.

10.8.3. Дәрежені есептеу алгоритмі

Дәрежені есептеу алгоритмінің (*index-calculus algorithm*) негізгі идеялары сандар теориясында XX ғасырдың 20-ші жылдарынан бері белгілі. Бірақ, *RSA* құрушылардың бірі, *Адлеман* тек 1979 жылы осы алгоритмге (10.36) шешу құралы ретінде

көрсетті және оның жұмыскөлемділігін зерттеді. Кәзіргі уақытта дәрежені есептеу алгоритмі және оның жақсартылған нұсқалары (10.36) сияқты теңдіктерде дискретті логарифмдерді есептеудің ең тез тәсілін береді.

Алгоритмді сипаттау үшін келесі түсінікті енгіземіз.

10.7 анықтама. Сан n p-тегіс деп аталады, егер ол тек p тең немесе кіші жай көбейткіштерге жіктелсе.

10.21 мысал. 15, 36, 45, 270, 2025 сандары *5*-тегіс (оларды жіктеуде тек *2, 3* және *5* көбейткіштері бар) деп аталады.

Енді тікелей алгоритмді сипаттауға көшеміз.

1 қадам. Бірінші t жай сандардан тұратын базалық көбейткіштер көптігін қалыптастырамыз

$$S = \{ p_1, p_2, \ldots, p_t \},$$

(t мәнін таңдау туралы ескерту төменде кілтіріледі).

2 қадам. Тізбекті түрде $k = 1,2,3,\ldots$, мәндерін беріп, $a^k \bmod p$ түрі бар $t + \varepsilon$ (ε – үлкен емес бүтін сан, төменде қараңыз) p^t-тегіс сандарын табамыз, тегістікті S көптігінің элементтеріне бөлу жолымен тексереміз. Табылған p^t-тегіс сандардың әрқайсысы базалық көбейткіштерді көбейту арқылы жазылады:

$$a^k \bmod p = \prod_{i=1}^{t} p_i^{c_i}, \quad c_i \geq 0. \tag{10.45}$$

Әр k мәні үшін өз c_i сандар жинағын аламыз.

3 қадам. 2 қадамда табылған әр p^t-тегіс сан үшін (10.45) логарифмерге көшеміз:

$$k = \sum_{i=1}^{t} c_i \cdot \log_a p_i . \tag{10.46}$$

(10.46) түрі бар t белгісіздігі бар $t+\varepsilon$ теңдіктен тұратын жүйе алынды. Белгісіз ретінде $\log_a p_i$ шамалары болады, теңдіктер саны белгісіздік санынан ε үлкен, бұл егер теңдіктердің кейбіреуі сызықты тәуелді болған жағдайда, жүйені шешу ықтималдығы жоғарлайды. Барлық есептеулерді *p–1* модулі бойынша жүргізіп жүйені сызықты алгебра әдістерімен шешеміз (дәреже көрсеткіштері және логарифмдер *p–1* модулі бойынша келтірілетінін еске саламыз). Нәтижесінде S көптігінен сандардың логарифмдер мәндерін аламыз: $\log_a p_1, \log_a p_2, \ldots, \log_a p_t$.

4 қадам. Кездейсоқ түрде r таңдап, $(y \cdot a^r)$ түрі бар p^t-тегіс санды табамыз:

$$y \cdot a^r \bmod p = \prod_{i=1}^{t} p_i^{\varepsilon_i}, \quad \varepsilon_i \geq 0. \tag{10.47}$$

5 қадам. (10.47) логарифмдеп, соңғы нәтижені аламыз

$$x = \log_a y = \left(\sum_{i=1}^{t} e_i \log_a p_i - r \right) \bmod (p-1), \tag{10.48}$$

r шамасы барлық қосындыдан алынады.

Сипатталған әдістің әділдігі алгоритмді құрастырудан көрінеді, ал оның тиімділігі келесі бақылаумен байланысты. Егер кездейсоқ түрде шексіз бүтін сандар көптігінен санды таңдаса, онда *1/2* ықтималдығымен ол *2* бөлінеді, *1/3* ықтималдығымен – *3*, *1/5* ықтималдығымен – *5* және ары қарай. Сондықтан, *1* ден $p-1$ дейін арақашықтықта S көптігінен тек кішкентай жай көбейткіштер қатысатын жіктеулері бар жеткілікті көп сан бар. Осы сандар алгоритмнің 2 және 4 қадамдарында ізделеді. Сан t көп болған сайын, яғни S жай көбейткіштер саны, 2 және 4 қадамдарда тегіс сандарды іздеу кезінде сәтсіздіктер аз болады, яғни бұл қадамдар тезірек орындалады. Бірақ үлкен t кезінде 3 қадамның жұмыскөлемділігі шұғыл жоғарлайды, $t+\varepsilon$ теңдіктерден тұратын жүйені шешу кезінде. Минималды жалпы есептеу уақытын беретін мәнді табу әдетте сандық әдістерді қолданумен орындалу мүмкін. Аналитикалық өрнектерді табу жеткілікті қиын. 3 қадамда теңдіктер жүйесінің шешімінің болу ықтималдығын жоғарлату үшін ε параметрі үлкен емес бүтін санға тең болып қабылданады. Алынған жүйеде сызықты тәуелді теңдіктер болу мүмкін (төменде келтірілетін мысалдағыдай). Үлкен p кезінде ε мәні *10* жоғары ықтималдықпен жүйенің жалғыз шешімі болатынын кепілдейді ([7, 34] қараңыз).

Егер алынған жүйеде шексіз көп шешім болса, онда 2 қадамға қайтып келіп k басқа мәндерін қолдану қажет.

Адлеман көрсетті, t оңтайлы мәндері кезінде алгоритмнің жұмыскөлемдігі үшін

$$t_{и.п.} < c_1 \cdot 2^{(c_2+o(1))\cdot\sqrt{\log p \cdot \log(\log p)}},$$

бұл жерде c_1, c_2 – кейбір оң тұрақтылар.

