МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих
систем захисту інформації
_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.5:510.22(043.3)

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Метод захисту інформації в кіберпросторі від кібервпливу

Виконавець: Іван ПОЛЯК
Керівник: д.т.н., професор Сергій ТОЛЮПА
Консультант розділу «Охорона навколишнього середовища»: Тетяна ДМИТРУХА
Нормоконтролер: д.т.н., професор Сергій ТОЛЮПА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«___» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
здобувача вищої освіти Поляка Івана Івановича

1. Тема: *Метод захисту інформації в кіберпросторі від кібервпливу*
затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: проаналізувати нормативно-правову базу щодо забезпечення національної безпеки України в інформаційній сфері; проаналізувати та порівняти існуючі системи шифрування, знайти їх переваги та недоліки; описати модифікований метод асиметричного шифрування та проаналізувати його особливості із використанням складеного модуля та способу розшифрування з використанням наслідку з Китайської теореми про залишки.
4. Зміст пояснювальної записки: проаналізовано існуючі методи шифрування, виявлено основні переваги та недоліки кожного з методів, описано запропонований метод асиметричного шифрування та проаналізовано його

особливості, розробка програмного забезпечення запропонованого методу, проведено експериментальне дослідження ефективності модифікованого алгоритмічно-програмного методу асиметричного шифрування.

5. КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	20.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	30.10.2023	*Виконано*
4.	Збір інформації	01.11.2023	*Виконано*
5.	Дослідження теоретичних аспектів кібернетичних загроз	07.11.2023	*Виконано*
6.	Модифікація алгоритмічно-програмного методу асиметричного шифрування	14.11.2023	*Виконано*
7.	Експериментальне дослідження розробленого алгоритмічно-програмного методу асиметричного шифрування	21.11.2023	*Виконано*
8.	Апробація роботи на міжнародній науково-практичній конференції «ЖИВУЧІСТЬ ТА РЕЗИЛЬЄНТНІСТЬ – 2023»	19.10.2023	*Виконано*
9.	Перевірка на антиплагіат	11.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	16.12.2023	*Виконано*
11.	Оформлення презентації	16.12.2023	*Виконано*
12.	Отримання рецензій	22.12.2023	*Виконано*

6. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Іван ПОЛЯК

(підпис, дата)

Керівник кваліфікаційної роботи Сергій ТОЛЮПА

(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота на тему: «Метод захисту інформації в кіберпросторі від кібервпливу» складається зі вступу, п’ятьох розділів, загальних висновків, списку використаних джерел, додатків. Загальний обсяг роботи - 100 сторінок. Робота місить 21 рисунок. Список використаних джерел містить 34 джерел.

Метою кваліфікаційної роботи є вдосконалення процесу асиметричного шифрування для роботи з велики ключами шифрування.

В роботі вирішено задачу розробки програмного забезпечення, яке реалізує запропонований модифікований метод шифрування.

В роботі модифіковано метод асиметричного шифрування, який відрізняється від існуючих тим, що використовує складений модуль, що формується з простих чисел. Це поєднано із використанням наслідку з Китайської теореми про залишки. Такий підхід дозволяє ефективно знижувати обчислювальну складність, що зростає експоненціально пропорційно збільшенню довжини шифрувального модуля.

Ключові слова: ІНФОРМАЦІЯ, ЗАХИСТ, МЕТОД, RSA, МЕТОДИ ШИФРУВАННЯ, ОБЧИСЛЮВАЛЬНА СКЛАДНІСТЬ.

ЗМІСТ

ВСТУП

Актуальність. Розвиток інформаційного суспільства та зростання ролі інформаційних технологій в усіх сферах життя створюють як можливості для розвитку, так і нові ризики. З однієї сторони, інформаційні системи стають все більш важливими для функціонування економіки, держави та суспільства. Вони використовуються для зберігання, обробки та передачі інформації, яка є критичною для життєдіяльності людей і організацій. З іншої - це також робить ці системи більш уразливими для кіберзагроз. Кібератаки можуть призвести до порушення доступу до інформації, її втрати або спотворення, а також до витоку конфіденційної інформації.

Одна з основних проблем у кіберпросторі є захист інформації, що обробляється в інформаційно-телекомунікаційних системах. Кібернетичний простір, який постійно розвивається та впроваджує нові технології, створює безпрецедентні умови для накопичення та використання інформації. Це також призводить до фундаментальної залежності від нормального функціонування інформаційних систем у всіх сферах життєдіяльності суспільства.

Така залежність робить інформаційні системи вразливими до кібератак, які можуть призвести до порушення цілісності, доступності та конфіденційності інформації, а також нанесення шкоди інформаційним ресурсам і телекомунікаційним системам. Особливу занепокоєність викликає можливість застосування інформаційних технологій та кіберпростору для військово-політичних цілей, тероризму та хакерських атак.

У таких умовах одним із головних завдань держави є прийняття заходів, які спрямовані на протидію незаконним діям у кіберпросторі, запобігання або зменшення негативних наслідків від реалізації кіберзагроз. Тому з розвитком глобального інформаційного суспільства кібербезпека стає обов’язковою складовою національної безпеки будь-якої держави.

Метою кваліфікаційної роботи є вдосконалення процесу асиметричного шифрування для ефективної роботи із великими ключами шифрування, щоб гарантувати криптографічну стійкість за рахунок зменшення обчислювальної складності та підвищення стійкості до криптографічних атак.

Для досягнення поставленої мети вирішуються такі задачі:

– проаналізувати нормативно-правову базу щодо забезпечення національної безпеки України в інформаційній сфері; проаналізувати та порівняти існуючі системи шифрування, знайти їх переваги та недоліки; описати модифікований метод асиметричного шифрування та проаналізувати його особливості із використанням складеного модуля та способу розшифрування з використанням наслідку з Китайської теореми про залишки;

– здійснити аналіз мови програмування та середовища розробки, що необхідні для ефективної розробки програмного продукту;

– на основі проведених досліджень розробленого алгоритмічно-програмного методу асиметричного шифрування, зіставити ефективність за часом роботи з тим, що демонструє класичний метод асиметричного шифрування RSA.

Галузь застосування. Розроблений програмний продукт щодо вдосконалення методу асиметричного шифрування RSA є важливим в сучасному інформаційному середовищі. Він може забезпечити криптостійкість асиметричного шифрування на найближчі 20 років шляхом зменшення його обчислювальної складності та збільшення резистентності до криптографічних атак.

Об'єктом дослідження є процес асиметричного шифрування даних.

Предметом дослідження є методи асиметричного шифрування даних.

Методи дослідження. В роботі використовуються методи абстрактної алгебри, методи математичного програмування, метод інтерполяції.

Новизна одержаних результатів полягає в наступному:

- розроблено метод асиметричного шифрування, що відрізняється від існуючих застосуванням складеного модуля, що складається з простих чисел,

кількість яких наближено дорівнює $\frac{1}{25}$ від довжини модуля, у поєднанні із застосуванням наслідку з Китайської теореми про залишки, що дозволяє зменшити обчислювальну складність експоненціально пропорційно до збільшення довжини модуля шифрування.

Практичне цінність отриманих результатів:

- запропонований метод асиметричного шифрування має зменшену обчислювальну складність порівняно з існуючими методами для довжини модуля понад 10 тисяч біт.

Апробація. Основні положення роботи доповідалися та обговорювалися на Міжнародній науково-практичній конференції «ЖИВУЧІСТЬ ТА РЕЗИЛЬЄНТНІСТЬ – 2023», м. Київ, 19 жовтня 2023 р., ІПМЕ ім. Г.Є. Пухова НАН України.

РОЗДІЛ 1. ЗАХИСТ ІНФОРМАЦІЙНИХ РЕСУРСІВ ЧЕРЕЗ КІБЕРНЕТИЧНУ БЕЗПЕКУ

Кібернетичні загрози можуть бути різноманітними, включаючи атаки хакерських груп, іноземні державні структури, терористичні та екстремістські організації, транснаціональні корпорації. Загрози можуть виникати як всередині країни, так і з-за її кордонів. Також можливе використання української інформаційної інфраструктури як "транзитного майданчику" для атак на треті сторони.

Кіберпростір має бути відкритим для інновацій, вільного обміну інформацією та ідеями. Заходи з забезпечення кібербезпеки не повинні суперечити правам і свободам громадян, включаючи право на приватне життя та свободу спілкування.

Забезпечення кібербезпеки вимагає впровадження комплексних заходів та співпрацю з приватним сектором і громадським суспільством в партнерстві з державними органами. Без такої співпраці вирішення цього завдання неможливе.

Забезпечення кібербезпеки України здійснюється в рамках відповідних правових норм, зокрема Конституції, Закону України "Про основні засади внутрішньої та зовнішньої політики", Доктрини інформаційної безпеки України, Стратегії національної безпеки України, Закону України "Про основи національної безпеки", а також міжнародних документів, включаючи Конвенцію Ради Європи про кіберзлочинність.

Метою цієї Стратегії є:

- визначення основних принципів формування державної політики у сфері кібернетичної безпеки України, включаючи створення нормативно-правової бази, яка відповідає міжнародним стандартам у цій галузі.