10.22 шешім. Дәрежені есептеу алгоритмі көмегімен теңдікті шешеміз

$$37 = 10^x \bmod 47. \tag{10.49}$$

Берілгені $y = 37$, $a = 10$, $p = 47$. Базалық көбейткіштер көптігін аламыз $S = \{ 2, 3, 5 \}$, $t = 3$, және $\varepsilon = 1$ деп қабылдаймыз, яғни төрт теңдіктен тұратын жүйені құрамыз. S сандардың логарифмдерін u_1, u_2 және u_3 деп белгілейміз, мысалы, $u_3 = \log_{10} 5 \bmod 47$. Алгоритмнің бірінші қадамы орындалды, екіншіге көшеміз.

Енді төрт 5-тегіс сандарын табамыз:

$$\begin{aligned}
10^1 \bmod 47 &= 10 = 2\cdot 5, & \vee\\
10^2 \bmod 47 &= 6 = 2\cdot 3, & \vee\\
10^3 \bmod 47 &= 13 = 13, &\\
10^4 \bmod 47 &= 36 = 2\cdot 2\cdot 3\cdot 3, & \vee\\
10^5 \bmod 47 &= 31 = 31, &\\
10^6 \bmod 47 &= 28 = 2\cdot 2\cdot 7, &\\
10^7 \bmod 47 &= 45 = 3\cdot 3\cdot 5. & \vee
\end{aligned}$$

Төрт 5-тегіс сан табылды, олар *1*, *2*, *4* және *7* дәрежелерге сәйкес келеді.

Алгоритмнің үшінші қадамын бастаймыз. Логарифмдерге көшеміз және алдынғы қадамда $\vee$ символымен белгіленген теңдіктерден жүйе құрамыз:

$$1 = u_1 + u_3, \tag{10.50}$$

$$2 = u_1 + u_2, \tag{10.51}$$

$$4 = 2\cdot u_1 + 2\cdot u_2, \tag{10.52}$$

$$7 = 2\cdot u_2 + u_3. \tag{10.53}$$

Алынған жүйеде (10.51) және (10.52) теңдіктері сызықты тәуелді екені көрініп тұр, сонымен бір төртінші тегіс санды тапқанымыз бекер емес. Жүйені шешу үшін, (10.51) аламыз (10.50). Нәтижеде аламыз

$$1 = u_2 - u_3. \tag{10.54}$$

(10.54) қосамыз (10.53). Нәтижесінде аламыз

$$8 = 3 \cdot u_2 - u_3. \qquad (10.55)$$

(10.50) тікелей u_2 табамыз:

$$u_2 = (8/3) \bmod 46 = 8 \cdot 3^{-1} \bmod 46 = 8 \cdot 31 \bmod 46 = 18.$$

Тексеруге болады, $10^{18} \bmod 47 = 3$ есептеп, сонымен, $u_2 - 3$ санының логарифмы. Енді (10.54) u_3 табамыз:

$$u_3 = u_2 - 1 = 18 - 1 = 17.$$

Тексереміз $10^{17} \bmod 47 = 5$.

(10.51) u_1 табамыз:

$$u_1 = 2 - u_2 = (2 - 18) \bmod 46 = -16 \bmod 46 = 30, \; 10^{30} \bmod 47 = 2.$$

Енді S сандардың логарифмдері белгілі. Алгоритмнің ең жұмыскөлемді кезеңі артта. Төртінші қадамға көшеміз. $k = 3$ бастаймыз:

$$37 \cdot 10^3 \bmod 47 = 37 \cdot 13 \bmod 47 = 11,$$
$$37 \cdot 10^4 \bmod 47 = 37 \cdot 36 \bmod 47 = 16 = 2 \cdot 2 \cdot 2 \cdot 2. \quad \blacktriangledown$$

Соңғы теңдікте логарифмге көшеміз (бұл бесінші қадам) және соңғы нәтижені аламыз:

$$\log_{10} 37 = 4 \log_{10} 2 - 4 = (4 \cdot 30 - 4) \bmod 46 = 24.$$

Сонымен (10.49) теңдіктің шешімі табылды $x = 24$. Тексеруге болады: $10^{24} \bmod 47 = 37$.

Берілген уақытқа ең жылдам деп қарастырылған дәрежені есептеу алгоритмнің нұсқасы есептеледі, ол *Number Field Sieve* деп аталады. *Number Field Sieve* (*NFS* – сандық өрістің шешудің жалпы әдісі — бүтін сандарды факторизациялау әдісі). Бұл әдіс жұқа алгебралық құрылымдарды қолданады және сипаттауға жеткілікті қиын. Оның жұмыскөлемдігі келесі бағамен беріледі

$$t_{u.n.} < c_1 \cdot 2^{(c_2 + o(1)) \sqrt[3]{\log p \cdot \log(\log p)^2}}, \qquad (10.56)$$

бұл жерде c_1 және c_2 – кейбір оң тұрақтылар.

Осы әдіс бүгінгі күні беріктілігі дискретті логарифмдерді (*Дифф-Хеллман*, *Шамир* және *Эль-Гамаль* жүйелері) есептеу қиындығында негізделген криптожүйелердің модульдер ұзындығын

таңдау үшін шарттарды мәжбүрлейді. Бұл криптожүйелердің көпуақытты беріктілігіне жету үшін модульдердің ұзындығын *1024* биттен жоғары алу ұсынылады [7, 20, 43].

Оқулықта сандарды факторизациялауға негізделген криптографиялық жүйелерді (*RSA* сияқты) бұзу әдістері қарастырылмайды. Бүгінгі күнге [7, 34] көбейткіштерге сандарды жіктеудің ең жылдам әдістері (10.56) өрнегі беретін уақыт бағасымен сипатталады. *RSA* жүйесінің беріктілігін қамтамасыз ету үшін модуль ұзындығы *1024* биттен аз болмау керек (яғни *RSA* модулін беретін жай сандар ең кемінде *512* биттен болу керек).

Бақылау сұрақтары және тапсырмалары

1. Жай және құрамды санның анықтамасын беріңіз. Оларға үш мысалдан келтіріңіз.
2. *"Өзаражай сан"* түсінігіне анықтама беріңіз. Өзаражай сандарға және өзаражай емес сандарға мысалдар келтіріңіз. Модуль *n* бойынша инверсия деген не?
3. Факторизациялау есебі деген не? Ең үлкен жалпы бөлгішке анықтама беріңіз.
4. *RSA* алгоритмі қандай мақсаттарға қолданылу мүмкін?
5. *RSA* алгоритмін қолданумен шифрлау үрдісін сипаттаңыз.
6. *Диффи-Хеллман* алгоритмі не үшін қолданылу мүмкін?
7. *Диффи-Хеллман* алгоритмін қолдану кезінде әрекеттер ретін сипаттаңыз.
8. *Эль-Гамаль* алгоритмі қандай мақсаттарға қолданылу мүмкін?
9. *Эль-Гамаль* алгоритмін қолдану кезінде әрекеттер ретін сипаттаңыз.
10. *Рабин* алгоритмі қандай мақсаттарға қолданылу мүмкін?
11. *Рабин* алгоритмін қолдану кезінде әрекеттер ретін сипаттаңыз.
12. Ашық кілті бар шифрлау алгоритмдерін қолдану кезінде қандай шабуылдар болу мүмкін?
13. Келесі өрнектердің нәтижелерін шығару *5*, *16*, *27*, *–4*, *–13*, *3 + 8*, *3 – 8*, *3·8* және *3·8·5*: а) *10* модулі бойынша; б) *11* модулі бойынша.
14. Дәрежені шығарудың жылдам алгоритмдерін қолданып, есептеу: $2^8 \bmod 10$, $2^7 \bmod 10$, $7^{19} \bmod 100$ және $7^{57} \bmod 100$.

15. Жай көбейткіштерге келесі сандарды жіктеу: *108*, *77*, *65*, *30* және *159*.

16. Келесі сандар жұптарынан (*25,12*), (*25,15*), (*13,39*) және (*40,27*) өзаражайларын табыңыз.

17. *Эйлер* функциясының мәнін табыңыз $\varphi(14)$ және $\varphi(15)$.

18. *Эйлер* функцияларының қасиеттерін қолданып, есептеңіз $\varphi(53)$, $\varphi(21)$ және $\varphi(159)$.

19. *Ферма* теоремасын қолданып $3^{13} \bmod 13$, $5^{22} \bmod 11$ және $3^{17} \bmod 5$ есептеңіз.

20. *Эйлер* теоремасын қолданып $3^{9} \bmod 20$, $2^{14} \bmod 21$ және $2^{107} \bmod 159$ есептеңіз.

21. *Евклид* алгоритмі көмегімен $gcd(21,12)$, $gcd(30,12)$, $gcd(24,40)$ және $gcd(33,16)$ табыңыз.

22. Жалпылама *Евклид* алгоритмі көмегімен x және y мәндерін келесі теңдіктерде табыңыз:

а) $21 \cdot x + 12 \cdot y = gcd(21,12)$; б) $30 \cdot x + 12 \cdot y = gcd(30,12)$;

в) $24 \cdot x + 40 \cdot y = gcd(24,40)$; г) $33 \cdot x + 16 \cdot y = gcd(33,16)$.

23. $3^{-1} \bmod 7$, $5^{-1} \bmod 8$, $3^{-1} \bmod 53$ және $10^{-1} \bmod 53$ есептеу.

24. *100* кіші барлық жай сандарды жазыңыз. Олардың қайсысы $p = 2 \cdot q + 1$ түрге сәйкес келеді, бұл жерде q жай сан?

25. *Диффи-Хеллман* жүйесінде $p = 11$ болғанда g параметрін таңдаудың барлық мүмкін нұсқаларын табу.

26. Келесі параметрлері бар *Диффи-Хеллман* криптографиялық жүйесі үшін ашық кілттерді e_A, e_B және жалпы кілтті K_{AB} табу:

а) $p = 23$, $g = 5$, $d_A = 5$, $d_B = 7$;

б) $p = 19$, $g = 2$, $d_A = 5$, $d_B = 7$;

в) $p = 23$, $g = 7$, $d_A = 3$, $d_B = 4$;

г) $p = 17$, $g = 3$, $d_A = 10$, $x_B = 5$;

д) $p = 19$, $g = 10$, $d_A = 4$, $d_B = 8$.

27. *Шамир* криптографиялық жүйесі үшін берілген параметрлерімен p, e_A және d_A жетпейтін параметрлерді табу және M хабарын A дан B жіберу үрдісін сипаттау:

а) $p = 19$, $e_A = 5$, $e_B = 7$, $M = 4$;

б) $p = 23$, $e_A = 15$, $e_B = 7$, $M = 6$;

в) $p = 19$, $e_A = 11$, $e_B = 5$, $M = 10$;

г) $p = 23$, $e_A = 9$, $e_B = 3$, $M = 17$;

д) $p = 17$, $e_A = 3$, $e_B = 13$, $M = 9$.

28. *Эль-Гамаль* криптографиялық жүйесі үшін берілген параметрлерімен p, g, d_B және k жетпейтін параметрлерді табу және M хабарын B жіберу үрдісін сипаттау:

а) $p = 19$, $g = 2$, $d_B = 5$, $k = 7$, $M = 5$;

б) $p = 23$, $g = 5$, $d_B = 8$, $k = 10$, $M = 10$;

в) $p = 19$, $g = 2$, $d_B = 11$, $k = 4$, $M = 10$;

г) $p = 23$, $g = 7$, $d_B = 3$, $k = 15$, $M = 5$;

д) $p = 17$, $g = 3$, $d_B = 10$, $k = 5$, $M = 10$.

29. *RSA* жүйесінде берілген параметрлерімен p_A, q_A және e_A жетпейтін параметрлерді табу және M хабарын A жіберу үрдісін сипаттау:

а) $p_A = 5$, $q_A = 11$, $e_A = 3$, $M = 12$;

б) $p_A = 5$, $q_A = 13$, $e_A = 5$, $M = 20$;

в) $p_A = 7$, $q_A = 11$, $e_A = 7$, $M = 17$;

г) $p_A = 7$, $q_A = 13$, $e_A = 5$, $M = 30$;

д) $p_A = 3$, $q_A = 11$, $e_A = 3$, $M = 15$.

30. $n_A = 187$ және $e_A = 3$ параметрлері бар *RSA* жүйесінің A пайдаланушысына $C = 100$ шифрланған хабары жіберілді. Қасқой оны жолай ұстады. Берілген *RSA* жүйесін бұзу үрдісін түсіндіру және A пайдаланушысына жіберілген ашық мәтінді анықтау.