- розробка сучасної національної системи кібербезпеки, яка забезпечить ефективну співпрацю між уповноваженими державними органами під час впровадження заходів з підвищення кібернетичної безпеки України.
- створення умов для активної співпраці між державним та приватними секторами, громадянами і громадським суспільством з метою протидії кіберзагрозам і розвитку міжнародного співробітництва в галузі кібернетичної безпеки.
- забезпечення кібернетичного захисту інформаційної інфраструктури України, зокрема об'єктів критичної інформаційної інфраструктури країни.
- розвиток системи підготовки фахівців у галузі кібернетичної безпеки.

1.1. Терміни і визначення.

Кібератака - це злочинна діяльність, спрямована на комп'ютерні системи, мережі, програми або дані з метою завдання шкоди, втрат або незаконного доступу. Це може включати в себе різноманітні види атак, такі як віруси, черви, троянські коні, фішинг, деніал-сервіс атаки (DDoS) та багато інших.

Кібербезпека – це стан захищеності інформаційних систем, даних і мереж від несанкціонованого доступу, використання, розголошення, модифікації або знищення.

Кібервійна - це форма війни або конфлікту, в якій кібератаки використовуються як засіб для завдання шкоди комп'ютерним системам, мережам, програмам або інформації ворога.

Кіберзагроза - це потенційна або активна небезпека, яка виходить від кібернетичних атак або інших зловживань в інтернеті та комп'ютерних системах.

Кіберзахист - це сукупність заходів, технологій, процедур і стратегій, спрямованих на захист комп'ютерних систем, мереж, даних та інформації від кіберзагроз і кібератак.

Кіберзлочинність - це злочинна діяльність, яка виконується в цифровому або кібернетичному середовищі, зазвичай за допомогою комп'ютерів, мереж і інтернету.

Кіберпростір - це віртуальне середовище, створене завдяки інформаційним технологіям і інтернету, в якому існують комп'ютери, мережі, програми, дані та користувачі.

Кібертероризм - це використання кібернетичних атак або кібератак з метою завдання шкоди або тисків на політичні, соціальні або економічні цілі з метою досягнення політичної або ідеологічної мети цільової групи або організації, включаючи терористичні організації.

Критична інформаційна інфраструктура держави - це сукупність об'єктів, систем та мереж, які мають важливе значення для функціонування країни та її економіки.

Об'єкт критичної інформаційної інфраструктури - це об'єкт, який забезпечує функціонування життєво важливих систем і послуг для суспільства та держави.

Суб'єкти забезпечення кібербезпеки - це організації, установи або індивіди, які займаються впровадженням заходів та забезпеченням безпеки в цифровому середовищі для захисту інформації, комп'ютерних систем, мереж і інших цифрових ресурсів. Ці суб'єкти відіграють важливу роль у забезпеченні цифрової безпеки на різних рівнях, включаючи індивідуальний, корпоративний, урядовий та міжнародний.

1.2. Загрози в сфері кібербезпеки

Кіберзлочинність. Кіберзлочини стають все більш поширеними в сучасному світі. Нові технології використовуються не лише для скоєння традиційних злочинів, а й для нових, характерних для інформаційного суспільства. Злочинці часто намагаються порушити роботу або несанкціоновано

використовувати інформаційні системи державних, фінансових, комерційних, військових та промислових підприємств. Інформація з обмеженим доступом, яка циркулює в цих системах, є привабливою для інших держав, організацій та окремих осіб. Кіберзлочини можуть мати серйозні наслідки, такі як фінансові втрати, порушення роботи критичної інфраструктури або навіть шкоду національній безпеці. Для боротьби з кіберзлочинністю необхідні комплексні заходи, які включають в себе розробку нормативно-правової бази, впровадження технічних засобів захисту та підвищення обізнаності про кіберзагрози.

Кібертероризм. Деякі підприємства, установи та організації, порушення роботи яких може призвести до шкоди життю та здоров'ю людей, є потенційними цілями для терористичних атак, зокрема із застосуванням сучасних інформаційних технологій. Терористичні атаки на інформаційні системи можуть бути використані для поширення дезінформації та пропаганди, що може призвести до дестабілізації суспільства. Ці атаки можуть також бути використані для кібершпіонажу, що може завдати шкоди національній безпеці і можуть призвести до серйозних економічних втрат, а також до порушення громадського порядку.

Кібервійна. Військова галузь переживає найбільш значущі зміни завдяки розвитку кіберпростору. Велика кількість країн усерйоз розвиває свої оборонні можливості в напрямку посилення здатностей для проведення військових операцій у кіберпросторі та захисту від аналогічних атак від супротивників, оскільки нові види загроз стають надзвичайно актуальними. Кіберпростір стає все більш важливим елементом військової стратегії. Країни, які не вміють ефективно використовувати кіберпростір, можуть стати вразливими перед кібератаками. Кібервійна може бути частиною традиційної війни або ж використовуватися як самостійний інструмент. Захист від кіберзагроз вимагає співпраці не лише міжнародних оборонних структур, але й активної участі громадянського суспільства та приватного сектору.

1.3. Вразливість інформаційної інфраструктури

У останній період частота кібератак та кіберзлочинів значно зросла. Вони спрямовані на інформаційні ресурси фінансових установ, транспортних підприємств, сектору енергетики, інституцій державного рівня, що відповідають за безпеку, оборону, і реагування на надзвичайні ситуації. Також стали об'єктом атаки сервери офіційних веб-сайтів та сервери цих установ. Потужний зріст кібератак на інформаційні ресурси держави свідчить про посилення активності хакерських груп з метою паралізації інформаційно-телекомунікаційних систем урядових органів.

Набирає обертів також політично мотивована діяльність у кіберпросторі, яка здійснюється групами активістів, які нападають на урядові та приватні веб-ресурси. Це призводить до відмов у роботі інформаційних ресурсів, а також завдає шкоди репутації та призводить до фінансових втрат.

Слабкий рівень захисту інформації, виявлений під час державних аудитів, може призвести до паралізування роботи критичних інформаційних об'єктів та, наслідок, зниження обороноздатності країни, економічної та фінансової нестабільності, а також погіршення іміджу та зниження привабливості для інвестицій і багато іншого.

1.4. Основні засади забезпечення кібернетичної безпеки

Діяльність із забезпечення кібербезпеки базується на таких засадах:

- Діяльність із забезпечення кібербезпеки повинна здійснюватися в рамках верховенства права та з дотриманням прав і свобод людини.
- Діяльність із забезпечення кібербезпеки повинна бути комплексною і включати правові, організаційні, технічні та інформаційні заходи.
- Запобіжні заходи повинні мати пріоритет перед заходами реагування на кібератаки.

- Кіберзлочини повинні бути суворо покарані, а постраждалі повинні бути відшкодовані.
- Держава та приватний сектор повинні співпрацювати у сфері кібербезпеки.
- Суб'єкти забезпечення кібербезпеки несуть відповідальність за захист критичної інфраструктури.
- Заходи із забезпечення кібербезпеки повинні бути дієвими, комплексними та постійними.

1.5. Основні шляхи протистояння загрозам у галузі кібербезпеки.

Для протидії реальним небезпекам, запобігання реалізації потенційних загроз у кіберпросторі і зменшення їх впливу на основні аспекти життя особи, суспільства і держави, Україна вживає широкий спектр заходів з метою забезпечення кібернетичної безпеки.

Розвиток кібербезпеки та прийняття нових законів і положень у цій галузі є невід'ємною частиною сучасного урядового управління і забезпечення стабільності та безпеки в цифровому світі. Застосування цих заходів допомагає державам виявляти, запобігати та ефективно реагувати на кіберзагрози та атаки.

Закон України "Про основні засади забезпечення кібербезпеки України" визначає правові та організаційні основи забезпечення захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.

Закон визначає основні цілі, напрями та принципи державної політики у сфері кібербезпеки. До основних цілей державної політики у сфері кібербезпеки належать:

- захист життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі від кіберзагроз;

- створення умов для розвитку інформаційного суспільства;
- забезпечення інформаційної безпеки України.

Основними напрямами державної політики у сфері кібербезпеки є:

- розроблення та реалізація нормативно-правової бази у сфері кібербезпеки;
- формування та розвиток системи кібербезпеки;
- підготовка та підвищення кваліфікації кадрів у сфері кібербезпеки;
- розвиток міжнародного співробітництва у сфері кібербезпеки.

Закон визначає повноваження державних органів, підприємств, установ, організацій, осіб та громадян у сфері кібербезпеки.

Державні органи у сфері кібербезпеки:

- розробляють та реалізують державну політику у сфері кібербезпеки;
- забезпечують формування та розвиток системи кібербезпеки;
- координують діяльність суб'єктів забезпечення кібербезпеки;
- беруть участь у міжнародному співробітництві у сфері кібербезпеки.

Підприємства, установи, організації у сфері кібербезпеки:

- забезпечують захист своїх інформаційних систем та мереж від кіберзагроз;
- впроваджують заходи з підвищення рівня кібербезпеки;
- беруть участь у формуванні та розвитку системи кібербезпеки.

Особи та громадяни у сфері кібербезпеки:

- вживають заходів щодо захисту своїх персональних даних та інших конфіденційних даних від кіберзагроз;
- не поширюють інформацію, яка може призвести до порушення кібербезпеки;
- беруть участь у формуванні та розвитку культури кібербезпеки.

Закон визначає основні засади координації діяльності суб'єктів забезпечення кібербезпеки. Координація діяльності суб'єктів забезпечення

кібербезпеки здійснюється з метою забезпечення єдиного підходу до вирішення завдань у сфері кібербезпеки.

Законом також передбачено відповідальність за порушення законодавства у сфері кібербезпеки.