31. *Рабин* криптографиялық жүйесін қолданып, ашық кілтті генерациялап, ашық мәтінші шифрлау: “*криптография*”, $p_A = 53$, $q_A = 71$. Мәтінді сандық түрде жазған кезде әр әріпті орыс алфавитіндегі екісандық нөміріне (нөмірлеуді *00* бастаныз, бос орынды *33* санымен ауыстырыңыз) ауыстыру қажет және мәтінді төрт саннан блоктарға бөлу.

32. A пайдаланушысы ашық кілтті $n_A = 4189$ генерациялады және *Рабин* криптографиялық жүйесі көмегімен өз мекеніне шифрланған хабарды алды: $C = 4076\ 2375\ 1903$. Жабық кілттерді қалпына келтіру және шифрланған хабарды керішифрлау (керішифрланған хабардың цифрлық жазбасында әр екісандық ондық сан орыс алфавитының сәйкес әріпінің нөмірі болады, нөмірлеу *00* басталады және *31* бітеді).

33. *«Сәбі қадамы, дәу қадамы»* әдісін қолданып, келесі теңдіктерді шешу:

а) $2^x \bmod 29 = 21$; б) $3^x \bmod 31 = 25$; в) $2^x \bmod 37 = 12$;

г) $6^x \bmod 41 = 21$; д) $3^x \bmod 43 = 11$.

34. Дәрежені есептеу алгоритмін қолданып, келесі теңдіктерді шешу:

а) $2^x \bmod 53 = 24$; ….б) $2^x \bmod 59 = 13$; в) $2^x \bmod 61 = 45$;
г) $2^x \bmod 67 = 41$; ….д) $7^x \bmod 71 = 41$.

ЕСІМДІК СІЛТЕМЕ

ПӘНДІК СІЛТЕМЕ

Ә

Б

М

О

П

Р

С

Т

Ф

Ш

Э

ӘДЕБИЕТТЕР ТІЗІМІ

1. Аграновский А.В. Практическая криптография: алгоритмы и их программирование / А.В. Аграновский, Р.А. Хади. – М.: СОЛОН-Пресс, 2009. – 256 с.: ил.

2. Бабаш А.В. Криптография / А.В. Бабаш, Г.П. Шанкин; под редакцией В.П. Шерстюка, Э.А. Применко. – М.: СОЛОН-Пресс, 2007. – 512 с.: ил.

3. Блінцов В.С. Математичні основи криптології + CD : Навчальний посібник для студ. вищих навч. закл. / В.С. Блінцов, Ю.Л. Гальчевський. – Миколаїв: Національний ун-т кораблебудування ім. адмірала Макарова, 2006. – 232 с.: іл.

4. Безпека інформаційних систем і технологій: Навч. посібник / В. І. Єсін, О. О. Кузнецов, Л. С. Сорока. – Х. : ХНУ імені В. Н. Каразіна, 2013. – 632 с. : іл.

5. Богуш В.М. Криптографічні застосування елементарної теорії чисел : Навч. посібник / В.М. Богуш, В.А. Мухачов. – К.: Державний ун-т інформаційно-комунікаційних технологій, 2006. – 126 с.: іл.

6. Введение в криптографию / Н.П. Варновский, Ю.В. Нестеренко, Г.А. Кабатянский и др.; под ред. В.В. Ященко. – М.: МЦНМО-ЧеРо, 1998. – 272 с.: ил.

7. Горбенко І.Д. Прикладна криптологія. Теорія. Практика. Застосування : монографія / І.Д. Горбенко, Ю.І. Горбенко. – Харків: Видавництво "Форт", 2012. – 880 с.: іл.

8. Горбенко І.Д. Захист інформації в інформаційно-телекомунікаційних системах : Навч. посіб. для студ. Ч. 1. Криптографічний захист інформації / І. Д. Горбенко, Т. О. Гріненко. – Х. : Харк. нац. ун-т радіоелектрон., 2004. – 368 с.: іл.

9. Грайворонський М.В. Безпека інформаційно-комунікаційних систем : Підручник / М. В. Грайворонський, О. М. Новіков. – К. : Видавнича група BHV, 2009. – 608 с.: іл.

10. Задірака В.К. Комп'ютерна криптологія : Підручник / В.К. Задірака, О.С. Олексюк. – К.: Тернопільська академія народного господарства; НАН України; Інститут кібернетики ім. В.М. Глушкова, 2002. – 504 с.: іл.

11. Захист інформації в мережах передачі даних / Юдін О.К., Корченко О.Г., Конахович Г.Ф. – К.: Вид-во ТОВ «НВП» ІНТЕРСЕРВІС», 2009. – 716 с.: іл.

12. Защита информации в компьютерных системах и сетях / Ю.В. Романец, П.А. Тимофеев, В.Ф. Шаньгин; под ред. В.Ф. Шаньгина. – М.: Радио и связь, 2001. – 376 с.: ил.

13. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях / М.А. Иванов – М.: КУДИЦ-ОБРАЗ, 2001. – 363 с.: ил.

14. Коблиц Н. Курс теории чисел и криптографии / Н. Коблиц. – М.: Научное издательство ТВП, 2001. – 272 с.: ил.

15. Конеев И.Р. Информационная безопасность предприятия / И.Р. Конеев, А.В. Беляев. – СПб, БХВ-Петербург, 2003. – 752 с.: ил.

16. Корченко О.Г. Охорона конфіденційної інформації підприємства : Навч. посіб. / О. Г. Корченко, Ю. О. Дрейс. – Житомир: ЖВІ НАУ, 2011. – 172 с.: іл.

17. Коутинхо С. Введение в теорию чисел. Алгоритм RSA / С. Коутинхо. – М.: Постмаркет, 2001. – 328 с.: ил.

18. Мао В. Современная криптография: теория и практика / В. Мао; пер. с англ. – М.: Издательский дом “Вильямс”, 2005. – 768 с.: ил.

19. Математичні основи криптографії : навч. посібник / Г.В. Кузнєцов, В.В. Фомічов, С.О. Сушко, Л.Я. Фомічова. – Дніпропетровськ: Національний гірничий університет, 2004. – 391 с.: іл.

20. Математичні основи криптоаналізу : Навч. посіб. / С.О. Сушко, Г.В. Кузнецов, Л.Я. Фомичова, А.В. Корабльов. – Д. : Національний гірничий університет, 2010. – 465 с.: іл.

21. Математические и компьютерные основы криптологии : учебное пособие / Ю.С. Харин, В.И. Берник, Г.В. Матвеев, С.В. Агиевич. – Минск: Новое издание, 2003. – 382 с.: ил.

22. Математические основы криптологии : учебное пособие / Ю.С. Харин, В.И. Берник, Г.В. Матвеев. – Минск: БГУ, 1999. – 319 с.: ил.

23. Методи та алгоритми симетричної криптографії: навч. пос. / Кузнецов О.О., Євсеев С.П., Смірнов О.А., Мелешко Є.В., Король О.Г. – Кіровоград: Вид. КНТУ, 2012. – 316 с.: іл.

24. Молдовян Н.А. Криптография с открытым ключом / Н.А. Молдовян, А.А. Молдовян. – СПб.: БХВ-Петербург, 2005. – 288 с.: ил.

25. Мукачев В.А. Методы практической криптографии / В.А. Мукачев, А.А. Хорошко. – К.: ООО "Полиграф-Консалтинг", 2005. – 215 с.: ил.

26. Ожиганов А.А. Основы криптоанализа симметричных шифров : учебное пособие / Ожиганов А.А.– СПб: СПбГУ ИТМО, 2008. – 44 с.: ил.

27. Основы криптографии : учебное пособие / А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин. – М.: Гелиос АРВ, 2002. – 480 с.: ил.

28. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, В.Е. Серов. – М.: "Горячая линия-Телеком", 2001. – 154 с.: ил.

29. Панасенко С.П. Алгоритмы шифрования. Специальный справочник / С.П. Панасенко. – СПб: БХВ-Петербург, 2009. – 576 с.: ил.

30. Петров А.А. Компьютерная безопасность. Криптографические методы защиты / А.А. Петров. – М.: Издательство ДМК, 2000. – 448 с.: ил.

31. Поповский В.В. Защита информации в телекоммуникационных системах: учебник / В.В. Поповский, А.В. Персиков. – Харьков: ООО "Компания СМИТ", Т. 1. – 2006. – 238 с.: ил.

32. Поточные шифры / А.В. Асосков, М.А. Иванов, А.А. Мирский и др. — М.: КУДИЦ-ОБРАЗ, 2003. – 336 с.: ил.

33. Ростовцев А.Г. Теоретическая криптография / А.Г. Ростовцев, Е.Б. Маховенко. – СПб.: АНО НПО Профессионал, 2005. – 480 с.: ил.

34. Рябко Б.Я. Криптографические методы защиты информации: учебное пособие для вузов / Б.Я. Рябко, А.Н. Фионов. – М.: Горячая линия – Телеком, 2005. – 229 с.: ил.

35. Саломаа А. Криптография с открытым ключом : пер. с англ / А. Саломаа. – М.: Мир, 1996. – 304 с.: ил.

36. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования ГОСТ 28147-

89. Гос. Ком. СССР по стандартам. - М., 1989. ftp://ftp.wtc-ural.ru/pub/ru.crypt/ГОСТ28147

37. Смарт Н. Криптография : пер. с англ / Н. Смарт Н. – М.: Техносфера, 2005. – 528 с.: ил.

38. Стандарт криптографической защиты – AES. Конечные поля / А.С. Зензин, М.А. Иванов ; под ред. М.А. Иванова. – М.: КУДИЦ-ОБРАЗ, 2002. – 176 с.: ил.

39. Стеганографія: Навч. посіб. / О.О. Кузнецов, С.П. Євсеєв, О.Г. Король. – Х. : Вид. ХНЕУ, 2011. – 232 с.: ил.

40. Тилборг ван Х.К.А. Основы криптологии. Профессиональное руководство и интерактивный учебник : пер. с англ / Тилборг ван Х.К.А. – М.: Мир, 2006. – 471 с.: ил.

41. Фергюссон Н. Практическая криптография : пер. с англ. / Н. Фергюссон, Б. Шнайер. – М.: издательский дом “Вильямс”, 2005. – 424 с.: ил.

42. Фомичев В.М. Дискретная математика и криптология / Фомичев В.М. – М.: Диалог-МИФИ, 2003. – 400 с.: ил.

43. Фороузан Б.А. Криптография и безопасность сетей: Учебное пособие / Б.А. Фороузан; пер. с англ. Под ред. А.Н. Берлина. – М.: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2010. – 784 с.: ил.

44. Черемушкин А.В. Криптографические протоколы. Основные свойства и уязвимости : учеб. пособие для студ. учреждений высш. проф. образования / А.В. Черемушкин. – М.: Издательский центр "Академия", 2009. – 272 с.: ил.

45. Шеннон К. Теория связи в секретных системах // В кн. Работы по теории информации и кибернетики / К. Шеннон. – М.: ИЛ, 1963. – 830 с.: ил.

46. Щербаков А.Ю. Прикладная криптография. Использование и синтез криптографических интерфейсов / А.Ю. Щербаков, А.В. Домашев. – М.: Издательско-торговый дом “Русская редакция”, 2003. – 416 с.: ил.

47. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер Б. – М.: Триумф, 2002. – 797 с.: ил.

БАҚЫЛАУ ТАПСЫРМАЛАРЫНА ЖАУАПТАР

2 тарау

18. “НСПРГХГ”.
19. “растение”.
20. K = 7.
21. “ЖЪЩУЩЦХЪКЖ”.
22. “криптография”.
23. “ЗЧЦЖГАИБ”.
24. “космонавт”.
25. “РСЮФТЯГО”.
26. “осторожность”.
27. “ЭЬЙУЛДШЗЦЮЗФ”.
28. “вероятность_”.
29. “ФЦЧЫМЧСЪ”.
30. “зашифрование”.
31. “OLYMVQGAJDGAR”.
32. “WANT TO TRAVEL”.