Закон є важливим кроком на шляху до забезпечення кібербезпеки України. Визначає правові та організаційні основи забезпечення захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі. Закон є основою для подальшого розвитку системи кібербезпеки України.

Зазначений закон був прийнятий в контексті зростаючих загроз у кіберпросторі та розширення використання інформаційних технологій в Україні. Він має на меті зміцнити кібербезпеку країни та створити систему заходів для захисту інформації, критично важливих об'єктів та інфраструктури.

1.6. Система гарантування безпеки кіберпростору

Система забезпечення кібербезпеки України складається з державних органів, підприємств, установ, організацій, осіб та громадян, які здійснюють заходи щодо захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.

Державні органи у сфері кібербезпеки:

- Державна служба спеціального зв'язку та захисту інформації України - є центральним органом виконавчої влади, який забезпечує реалізацію державної політики у сфері кібербезпеки, захист державних інформаційних ресурсів та об'єктів критичної інфраструктури від кіберзагроз.
- Міністерство цифрової трансформації України - забезпечує реалізацію державної політики у сфері цифрового розвитку, цифрових трансформацій і цифровізації.

- Міністерство внутрішніх справ України - забезпечує безпеку громадян, суспільства та держави від кіберзагроз.
- Служба безпеки України - здійснює контррозвідувальну діяльність у сфері кібербезпеки.
- Державна прикордонна служба України - забезпечує захист державного кордону України від кіберзагроз.
- Національний банк України - забезпечує захист фінансової системи України від кіберзагроз.
- Державна аудиторська служба України - здійснює аудит у сфері кібербезпеки.

Підприємства, установи, організації у сфері кібербезпеки:

- Забезпечують захист своїх інформаційних систем та мереж від кіберзагроз.
- Впроваджують заходи з підвищення рівня кібербезпеки.
- Беруть участь у формуванні та розвитку системи кібербезпеки.

Особи та громадяни у сфері кібербезпеки:

- Вживають заходів щодо захисту своїх персональних даних та інших конфіденційних даних від кіберзагроз.
- Не поширюють інформацію, яка може призвести до порушення кібербезпеки.
- Беруть участь у формуванні та розвитку культури кібербезпеки.

Держава робить і вже зробила ряд заходів щодо забезпечення кібербезпеки України. Серед них:

- Прийняття Закону України "Про основні засади забезпечення кібербезпеки України". Закон визначає правові та організаційні основи забезпечення захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.
- Формування та розвиток системи кібербезпеки України. Система кібербезпеки України складається з державних органів, підприємств,

установ, організацій, осіб та громадян, які здійснюють заходи щодо захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.

- Розробка та реалізація нормативно-правової бази у сфері кібербезпеки. Державні органи розробляють та реалізують нормативно-правову базу у сфері кібербезпеки, яка спрямована на забезпечення захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.
- Підготовка та підвищення кваліфікації кадрів у сфері кібербезпеки. Держава забезпечує підготовку та підвищення кваліфікації кадрів у сфері кібербезпеки, які здатні забезпечити захист життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.
- Розвиток міжнародного співробітництва у сфері кібербезпеки. Держава розвиває міжнародне співробітництво у сфері кібербезпеки, яке спрямоване на забезпечення захисту життєво важливих інтересів людини і громадянина, суспільства та держави, національних інтересів України у кіберпросторі.

Держава продовжує вживати заходів щодо забезпечення кібербезпеки України. У планах держави:

- Подальше формування та розвиток системи кібербезпеки України.
- Удосконалення нормативно-правової бази у сфері кібербезпеки.
- Посилення підготовки та підвищення кваліфікації кадрів у сфері кібербезпеки.
- Розвиток міжнародного співробітництва у сфері кібербезпеки.

1.7. Висновки до розділу

На першому етапі розвитку системи кібербезпеки, серед найважливіших заходів, враховується розробка та вдосконалення нормативно-правової бази. Це включає створення ключових компонентів національної системи кібернетичної безпеки, підготовку Збройних Сил України до можливих кіберзагроз, формування бази для навчання спеціалістів, які будуть захищати сектор безпеки та критичну інформаційну інфраструктуру держави. Також на цьому етапі важливо встановити норми та умови співробітництва між державним і приватним секторами щодо протидії кіберзлочинності та забезпечення інформаційної безпеки. Приділяється значна увага інформаційним заходам для освіти громадян і бізнесу в сфері кібербезпеки.

Важливим є необхідність постійного вдосконалення заходів та політики кібербезпеки, оскільки швидкі зміни технологій та загроз роблять інформаційні ресурси дедалі вразливішими. Також важливим є впровадження ефективних заходів захисту, включаючи технічні, організаційні та правові аспекти.

Тому забезпечення кібербезпеки інформаційних ресурсів є критично важливим завданням, яке вимагає системного та комплексного підходу. Досягнення високого рівня кібернетичної безпеки стає дієвим заходом для забезпечення стабільності та надійності інформаційних систем у сучасному цифровому середовищі.

РОЗДІЛ 2. ОРГАНІЗАЦІЯ КІБЕРНЕТИЧНОЇ БЕЗПЕКИ НА БАЗІ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ

2.1. Застосування криптографічних методів для захисту інформаційних ресурсів.

Зі збільшенням використання обміну даними та комунікації через Інтернет, захист даних від кібератак стає критично важливим. На сьогоднішній день забезпечення конфіденційності та приватності даних є серйозним викликом для дослідників і фахівців у сфері кібербезпеки. Конфіденційність даних означає захист даних від несанкціонованого доступу або крадіжки. Вона може бути досягнута за допомогою криптографії шляхом шифрування та дешифрування даних. Метою криптографії є захист критично важливих даних або документів на жорсткому диску або під час передачі через незахищений канал зв'язку. Двома основними видами шифрування є симетричне шифрування та асиметричне шифрування. У симетричному шифруванні використовується лише один ключ, і всі сторони користуються одним і тим самим секретним ключом. В асиметричному шифруванні використовується декілька ключів: один ключ для шифрування, а інший для дешифрування. Ключ шифрування називають «відкритим ключем», а ключ дешифрування — «закритим ключем».

Шифрування даних - це мистецтво захисту повідомлень шляхом перетворення їх на приховані тексти, тоді як зворотний процес відновлення оригінальних текстів з прихованих текстів називається дешифруванням. Шифрування/розшифрування стає можливим за допомогою певних ключів. Кожен алгоритм шифрування намагається максимально ускладнити процес розшифрування без допомоги ключа, який використовувався при шифруванні. На рис. 2. 1 показано загальну ідею шифрування та дешифрування.

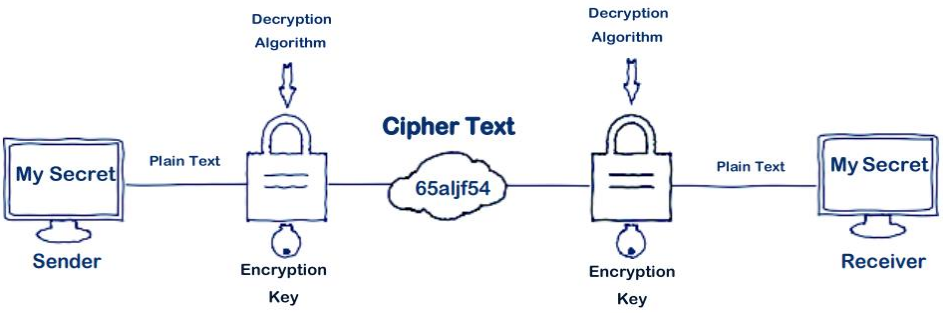


Рис. 2.1. Загальна ідея шифрування та дешифрування

2.2. Методи симетричного шифрування

Цей метод передбачає шифрування та дешифрування за допомогою одного ключа, який називається приватним ключем. Його також називають секретним ключем. Для обміну цим приватним ключем між відправником і одержувачем потрібен захищений канал. Криптографічні алгоритми з симетричним ключем поділяються на два типи залежно від вхідних даних: блокові шифри та потокові шифри. У системах на основі блокових шифрів дані обробляються або шифруються групою бітів фіксованої довжини, яка називається блоком, тоді як у системах на основі потокових шифрів дані обробляються потоком бітів. Рис. 2.2 ілюструє процес симетричного шифрування.

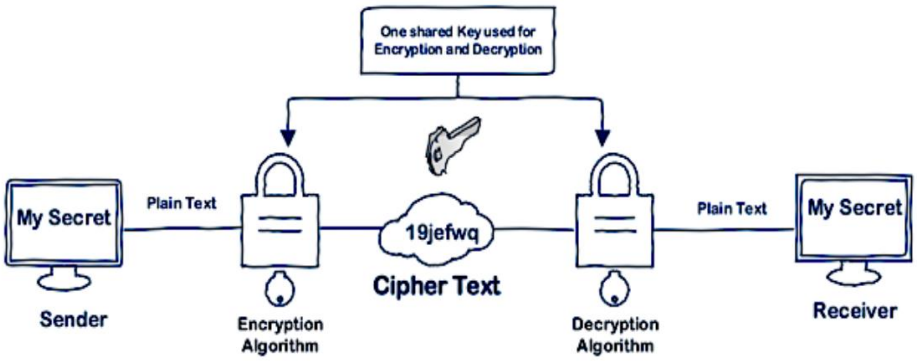
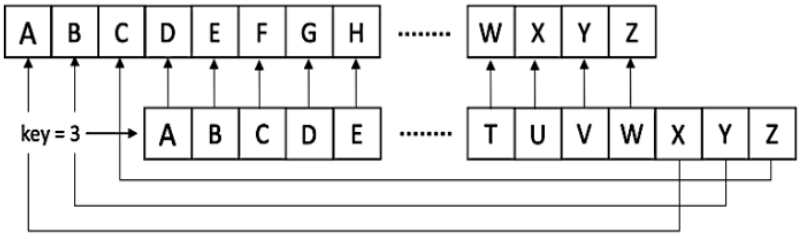


Рис. 2.2 Схема симетричного шифрування

У криптографії використовується багато методів шифрування. У цьому розділі ми детально розглянемо деякі поширені алгоритми шифрування на основі потокових і блокових шифрів, а також пояснимо різні режими шифрування на основі блокових шифрів.

2.2.1 Шифр Цезаря

Шифр Цезаря [1], [2] є однією з найпростіших схем симетричного блочного шифрування, тому його легко зламати. Римський правитель Юлій Цезар створив і використовував цю схему шифрування для передачі військових наказів своїм легіонам. Це шифр заміни, в якому ключі шифрування і розшифрування збігаються. Ключі, що використовуються в цій схемі, є цілими числами, і найчастіше використовується число 3. У цій техніці шифрування кожен алфавіт зсувається вправо або вліво на значення ключа, як показано на рис. 2.3 (з ключем = 3).



Key	3
Plain Text	MY INFORMATION
Cipher Text	PB LQIRUPDWLRQ

Рис. 2.3. Метод зсуву в шифрі Цезаря

2.2.2 Data Encryption Standard (DES)

DES [3], [4] - один з базових алгоритмів блочного шифрування з симетричним ключем, який використовує відкриті тексти у вигляді блоків, кожен з яких містить 64 біти, і перетворює зашифровані тексти за допомогою ключів довжиною 64 біти. З цих 64 біт 8 біт ключа використовуються для непарної парності, яка не враховується в довжині ключа. Таким чином, існує 2^{56} можливих способів знайти правильний ключ. Алгоритм DES виконує дві перестановки

(початкову і кінцеву) і 16 кроків обробки, кожен з яких називається раундом, і для кожного раунду використовується свій ключ. В основі DES лежать дві криптографічні операції: підстановка і транспозиція. У кожному раунді DES виконується певна кількість підстановок і транспозицій. Перед початком першого раунду до відкритого тексту застосовується початкова перестановка. Наприклад, початкова перестановка замінює перший біт відкритого тексту на 58-й біт, а другий - на 50-й біт і так далі. Отриманий в результаті перестановки блок ділиться на дві половини, кожна з яких має 32 біти, і кожна з них проходить через 16 раундів шифрування. Фінальна перестановка застосовується до об'єднаного блоку для отримання зашифрованого тексту. DES був визнаний вразливим і тому був замінений на 3DES [6]. Загальний принцип роботи DES пояснюється на рис. 2.4 [5].

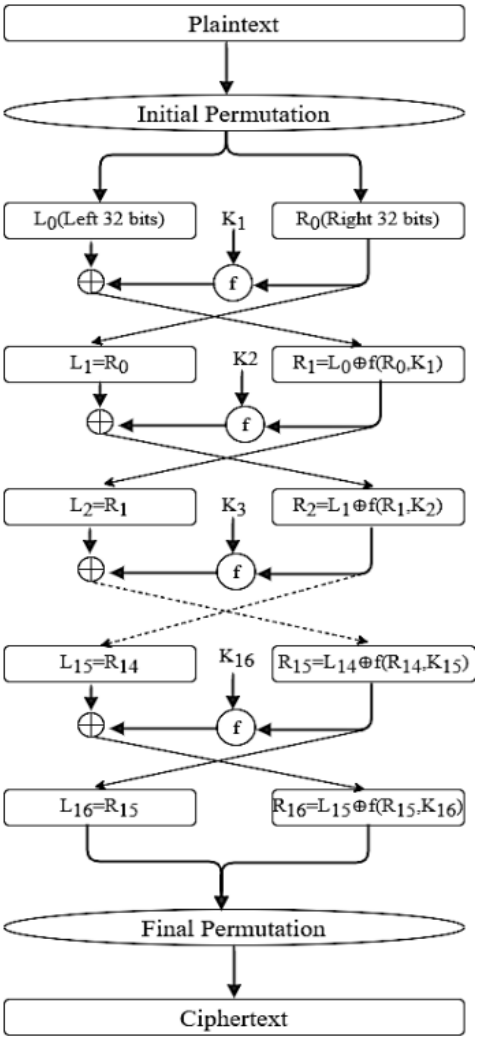


Рис. 2.4. Процес роботи DES-алгоритму

2.2.3 Triple Data Encryption Standard (3DES)

Triple-DES [6] - це алгоритм шифрування блочного шифру. Як видно з назви, 3DES тричі застосовує DES до кожного блоку даних, щоб підвищити безпеку зашифрованих даних. Оскільки безпека 3DES втричі вища, ніж у DES, він вважається кращим, ніж DES. Однак, у порівнянні зі своїм попередником, він споживає значну кількість часу.

3DES працює так само, як і DES, в циклі довжиною 3. Спочатку вихідний відкритий текст шифрується одним ключем, отриманий зашифрований текст знову шифрується за допомогою іншого ключа, і, нарешті, це знову виконується з третім ключем.

2.2.4 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) [7] - алгоритм блочного шифрування, який прийшов на заміну DES і Triple DES. Він шифрує і розшифровує 128-бітний блок даних. Залежно від вибору розміру ключа, 128 біт, 196 біт або 256 біт, AES може використовувати 10, 12 або 14 раундів для шифрування. Кожен раунд складається з чотирьох операцій: заміна байтів, клавіші зсуву, перемішування стовпця і додавання круглого ключа. Однак, операція перемішування стовпця не виконується в останньому раунді. У кожному раунді шифрування використовуються окремі раундові ключі, згенеровані із заданого ключа шифру. Дані, які потрібно зашифрувати, розбиваються на блоки. Кожен блок представляється у вигляді масиву даних, який називається масивом станів. AES не є вразливим, як DES, і, як відомо, забезпечує хороший рівень безпеки. Процес шифрування AES показано на рис. 2.5.

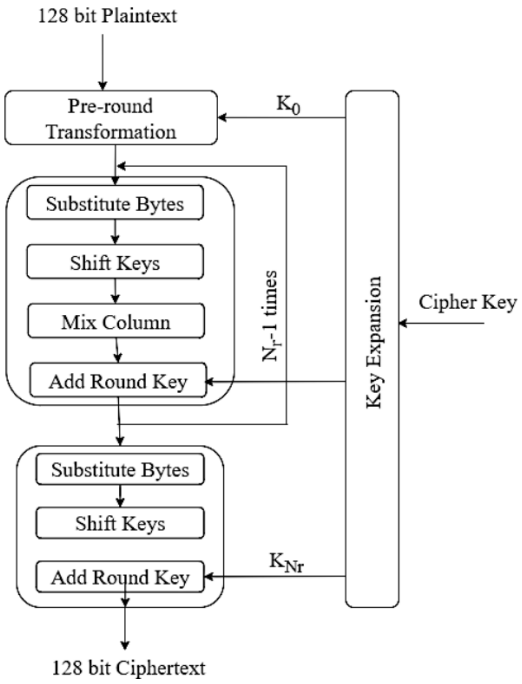


Рис. 2.5. Процес шифрування за допомогою AES

2.2.5 BlowFish

BlowFish [8] - це алгоритм шифрування на основі блочного шифру, довжина ключа якого варіюється від 32 до 448 біт. Кожен блок обробляє 64 біти даних. BlowFish шифрує дані за допомогою 16 раундів операцій. [9] На кожному раунді дані проходять залежну від ключа перестановку в P-блоці та підстановку в S-блоці. Кожен S-блок містить 32 біти даних. На рис. 6 показано функцію F BlowFish, яка розбиває 32-бітові дані на чотири чверті, кожна з яких містить 8 біт. Ці чверті будуть входами для S-блоку. У S-блоках виконуються операції XOR і MOD 2^{32} для отримання кінцевих зашифрованих даних. Зворотний процес виконується для розшифрування даних.

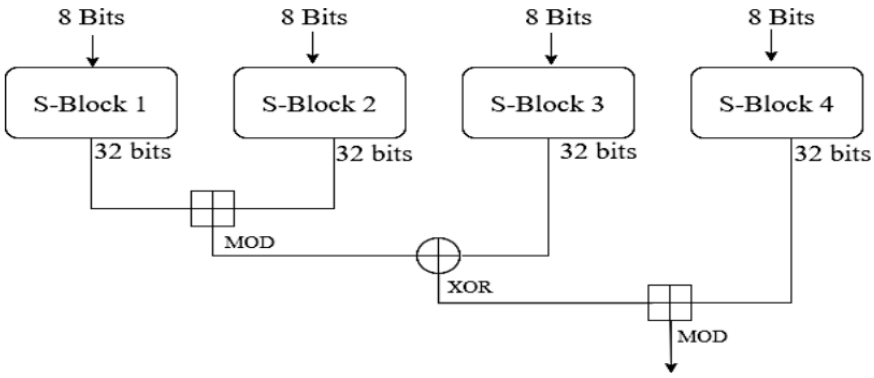


Рис. 2.6. Функція F алгоритму BlowFish

2.2.6 Twofish

Twofish [10] - це також система симетричного шифрування на основі блочного шифру, яка працює подібно до BlowFish. Однак, на відміну від BlowFish, Twofish вважається більш гнучкою. Twofish дозволяє користувачам налаштовувати швидкість шифрування, час встановлення ключа, розмір коду і швидко працює на 8-розрядних процесорах, а також на смарт-картах, вбудованих чіпах тощо. Програма є вільно доступною для використання, оскільки це незапатентоване, безліцензійне програмне забезпечення. Twofish шифрує документи з розміром блоку 128 біт з розмірами ключів 128, 198 або 256 біт за 16 раундів шифрування. Структурні блоки Twofish показані на рис. 2.7.