3 тарау

10. а) *32* бит; б) *188* блок.
11. а) *24*; б) *5* бит.
12. а) *20 922 789 888 000*; б) *43* бит.
13. $(11011100)_2$.
14. $(11111010)_2$.
15. а) $(00000000)_2$; б) $(11111111)_2$; в) $(01001101)_2$; г) $(10110010)_2$.
16. $(00111)_2$.
17. $(011)_2$.
18. $(11110)_2$.
19. Түзу *P*-блогымен.
20. Кеңейту *P*-блогымен.
21. Қысу *P*-блогымен.
22. Түзу *P*-блогымен.

23.

Кіріс: сол жақ бит		Кіріс: оң жақ бит 0	1
	0	01	11
	1	00	10

24. а) *10*; б) *00*.

25. а) *110*; б) *011*.

4 тарау

10. а) *7*; *8*; *13*; *4*; *10*; *6*; *3*; *5*; *15*; *14*; б) *3*; *21*; *14*; *18*; *19*; *2*; *15*; *1*; *9*; *11*.

11. а) $k_3 = 0{,}1$; $k_4 = 0{,}2$; $k_5 = 0{,}3$; $k_6 = 0{,}8$; $k_7 = 0{,}4$; $k_8 = 0{,}9$; $k_9 = 0{,}9$; $k_{10} = 0{,}5$; $k_{11} = 0{,}4$; $k_{12} = 0{,}5$; б) $k_4 = 0{,}4$; $k_5 = 0{,}4$; $k_6 = 0{,}1$; $k_7 = 0{,}5$; $k_8 = 0{,}3$; $k_9 = 0{,}9$; $k_{10} = 0{,}8$; $k_{11} = 0{,}6$; $k_{12} = 0{,}5$; $k_{12} = 0{,}3$.

12. $a = 5$; $b = 3$; $c = 11$.

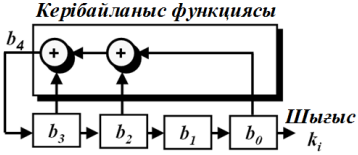
13. а) $x_{11} = 35$; б) $x_{11} = 679$.

14. а) $(110001011001)_2$; б) $(110111100011)_2$.

15. а) $x_1 = 81$; $x_2 = 6$; $x_3 = 36$; $x_4 = 422$; $x_5 = 225$; $x_6 = 370$; $x_7 = 119$; $x_8 = 177$; б) $x_1 = 81$; $x_2 = 144$; $x_3 = 59$; $x_4 = 629$; $x_5 = 639$; $x_6 = 485$; $x_7 = 648$; $x_8 = 660$.

16. 18, 4, 232, 61, 37, 201, 252, 166, 151, 162, 244, 66, 137, 130, 28, 248, 127, 77, 205, 226.

17. а)



б) *31* бит.

18. а) $x^4 + x^3 + x^2 + 1$; б) *15* біт.

19. $(1111000100010111000)_2$.

20. 5 разряд.

21. Сызықты жылжымалы регистрлердің екілік тізбегінің максималды периоды: $R_1 - 524287$ бит; $R_2 - 4194303$ бит; $R_3 - 8388607$ бит.

4.22. а) $f(x, y, z) = f(1, 0, 0) = 0$; б) $f(x, y, z) = f(0, 1, 1) = 1$; R_2 және R_3 сызықты жылжымалы регистрлері синхрондалады; в) $f(x, y, z) = f(0, 0, 0) = 0$; г) $f(x, y, z) = f(1, 1, 1) = 1$; R_1, R_2 және R_3 сызықты жылжымалы регистрлері синхрондалады.

5 тарау

11. $L_0||R_0 = cc1fc6e0f0aae8a5_{16}$.

12. $R_0^E = fa155575150b_{16}$.

13. $R_1^E = 8a2b57d029a6_{16}$.

14. $R_1^{\oplus} = cf430fca9568_{16}$.

15. $R_{12}^{\oplus} = 5dfd4bd5b555_{16}$.

16. а) 5_{16}; б) e_{16}; в) f_{16} және г) 0_{16}.

17. $R_{13}^S = 75140940eb51_{16}$.

18. $R_2^P = 1680e976_{16}$.

19. $R_3^L = 48a3638f_{16}$.

20. $L_5 = 48a3638f_{16}$ және $R_5 = 4178632b_{16}$.

21. $L_6 = 4178632b_{16}$ және $R_6 = 48a9b329_{16}$.

22. $C_0 = c3c033a_{16}$ және $D_0 = 33f0cfa_{16}$.

23. а) $C_2 = 0f00ceb_{16}$ және $D_2 = cfc33e8_{16}$; б) $C_3 = 3c033ac_{16}$ және $D_3 = 3f0cfa3_{16}$; в) $C_7 = 033ac3c_{16}$ және $D_7 = 0cfa33f_{16}$; г) $C_{12} = 7587806_{16}$ және $D_{12} = f467e19_{16}$.

24. $k_2 = 4568581abcce_{16}$.

25. а) $k_4 = da2d032b6ee3_{16}$; б) $k_9 = 84bb4473dccc_{16}$; в) $k_{13} = 99c31397c91f_{16}$; г) $k_{16} = 181c5d75c66d_{16}$.

26. $f07704d0741eb2c2_{16}$.

6 тарау

14. Бірінші әсер керішифрланған деректердің блогының бірыңғай биты бұзылады. Екінші әсер керішифрланған деректердің келесі *n/l* блогы бұзылады.

15. Керішифрланған деректердің бірінші блогында шамамен жарты биттер бұзылады.

16. Керішифрланған деректердің бірінші блогында шамамен жарты биттер бұзылады; екіншіде – бір (*5*) бит.

17. Ашық деректердегі қате барлық келесі шифрланған деректерге әсер етеді, бірақ керішифрлау барысында өзінен өзі жойылады.

18. Бірінші әсер керішифрланған деректердің бірінші блогының екінші битының бұзылуы. Екінші әсер керішифрланған деректердің екіншіден тоғызыншыға дейін блоктарының бұзылуы.

19. Керішифрланған деректердің бірінші блогының екінші битының бұзылуы болады. Блоктардың келесі тізбектері дұрыс керішифрланады.

20. Керішифрланған деректердің тек екінші блогының екінші битының бұзылуы болады.

7 тарау

11. $N_1 - 0cbd4791_{16}$, $N_2 - 191a2ab8_{16}$.
12. $N_1 - cc29563b_{16}$, $N_2 - 434665b2_{16}$.
13. $N_1 - 10d3b61a_{16}$, $N_2 - 02af83f6_{16}$.
14. $N_1 - 0d3b61a5_{16}$, $N_2 - 3153698e_{16}$.
15. Имитоендірменің ұзындығы *32* биттен кем болмау керек.
16. Жалған кедергілерді тықпалау ықтималдығы $P_{ЛП} \approx 5{,}96 \cdot 10^{-10}$.

8 тарау

11. $x^7 + x^6 + x^2 + x$.
12. Бөлуден бөлшек: $x^7 + x^5 + x^4 + x^2 + 1$. Бөлуден қалдық: $x^5 + + x^3 + x$.
13. $\{ f2 \} \cdot x^7 + \{ 1b \} \cdot x^5 + \{ 03 \} \cdot x^3 + \{ 17 \} \cdot x + \{ 1d \}$.
14. $\{ 06 \} \cdot x^6 + \{ 05 \} \cdot x^5 + \{ 19 \} \cdot x^4 + \{ 15 \} \cdot x^3 + \{ 1d \} \cdot x^2 + \{ 13 \} \cdot x + \{ 04 \}$.
15. $\{ 19 \} \cdot x^3 + \{ 11 \} \cdot x^2 + \{ 1c \} \cdot x + \{ 0a \}$.
16. $x^7 + x^6 + x^3 + x^2 + x$.
17. $x^5 + x^2 + 1$.
18. $x^7 + x^6 + x^5 + x + 1$.
19. $49ded28945db96f17f39871a7702533b_{16}$.
20. $2b359f6849506a2f27f9ca443ea5b6b_{16}$.
21. $2b359f6849506a2f27f9ca443ea5b6b_{16}$.
22. $f187de773b9639498953db7f1ad2245_{16}$.
23. $584dcaf11b4b5aacdbe7caa81b6bb0e5_{16}$.
24. $72dbe546102a2bf01ad3ef5836ab483d_{16}$.
25. $aa8f5f0361dde3ef82d24ad26832469a_{16}$.
26. $d014f9a8c9ee2589e13f0cc8b6630ca6_{16}$.
27. $w[6] = 5846f2f9_{16}$; $w[7] = 5c43f4fe_{16}$; $w[8] = 544afef5_{16}$; $w[9] = 5847f0fa_{16}$; $w[10] = 4856e2e9_{16}$; $w[11] = 5c43f4fe_{16}$.
28. $w[8] = a573c29f_{16}$; $w[9] = a176c498_{16}$; $w[10] = a97fce93_{16}$; $w[11] = a572c09c_{16}$; $w[12] = 1651a8cd_{16}$; $w[13] = 0244beda_{16}$; $w[14] = 1a5da4c1_{16}$; $w[15] = 0640bade_{16}$.

9 тарау

8. $k_2^{(1)} = 96a2_{16}$, $k_2^{(2)} = 978e_{16}$, $k_2^{(3)} = b820_{16}$, $k_2^{(4)} = b940_{16}$, $k_2^{(5)} = 1af6_{16}$, $k_2^{(6)} = 7b7d_{16}$.

9. $k_2^{(1)} = e8d8_{16}$, $k_2^{(2)} = 5099_{16}$, $k_2^{(3)} = 4c25_{16}$, $k_2^{(4)} = 95d7_{16}$, $k_2^{(5)} = eb43_{16}$, $k_2^{(6)} = a108_{16}$, $k_2^{(7)} = 8382_{16}$, $k_2^{(8)} = 47e0_{16}$, $k_2^{(9)} = b324_{16}$.

10. $k_3^{(1)} = fca2_{16}$, $k_3^{(2)} = 6bff_{16}$, $k_3^{(3)} = ff29_{16}$, $k_3^{(4)} = 9427_{16}$, $k_3^{(5)} = 9b4b_{16}$, $k_3^{(6)} = a576_{16}$, $k_3^{(7)} = bad1_{16}$, $k_3^{(8)} = 6842_{16}$, $k_3^{(9)} = efa4_{16}$.

11. $k_5^{(1)} = 92d4_{16}$, $k_5^{(2)} = c7e8_{16}$, $k_5^{(3)} = f424_{16}$, $k_5^{(4)} = 1266_{16}$, $k_5^{(5)} = 3370_{16}$, $k_5^{(6)} = 8035_{16}$, $k_5^{(7)} = 1af6_{16}$, $k_5^{(8)} = 7b3d_{16}$.

12. $k_1^{(1)} = 5773_{16}$, $k_1^{(2)} = 7f25_{16}$, $k_1^{(3)} = ab30_{16}$, $k_1^{(4)} = e2ef_{16}$, $k_1^{(5)} = 529a_{16}$, $k_1^{(6)} = 20e2_{16}$, $k_1^{(7)} = c77b_{16}$, $k_1^{(8)} = 9a70_{16}$, $k_1^{(9)} = 7bcc_{16}$.

13. $X_1^{(1)} = abc3_{16}$, $X_2^{(1)} = 5c2d_{16}$, $X_3^{(1)} = e77a_{16}$, $X_4^{(1)} = 1c2d_{16}$.

14. $y_1^{(1)} = 5c92_{16}$, $y_2^{(1)} = 62b8_{16}$.

15. $C = 4825e238d0fb9c46_{16}$.

16. $f_1^{(1)} = 4cb9_{16}$, $f_2^{(1)} = 4000_{16}$.

10 тарау

13. а) $5 \bmod 10 = 5$, $16 \bmod 10 = 6$, $27 \bmod 10 = 7$, $-4 \bmod 10 = 6$, $-13 \bmod 10 = -3 \bmod 10 = 7$, $(3 + 8) \bmod 10 = 1$, $(3 - 8) \bmod 10 = 5$, $(3 \cdot 8) \bmod 10 = 4$, $(3 \cdot 8 \cdot 5) \bmod 10 = (4 \cdot 5) \bmod 10 = 0$. *б)* $5 \bmod 11 = 5$, $16 \bmod 11 = 5$, $27 \bmod 11 = 5$, $-4 \bmod 11 = 7$, $-13 \bmod 11 = -2 \bmod 11 = 9$, $(3 + 8) \bmod 11 = 0$, $(3 - 8) \bmod 11 = 6$, $(3 \cdot 8) \bmod 11 = 2$, $(3 \cdot 8 \cdot 5) \bmod 11 = (2 \cdot 5) \bmod 11 = 10$.