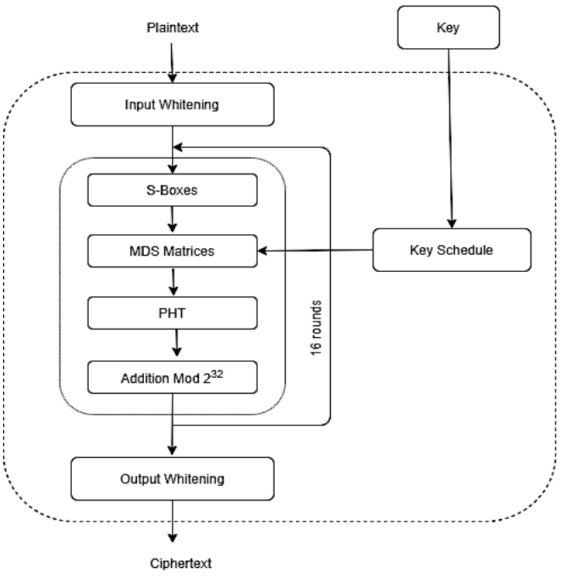


Рис. 2.7. Структурні елементи Twofish

Фактична обробка кожного раунду Twofish починається і закінчується попереднім відбілюванням і пост-відбілюванням (тобто текстові блоки об'єднуються за допомогою XOR з додатковими підключами), відповідно. На вхід функції F подаються два 32-бітних слова, які розбиваються на чотири байти і надсилаються до чотирьох різних S-блоків, залежних від ключів. Виходи цих чотирьох S-блоків об'єднуються за допомогою матриці максимальної відстані, що розділяється (MDS), щоб сформувати 32-бітне слово. Потім ці два 32-бітних слова об'єднуються за допомогою псевдоперетворення Хадамара (PHT), до них додаються два круглих підключача, а потім до них додається права половина тексту за допомогою операції XOR. До і після операції XOR виконується обертання на 1 біт. Після повторення цих раундів 16 разів, остання заміна змінюється на протилежну, і виконується операція XOR між чотирма ключовими словами з іншими чотирма ключовими словами, щоб отримати остаточний зашифрований текст.

2.2.7 Threefish

Threefish [11] - це стандарт шифрування на основі блочного шифру з можливістю налаштування, який приймає три вхідні дані: ключ, налаштування та звичайний текст, який потрібно зашифрувати. Threefish використовує для шифрування блоку даних ключ тієї ж довжини, що і розмір блоку даних. Цей метод шифрування використовується для блоків даних розміром 256, 512 і 1024 біт. Схема Threefish створює зашифровані дані шляхом повторення однієї і тієї ж послідовності операцій 72 рази (або раунди), за винятком 1024-бітових блоків даних, для яких потрібно 80 раундів. Для всіх цих розмірів блоків даних використовується 128-бітове значення підстроювання. Операції стандарту шифрування Threefish бувають трьох типів: додавання, XOR та обертання. Threefish також є безкоштовним для користувачів, оскільки це незапатентований

і вільний від ліцензій стандарт шифрування. На рис. 2.8 детально показано, як працює кожен раунд Threefish-256.

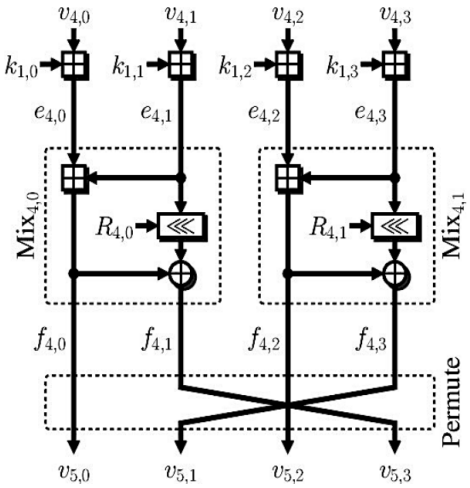


Рис. 2.8. Схема роботи одного раунду алгоритма Threefish-256

2.2.8 International Data Encryption Algorithm (IDEA)

IDEA [12] - це алгоритм блочного шифрування, який обробляє 64-розрядні блоки даних за допомогою 128-розрядного ключа. Цей 64-бітний блок даних розбивається на чотири рівні підблоки, кожен розміром 16 біт. Кожен з цих підблоків проходить вісім раундів повторюваних послідовностей операцій і одну фазу перетворення на виході. Для кожного раунду роботи системі потрібно шість унікальних ключів, які генеруються з 128-бітового оригінального ключа. Вихід кожного раунду подається на вхід наступного раунду, за винятком восьмого раунду. Вихід восьмого раунду подається на фазу перетворення виходу, яка виконує лише арифметичні операції і потребує чотирьох ключів. Фаза перетворення вихідних даних виробляє остаточний ключ шифру. Весь процес шифрування потребує 52 ключі. Процес роботи IDEA зображено на рис. 2.9.

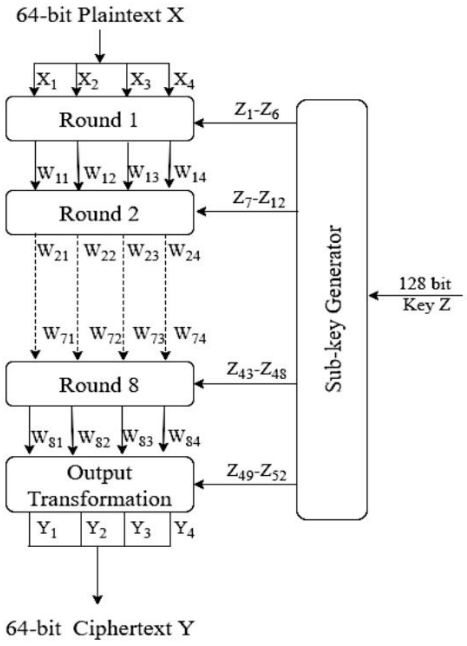


Рис. 2.9. Процес роботи алгоритму шифрування IDEA

2.3. Методи асиметричного шифрування.

Метод асиметричного шифрування передбачає використання декількох ключів для шифрування та дешифрування. Використовуються два пов'язані між собою ключі шифрування, перший з яких називається відкритим ключем, а другий - приватним ключем. Відкритий ключ доступний будь-кому, хто хоче відправити повідомлення відправнику. Другий приватний ключ залишається секретним, його знає лише відправник. Повідомлення, зашифроване відкритим ключем, можна розшифрувати за допомогою приватного ключа. Повідомлення, зашифроване приватним ключем, можна розшифрувати за допомогою відкритого ключа. Безпека відкритого ключа не вимагається, оскільки він доступний і може передаватися через Інтернет. Асиметричне шифрування вважається найкращим вибором для передачі інформації. Асиметричне шифрування використовується в клієнт-серверній моделі зв'язку, сертифікат створюється для визначення місцезнаходження сервера, сертифікат містить профіль організації та інформацію, сервер і клієнт потребують безпечного зашифрованого зв'язку, запит надсилається через мережу іншій стороні, яка

повертає їм сертифікат назад. SSL/TLS використовує асиметричне та симетричне шифрування.

Перші ідеї асиметричного шифрування з'явилися в 1976 році в роботі "Нові напрямки в сучасній криптографії" В. Діффі та М. Геллмана. У своєму дослідженні вони представили новий метод організації безпечного обміну інформацією без попереднього обміну ключами.

2.3.1 Метод Діффі-Геллмана

У 1976 році В. Діффі та М. Геллман представили криптографічну систему з відкритим ключем, що базується на складності обчислення дискретного логарифма. Цей метод використовується для створення ключів, їх розподілу та для комутативних цілей, хоча не підходить для шифрування [13].

Існує також варіація цього методу, яку вперше висунули Ш. МакКерлі та К. МакКерлі. Вони внесли ідею використання модуля у вигляді складеного числа. Пізніше М. Кобліц та Н. Кобліц розширили цю версію методу, введучи використання еліптичних кривих.

Метод Ель-Гамаля, який на відміну від методу Діффі-Геллмана, придатний для шифрування та цифрового підпису, також базується на складності обчислення дискретного логарифма. Це був перший алгоритм асиметричного шифрування, який підходив для обробки як повідомлень, так і цифрових підписів, і не був обмежений патентами США.

Важливою особливістю цього методу є його непридатність для використання на незахищених від модифіцікацій лініях зв'язку, оскільки метод Діффі-Геллмана вразливий до атаки "людина посередині".

За цим алгоритмом, учасники обміну повідомленнями, позначені як А та В, узгоджують значення великого простого числа p та його простого дискретного кореня, числа a. Учасник А вибирає число k_a, Учасник В – число k_b, дотримуючись умов $1 < k_a < p - 1$ та $1 < k_b < p - 1$. Засекречені числа k_a та k_b

зберігаються учасниками А та В. Учасник А створює відкритий ключ за допомогою формули

$$Y_a \equiv a^{ka} \pmod{p}. \quad (2.1)$$

Так само учасник В створює відкритий ключ за допомогою формули

$$Y_b \equiv a^{kb} \pmod{p}. \quad (2.2)$$

Після взаємного обміну відкритими ключами Y_a та Y_b, учасники розраховують значення спільного секретного числа K:

$$K \equiv Y_a^{k_b} \pmod{p} \equiv a^{k_a k_b} \pmod{p}; \quad (2.3)$$

$$K \equiv Y_b^{k_a} \pmod{p} \equiv a^{k_b k_a} \pmod{p}. \quad (2.4)$$

Число *K,* яке отримали, є секретним для потенційного зловмисника, адже розв'язання рівнянь Y_a та Y_b для великих чисел не є можливим. Метод Діффі-Геллмана можна використовувати з трьома і більше учасниками.

2.3.2 Метод Меркле-Геллмана

У 1978 році Р. Меркл та М. Геллман представили новий метод асиметричного шифрування загального призначення. Ця криптосистема відноситься до ранцевих алгоритмів. Варіант Методу Меркла-Геллмана був придатний лише для шифрування повідомлень, і згодом А. Шамір представив модифікацію, яка крім шифрування підтримує також функцію цифрового підпису.

Безпека ранцевих алгоритмів ґрунтується на відомій математичній задачі. Суть такого алгоритму полягає в шифруванні повідомлення через вирішення комплексу задач, пов'язаних із укладанням ранця. З урахуванням того, що розв'язок цієї задачі для невеликої кількості елементів є досить простим, для практичного застосування розмір ранця повинен бути більше 250 елементів, і кожен елемент повинен мати довжину від 200 до 400 біт, а довжина модуля алгоритму - від 100 до 200 біт. Для того, щоб отримати числа із вказаними розмірами використовувалися генератори випадкових чисел.

Цей метод створення криптосистем виявився недостатньо надійним. Шамір та Циппель успішно зламали криптосистему Меркле-Геллмана, виявивши вразливості та відновивши швидкозростаючу послідовність ранця в своїх дослідженнях. Хоча в подальшому було розроблено кілька модифікацій криптосистем, що базуються на задачі про укладення ранця, жодна з них не була стійкою до криптоаналітичних атак.

2.3.3 Метод RSA

Після створення першої асиметричної криптосистеми у 1978 році Меркле-Геллмана, Рон Рівест, Аді Шамір і Леонар Адлеман опублікували статтю, у якій представили алгоритм із публічним ключем, придатним як для шифрування, так і для формування цифрового підпису [14]. Цей алгоритм отримав назву за ініціалами його творців. Він ефективно замінив менш захищений на той час алгоритм DEC.

У наш час криптосистема RSA широко використовується як у комерційних, так і у некомерційних продуктах у різних галузях. Вона застосовується в операційних системах компаній, таких як Apple, Microsoft, Sun, Novell. Метод RSA також застосовується у захищених телефонах, мережевих платах Ethernet, мережевому протоколі SSH, у протоколі захисту SSL та криптографічній програмі PGP.

2.3.4 Криптосистеми на еліптичних кривих

Криптосистеми, які використовують еліптичні криві, відносяться до групи асиметричних криптосистем. Безпека цих систем базується на складності вирішення задачі дискретного логарифму у групі точок еліптичної кривої, яка існує над скінченим полем [15]. Це забезпечує високий рівень криптостійкості.

Деякі варіанти криптосистем на еліптичних кривих ґрунтуються на складності факторизації великих цілих чисел, особливо тоді, коли еліптична крива задається над скінченим полем за складеним модулем. Однак такі системи зустрічаються вкрай рідко.

У криптографії для використання, зазвичай, вибирають скінченні поля. Для точок, розташованих на еліптичній кривій, визначається операція додавання, яка виконує ту ж саму функцію, що і операція множення в методах RSA та Ель-Гамаля.

Головною перевагою криптосистем, що використовують еліптичні криві, є висока швидкість обробки інформації, проте для досягнення цього результату необхідно використовувати алгоритми швидкого обчислення.

Вищий рівень криптостійкості відповідних систем дозволяє використовувати ключі шифрування меншої довжини, ніж інші види асиметричних криптосистем. Це можна пояснити тим, що для обчислення обернених функцій на еліптичних кривих можна застосовувати лише алгоритми з експоненціальним зростанням обчислювальної складності, в той час як для інших асиметричних криптосистем також придатні субекспоненціальні методи.

Криптосистеми, які базуються на еліптичних кривих, представляють собою найбільш поширений метод для створення цифрового підпису, а їх використання впроваджено у стандарти по всьому світу наприкінці 90-х років.

Безпека цих систем ґрунтується не тільки на стійкості алгоритму, що використовує еліптичні криві, але й на стійкості використовуваної геш-функції, яка застосовується для підготовки документів до процедури шифрування з метою зменшення їх об’єму.

2.4. Класичний метод асиметричмного шифрування RSA

Для наступних етапів дослідження вибрано асиметричний метод шифрування, а саме RSA, оскільки він відзначається високою стійкістю. Також

важливо відзначити, що цей метод є одним з найпоширеніших, тому будь-яка модифікація, розроблена на його основі, буде сумісною з традиційним підходом RSA.

Метод RSA був розроблений незадовго до активного розповсюдження електронної пошти. Він втілював такі важливі ідеї [16]:

- Шифрування із використанням публічного ключа забезпечує можливість обходу необхідності в наявності окремого безпечного каналу для передачі ключів користувачів перед початком обміну повідомленнями. Це можливо завдяки тому, що у методі RSA закриті ключі не вимагають спеціального каналу для їх передачі.
- Іноді виникає необхідність у перевірці автора повідомлення. Для цього автор використовує свій цифровий підпис, який генерується за допомогою закритого ключа для шифрування, а перевіряється за допомогою відкритого ключа. Цей механізм гарантує непідробність цифрових підписів і, крім того, унеможливлює можливість спростування авторства повідомлення в майбутньому. Така особливість також застосовується в сфері електронних фінансових операцій.

Обмін повідомленнями реалізується відповідно до такої концепції: припустимо, що Аліса прагне надіслати повідомлення Бобу. З цією метою Боб генерує на своєму боці відкритий і закритий ключі шифрування. Далі Боб розповсюджує свій відкритий ключ у мережі, щоб будь-який користувач, який планує відправити йому повідомлення, зміг його зашифрувати. З огляду на те, що закритий ключ відомий лише Бобу, жодна інша особа не зможе прочитати зашифроване повідомлення. Коли Аліса отримала відкритий ключ від Боба, зашифровує ним своє повідомлення та висилає його Бобу, який, у свою чергу, розшифровує повідомлення своїм ключем. Всі асиметричні криптосистеми працюють за подібною схемою.

Для генерації цифрового підпису Боб використовує свій закритий ключ для шифрування повідомлення, що при цьому служить для нього унікальним

ідентифікатором. Потім Аліса, яка отримала відкрите повідомлення та його підпис, розшифровує підпис за допомогою відкритого ключа. Якщо розшифрована інформація співпадає з оригінальним повідомленням, цифровий підпис вважається дійсним.

З метою полегшення процесу підписування документів, їх передусім піддають обробці геш-функцією.

2.4.1 Математичні основи алгоритму RSA

Алгоритм шифрування RSA розпочинається з визначення відкритого ключа, який представляє собою пару позитивних цілих чисел (e, n). Цей етап включає в себе вибір двох великих простих чисел p і q, які залишаються конфіденційними [17]. Число n визначається як їхній добуток, тобто $n = pq$. Крім того, розраховується функція Ейлера:

$$\varphi(n) = (p - 1)(q - 1) \tag{2.5}$$

Потім випадковим чином обирається просте число e, яке слугуватиме ключем шифрування і відповідає умовам $e < \varphi(n)$ та НСД $(e, \varphi(n)) = 1$. Для ключа розшифрування формується пара чисел (d, n), де число d є оберненим до числа e:

$$d \equiv e^{-1} (\text{mod } \varphi(n)). \tag{2.6}$$

Для шифрування повідомлення його слід конвертувати у ціле число від 0 до $n - 1$. У випадку, якщо розмір представленого числа більший, повідомлення розбивається на цифрові блоки, що шифруються окремо. Наприклад, якщо розмір n становить 256 біт, кожен цифровий блок повідомлення не повинен перевищувати 255 біт. Ця сама довжина буде залишатися і після шифрування завдяки використанню модульної конгруенції.

Для отримання шифротексту C, повідомлення M зашифровується шляхом застосування операції піднесення до ступеня e за модулем n:

$$C \equiv M^e (\text{mod } n). \tag{2.7}$$

Розшифрування відбувається відповідно до теореми Ейлера, використовуючи операцію піднесення до ступеня d за модулем n:

$$M \equiv C^d \pmod{n}. \quad (2.8)$$

2.4.2 Реалізація алгоритму RSA

В алгоритмі RSA використовується піднесення до ступеня, що реалізується через послідовні операції піднесення до ступеня квадрату і ділення [18].