14. $2^8 \bmod 10 = 6$, $2^7 \bmod 10 = 7$, $7^{19} \bmod 100 = 43$, $7^{57} \bmod 100 = 7$.

15. $108 = 2 \cdot 2 \cdot 3 \cdot 3 \cdot 3$, $77 = 7 \cdot 11$, $65 = 5 \cdot 13$, $30 = 3 \cdot 3 \cdot 5$, $159 = 3 \cdot 53$.

16. (*25,12*) және (40,27) жұптары өзаражай сандар, қалғандары – жоқ ((*25,15*) сандары 5 бөлінеді, (*13,39*) сандары *13* бөлінеді).

17. $\varphi(14) = 6$, $\varphi(15) = 8$.

18. $\varphi(53) = 52$, $\varphi(21) = \varphi(7) \cdot \varphi(3) = 6 \cdot 2 = 12$, $\varphi(159) = \varphi(3) \cdot \varphi(53) = 2 \cdot 52 = 104$.

19. $3^{13} \bmod 13 = 3 \cdot 3^{12} \bmod 13 = 3$, $5^{22} \bmod 11 = 5^2 \cdot 5^{10} \cdot 5^{10} \bmod 11 = 25 \bmod 11 = 3$, $3^{17} \bmod 5 = 3$.

20. $3^9 \bmod 20 = 3 \cdot 3^8 \bmod 20 = 3$, $2^{14} \bmod 21 = 2^2 \cdot 2^{12} \bmod 21 = 4$, $2^{107} \bmod 159 = 2^3 \cdot 2^{104} \bmod 159 = 8$.

21. $gcd(21,12) = 3$, $gcd(30,12) = 6$, $gcd(24,40) = gcd(40,24) = 8$, $gcd(33,16) = 1$.

22. а) $x = -1$, $y = 2$. б) $x = 1$, $y = -2$. в) $x = 2$, $y = -1$. г) $x = 1$, $y = -2$.

23. $3^{-1} \bmod 7 = 5$, $5^{-1} \bmod 8 = 5$, $3^{-1} \bmod 53 = 18$, $10^{-1} \bmod 53 = 16$.

24. 100 кіші жай сандар: *2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 73, 79, 83, 89, 97*. Олардың ішінен *5, 7, 11, 23, 47, 59* және *83* сандары $p = 2 \cdot q + 1$ түріне сәйкес келеді.

25. $p = 11$ кезде g параметрі ретінде *2, 6, 7* және *8* сандары таңдалу мүмкін.

26. а) $e_A = 20, e_B = 17, K_{AB} = 21$; б) $e_A = 13, e_B = 14, K_{AB} = 10$; в) $e_A = 21, e_B = 9, K_{AB} = 16$; г) $e_A = 8, e_B = 5, K_{AB} = 9$; д) $e_A = 6, e_B = 17, K_{AB} = 16$.

27. а) $d_A = 11, d_B = 13, x_1 = 17, x_2 = 5, x_3 = 6, x_4 = 4$; б) $d_A = 3, d_B = 19, x_1 = 8, x_2 = 12, x_3 = 3, x_4 = 6$; в) $d_A = 5, d_B = 11, x_1 = 14, x_2 = 10, x_3 = 3, x_4 = 10$; г) $d_A = 5, d_B = 15, x_1 = 7, x_2 = 21, x_3 = 14, x_4 = 17$; д) $d_A = 11, d_B = 5, x_1 = 15, x_2 = 2, x_3 = 8, x_4 = 9$.

28. а) $e_B = 13, r = 14, s = 12, M' = 5$; б) $e_B = 16, r = 9, s = 15, M' = 10$; в) $e_B = 15, r = 16, s = 14, M' = 10$; г) $e_B = 21, r = 14, s = 12, M' = 5$; д) $e_B = 8, r = 5, s = 12, M' = 10$.

29. а) $n_A = 55, \varphi(n_A) = 40, d_A = 27, C = 23, M' = 12$; б) $n_A = 65, \varphi(n_A) = 48, d_A = 29, C = 50, M' = 20$; в) $n_A = 77, \varphi(n_A) = 60, d_A = 43, C = 52, M' = 17$; г) $n_A = 91, \varphi(n_A) = 72, d_A = 29, C = 88, M' = 30$; д) $n_A = 33, \varphi(n_A) = 20, d_A = 7, C = 9, M' = 18$.

30. $M = 111$.

31. Ашық кілт $n_A = 3763$, шифрланған хабар $C = 1194\ 1937\ 1734\ 2018\ 0400\ 1932\ 0831$.

32. $p_A = 59, q_A = 71$; “*защита*”.

33. а) $x = 17$; б) $x = 10$; в) $x = 28$; г) $x = 14$; д) $x = 30$.

34. а) $x = 10000$; б) $x = 20000$; в) $x = 1000$; г) $x = 12345$; д) $x = 25000$.

Оқулық басылым

Бахытжан Сражатдинович Ахметов
Александр Григорьевич Корченко
Владимир Павлович Сиденко
Юрий Александрович Дрейс
Жулдыз Кенесхановна Алимсеитова

ҚОЛДАНБАЛЫ КРИПТОЛОГИЯ: ШИФРЛАУ ӘДІСТЕРІ

Оқулық

ОБО РБ бастығы *З. А. Ғұбайдулина*
Редакторы *Т. С. Жақсыбаева*
Компьютерде беттеген *Л. Өмірбекова*

Басуға қол қойылды __.__.2016 ж.
Таралымы 100 дана. Пішімі 60х84х 1/16. № 1 баспаханалық қағаз.
Е.-б.т. 12,1. Ш.б.т. 12,2. Тапсырыс № 4. Бағасы келісімді.

Қ. И. Сәтбаев атындағы Қазақ ұлттық зерттеу техникалық университеті басылымы

Қ. И. Сәтбаев атындағы ҚазҰЗТУ Оқу-баспа орталығы,
Алматы қ., Сәтбаев көшесі, 22

ISBN 978-601-228-331-0