Здійснюється такий послідовний процес:

1. Припустимо, що бінарний вигляд e можна представити у вигляді $e_k e_{k-1} \ldots e_1 e_0$.
2. Встановлюється значення для змінної C = 1.
3. Для $i = k, k - 1$ виконуються наступні етапи:
   - C присвоюється залишок від ділення C^2 на n.
   - У випадку, коли $e_1 = 1$, то C присвоюється значення, що дорівнює залишку від ділення добутку $C \times M$ на n.
4. Отримуємо зашифроване повідомлення C.

Для того, щоб зашифрувати повідомлення, тобто для операції $M^e \pmod{n}$, потрібно провести не більше, ніж $2\log_2(e)$ операцій ділення. Цей показник визначає необхідний час для виконання відповідних операцій. Потрібний час для зашифрування одного блоку не зростає швидше, ніж квадрат довжини числа n.

2.4.3 Оцінка надійності алгоритму RSA

Безпека методу RSA ґрунтується на ускладненості завдання факторизації числа n, тобто його розкладання на прості множники. На сьогоднішній день відсутнє ефективне рішення цієї обчислювальної задачі. Це забезпечує надійність методу RSA. Однак твердження про те, що його безпека ґрунтується

на складності факторизації великих чисел, залишається гіпотетичним, оскільки теоретично можлива ефективніша процедура факторизації із поліноміальною складністю на звичайному комп'ютері. На даний час найефективніший алгоритм факторизації довільних цілих чисел має субекспоненціальну часову оцінку. Це означає те, що теоретична кількість кроків для завершення цього алгоритму великою мірою перевищує значення полінома ln *n*. Ця особливість забезпечує досягнення задовільного рівня безпеки в протоколі шифрування RSA при використанні числа n, що є ключовим для практичного застосування цього методу.

Числа, що використовуються як *n*, відомі як RSA-числа. Це сукупність великих напівпростих чисел, які можна представити як добуток двох простих чисел. Довжина числа *n* зазвичай складає 2048 бітів чи більше [19].

2.5. Аналіз криптосистеми методу RSA.

На сьогоднішній день було проведено значну кількість досліджень щодо потенційно ефективних методів криптографічного аналізу системи RSA. Це питання регулярно входить до списку обговорюваних тем на авторитетних щорічних конференціях з криптографії, таких як, CRYPTO, EUROCRYPT, ASIACRYPT.

2.5.1 Методи факторизації для криптоаналізу методу RSA.

Одним із найбільш перспективних напрямів для вирішення задачі криптоаналізу методу RSA є застосування обчислювальних методів факторизації. Серед цих методів виділяють чотири основні:

- метод Ферма - використовується для виявлення ключів, у яких множники *p* і *q* мають близькі значення один до одного;

- *P*-метод Полларда - використовується для виявлення невеликих простих дільників ключа перед застосуванням більш потужних алгоритмів факторизації;
- метод решета числового поля - найефективніший спосіб факторизації чисел з більш ніж 110 знаками у десятковій системі. Цей метод був успішно використаний для факторизації найбільших чисел;
- метод квадратичного решета - наступний за швидкістю після методу решета числового поля і є більш простим за нього.

Методи квадратичного решета і решета числового поля ґрунтуються на виявленні гладких чисел відносно порядку √n. З метою оптимізації обчислень вони застосовуються в числових полях, що робить процес більш складним для даного алгоритму.

2.5.2 Непрямі методи криптоаналізу RSA

Через високий рівень складності математичного апарату криптографічних алгоритмів було розроблено інакший підхід до атак на них - використовування непрямих методів [20]. Такі методи атак використовують різні вразливості криптографічних алгоритмів, які виникають при їх неправильній реалізації, наприклад:

- неправильний вибір публічної або приватної експоненти;
- неправомірне використання системи;
- послідовності у шифрованому тексті.

Прикладом схожої атаки може бути атака Вінера, яка ґрунтується на використанні наближених дробів. Зазначений метод був розроблений для злому RSA в умовах, коли значення закритого ключа d є невеликим. Автор стверджував, що при виконанні такої умови:

$$d < \frac{1}{3} n^{\frac{1}{4}}, \tag{2.9}$$

за лінійний час можна знайти закритий ключ.

Атака виконується у такий спосіб:

1. Для початку треба розкласти $\frac{e}{n}$ у ланцюговий дріб [a_1, a_2…].
2. Для цього дробу [a_1, a_2…] треба знайти множину всіх дробів $\frac{k_n}{d_n}$.
3. Дослідити дріб, який буде підходити $\frac{k_n}{d_n}$.
   1) Знайти значення $\varphi(n)$, шляхом обчислення $f_n = \frac{ed_n - 1}{k_n}$.
   2) Обчислити $x^2 - ((n - f_n) + 1)x + n = 0$, знайшовши корені ($p_n$, q_n).
4. Якщо добуток отриманих значень коренів (p_n, q_n) дорівнює n, то атака вважається успішною, і подальші кроки дозволяють отримати закритий ключ d. У випадку, якщо ця умова не виконується, обирається новий дріб $\frac{k_{n+1}}{d_{n+1}}$, у якого виконується крок 3.

Час виконання алгоритму Вінера складає O(log(n)), що означає, що відновлення закритого ключа займає лінійний час і залежить від довжини n.

Ще один надзвичайно цікавий клас алгоритмів вимагає знання декількох найбільш або найменш значущих бітів закритого ключа. Припустимо, що криптоаналітик володіє певним значенням d_0 для конкретного відомого повідомлення M, таким чином, що $d \equiv d_0 \bmod M$. У такому випадку можна виразити закритий ключ так:

$$d = d_1 M + d_0, \tag{2.10}$$

для нас єдиним невідомим є d_1. Потім вводиться $M = n^{\beta - \delta}$, і невідома частина закритого ключа d_1 обмежується $|d_1| < n^{\beta}$.

Такий самий підхід використовується при знанні найбільш або найменш значущих бітів відкритого ключа.

Приклади успішних атак на алгоритм RSA часто базуються на використанні непрямих методів, хоча в загальному випадку вони не виявляються більш ефективними в порівнянні з завданням криптоаналізу, побудованим на математичній проблемі складності факторизації великих чисел, що є основою криптомодуля методу RSA.

2.5.3 Методи криптоаналізу RSA по стороннім каналам

Інший тип атак базується на використанні побічних каналів для отримання інформації про криптосистеми, включаючи дані про енергоспоживання, виникнення збоїв та інші процеси, що відбуваються під час виконання алгоритму. Ці атаки відомі як SCA (Side Channel Attacks) [21].

Приховані чи сторонні канали включають два чи більше процеси, які взаємодіють через спільний ресурс, на який вони впливають. Зловмисники використовують ці канали для ухилення від захисту операційної системи. На відміну від інших методів криптоаналізу, SCA аналізує не самі алгоритми шифрування, а його конкретну реалізацію. У такий спосіб, досліджують кореляцію між характеристиками, такими як споживання енергії та час виконання алгоритму, та внутрішнім станом процесів, що пов'язані із закритим ключем всередині пристрою. Такі атаки є простішими та ефективнішими порівняно з традиційним криптоаналізом, що базується на математичному аналізі.

Однією з перших розроблених атак SCA є часова атака. Це відбувається, коли сторонній канал передає інформацію про час, який витрачається на виконання конкретної операції. Припустимо, криптоаналітик використовує часову атаку на функцію перевірки пароля, яка реалізована через порівняння рядків. Він дивиться за тим, як система завантажує введений рядок у пам'ять і послідовно перевіряє кожен байт. Кількість порівнянь, які виконує функція, прямо пов'язана з кількістю байтів, які збігаються у рядку.

Інша атака SCA базується на аналізуванні потужності. Сторонній канал для такого типу атаки представлений струмом, який протікає через транзисторні переходи, коли транзистор увімкнено. У такий спосіб, криптоаналітик спостерігає за змінами в режимі роботи шифрувального пристрою через зміну загального струму. Так як транзистори у відкритому стані збільшують кількість

струму, а в закритому стані - навпаки, зміна цієї кількості струму вказує на зміни внутрішнього стану пристрою. Інформація про зміни та закономірності в споживаній електроенергії надає детальне уявлення про низькорівневі процеси всередині шифрувального пристрою.

Для проведення атаки на алгоритм RSA використовується аналіз модульного піднесення до ступеня. Один із широко використовуваних алгоритмів піднесення до квадрата та множення включає побітове зчитування ступеня зліва направо. Починаючи з регістра, де встановлено 1, виконується операція піднесення до квадрата до першого біта, який дорівнює 0, а потім виконується операція множення. У такий спосіб, в споживаній електроенергії можна виявити послідовність операцій, розділених спадаючими піками. Нижче споживання електроенергії відповідає операціям піднесення до квадрата, а високе - множенню. Проводячи аналогію між отриманим сигналом і відповідною залежністю вхідного біта та відповідною операцією, можна визначити реакцію на експоненту.

Кількість неправильних операцій, що виникають під час функціонування шифруючого модуля, також представляє собою значущий сторонній канал для дослідження криптоаналітиками. При використанні цього типу атаки аналізуються дві категорії помилок:

- обчислювальні помилки, які виникають внаслідок невірної реалізації криптографічного модуля чи в результаті зовнішніх втручань у напругу;
- помилки, які викликані умисним введенням пошкоджених вхідних даних у криптографічний модуль, який піддається атаці.

Електромагнітний аналіз є таким типом атаки. Щоб його провести криптоаналітику потрібно, замість вимірювання енергоспоживання шифрувального пристрою, збирати дані щодо його електромагнітних випромінювань. Зазвичай такий вид аналізу надає більш деталізовану інформацію, ніж аналіз потужності.

2.6. Модифікації методу RSA

Після появи односторонніх функцій та їх використання в методі RSA, стали виникати варіації цього методу.

2.6.1 Модифікація RSA з використанням Китайської теореми про залишки

Дж.-Дж. Квісквотером та К. Коуврером у 1982 році представили нову схему розшифрування для алгоритму RSA, яка отримала назву методу Квісквотера-Коуврера. Цей метод використовує наслідок з Китайської теореми про залишки [22].

Під час дешифрування повідомлення за допомогою алгоритму RSA виконується операція піднесення до ступеня за модулем n. Метод Квісквотера-Коуврера дозволяє зменшити кількість таких операцій, виконуючи замість операції за формулою дві аналогічні:

$$m_p \equiv C^d \pmod{p}, \qquad (2.11)$$
$$m_q \equiv C^d \pmod{q}.$$

Так як числа p і q мають порядок, що в 2 рази менший, ніж число n, можна значно ефективніше виконати 2 піднесення до ступеня за модулем $\frac{n}{2}$, ніж одне піднесення до ступеня за модулем n. По завершенню, необхідно відновити повідомлення M, скориставшись m_p і m_q, що здійснюється через наслідок з Китайської теореми про остачі.

Теорема формується так. Якщо натуральні числа $a_1, a_2,\ldots,a_n$, які є попарно взаємнопростими, то для цілих $r_1, r_2,\ldots,r_n$, таких, що $0 \leq r_i \leq a_i$ для всіх $i \in \{1, 2,\ldots,n\}$, знайдеться число N, що при діленні на a_i дає залишок r_i при всіх $i \in \{1, 2,\ldots,n\}$. Окрім того, якщо будуть знайдені два такі числа N_1 та N_2, що задовольняють попередню умову, то $N_1 \equiv N_2 \pmod{\prod_{i=1}^{n} a_i}$.

Дешифрування за допомогою методу Квісквотера-Коуврера призводить до збільшення швидкості приблизно в 4 рази.

2.6.2 Модифікація алгоритму RSA з використанням еліптичних кривих

Окремий вид асиметричних методів оперує еліптичними кривими над конкретним полем - множині точок (x, y), що відповідають рівнянням [23]:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (2.12)$$

Точки, які лежать на визначеній еліптичній кривій, визначається операція додавання, аналогічна операції множення у криптографічному алгоритмі RSA.

У 1991 році Ю.М. Моркер, Т. Окамото К. Кояма, та С.А. Ванстоун представили першу роботу в цьому напрямі у криптографії, розробивши модифікацію методу RSA, яка використовує еліптичні криві над кільцем Z_n. Цю модифікацію в подальшому назвали KMOV. Дослідники створили три класи односторонніх функцій, кожен з яких вирішує різні завдання. Перший - не є придатним для використання у криптосистемі з відкритим ключем, але ідеально підходить для створення цифрового підпису. Другий – не має такої проблеми і може слугувати альтернативою алгоритму RSA. Третій - схожий на криптосистему Рабіна. Автори також використали модуль $n = pq$, і стійкість їхніх криптоалгоритмів ґрунтується на складнощах факторизації числа n. Перевагами даних алгоритмів вказувалась підвищена стійкість до атак з невеликим ключем (наприклад, атака Вінера), проте згодом це твердження було експериментально спростовано. Аналогічно, недоліками є уповільнення роботи алгоритму порівняно з ефективністю методу RSA.

Пізніше з'явилося більше асиметричних криптоалгоритмів, які використовують еліптичні криві. У 1993 році Н. Демитко вніс покращення в криптосистему KMOV. Повідомлення тепер виступає як перша координата точки $M(x, y)$, що належить еліптичній кривій

$$y^2 \equiv (x^3 + ax + b) \pmod{n}. \quad (2.13)$$

Шифрування відбувається через *e*-кратне додавання точки *M*, і шифротекст визначається за першою координатою точки *C*. Закритий ключ *d* вибирається на основі символів Лежандра $\frac{z}{p}$ і $\frac{z}{q}$,

$$z \equiv (x_c^3 + ax_c + b) \pmod{n}. \quad (2.14)$$

Також можна пришвидшити розшифрування шляхом використання Китайської теореми про остачі.

1995 року було представлено іншу криптосистему, побудовану на сингулярній еліптичній криві, яка, при дешифруванні, дозволяла зекономити час удвічі в порівнянні з методом RSA.

Із початку 21 століття криптосистеми, які засновані на еліптичних кривих, розвинулися настільки, що вони стали частиною світових стандартів безпеки. Величезна кількість досліджень підтверджує переваги криптосистем, які використовують еліптичні криві, порівняно з іншими криптосистемами з відкритим ключем, з точки зору ефективності алгоритму і рівня захищеності, що зростає пропорційно довжині ключа.

2.6.3 Модифікація RSA з використанням *n*-розширення

1997 року було створено модифікацію методу RSA, яка використовує *n*-розширення. Модуль розраховується наступним чином

$$n^k = (pq)^k, \quad (2.15)$$

p i q – довільні прості числа, *k* – кількість блоків, на які можна розбити повідомлення *M*: $M_0, M_1, \ldots, M_{k-1}$, числа $M_i < n$, $i \in [0, k-1]$.

Шифрування відбувається подібно до алгоритму RSA, так само як і дешифрування першого блоку. Наступні блоки формуються лінійні рівняння, для яких збігається час розв'язання із часом звичайного дешифрування в RSA. Це можна пояснити тим, що при рівномірному розподілі на множині відкритих текстів *M* складність знаходження відкритого тексту за відомим шифротекстом така сама, як і складність криптосистеми RSA [24].

Т. Такагі 1998 року створив нову модифікацію. В якості модуля розраховувалося число

$$n = p^k q. \tag{2.16}$$

Шифрування відбувається так само, як і в методі RSA, але розшифрування проводиться частинами. Зазначимо, що спочатку обчислюється частина повідомлення за модулем p^k, використовуючи алгоритм Такагі для n-розширення, а інша частина розрахочується за модулем q. На завершення застосовується наслідок з Китайської теореми про остачі.

Для малих значень модуля n, такий метод є швидшим приблизно у 1,5 рази порівняно із RSA.

Уже 2000 року модифікацію Такагі було доповнено використанням модуля

$$n = p^k q^l. \tag{2.17}$$

У випадку, коли числа k та l мають близькі значення, то складність розшифрування є майже мінімальною. Для того, щоб забезпечити стійкість такої криптосистеми, необхідно, щоб НСД(k,l) = 1.

2.6.4 Модифікація RSA із використанням складеного модуля

1998 року Д. Хопкінс, С. Лангфорд, Т. Коллінз і М. Сабін отримали патент на використання складеного модуля в алгоритмі RSA. Складений модуль формується як добуток більше ніж двох простих чисел. У такий спосіб, для створення модуля певної довжини можна використовувати прості числа меншої розрядності, що призводить до значного прискорення процесу формування ключів, при цьому не втрачаючи стійкості криптоалгоритму [25].

2.6.5 Методи модифікацій RSA для протидії криптоаналізу по стороннім каналам

Для запобігання криптоаналітичним атакам через сторонні канали було викрито ряд методів, які призначені забезпечити стійкість криптографічних методів від подібних загроз. Виділяють кілька загальних стратегій протидії:

- впровадження затримок у часі, періодів очікування, зміни послідовності операцій та додавання фіктивних інструкцій для методу;
- переосмислення асемблерних інструкцій для виконання математичних операцій і перезаписів пам’яті з метою маскування змін у споживанні електроенергії;
- впровадження алгоритмічних змін, для приховування згенерованих ключів за допомогою випадкової маски для кожного циклу шифрування.

Серед різноманітних стратегій протидій, методи алгоритмічних змін вирізняються своєю ефективністю, універсальністю та потужністю. Без зайвих ускладнень, цей спосіб підвищення криптографічної стійкості вважається найпростішим та найбільш вартісним у виконанні. Можна виділити дві основні групи методів для протистояння криптоаналітичним атакам через сторонні канали:

- Програмні стратегії, які враховують рандомізацію послідовностей виконання запрограмованих інструкцій методу, бітовий розклад даних та включення фіктивних інструкцій в алгоритм.
- Апаратні підходи, які охоплюють рандомізацію виконавчого часу та споживання чи компенсацію електроенергії, і випадковий порядок виконання регістрів.

Впровадження програмних методів для боротьби з криптоаналітичними атаками через сторонні канали ускладнює криптографічні алгоритми, зокрема з точки зору витрати часу. Таким чином, враховуючи значення даних, що захищаються, розглядається можливість комбінування програмних та апаратних методів при розробці, щоб зменшити вплив їхніх недоліків на криптосистему.

Одним із широко вживаних підходів для захисту від атак SCA є введення певного рівня рандомізації в дані щодо часу виконання, спожитої потужності та електромагнітних випромінювань. Особливо ефективним для такого

застосування є метод, який використовується в криптосистемах, заснованих на еліптичних кривих, де можна відповідним чином рандомізувати проективні координати поля еліптичних кривих. Використовується стійкий до атак SCA метод скалярного множення, який може працювати з будь-якою кількістю попередньо обчислених точок. Існують три стандартні методи рандомізації:

- Маскування базової точки довільною.
- Рандомізація закритого скаляра множником, що дорівнює порядку еліптичної кривої.
- Використання проективних координат для рандомізації базової точки.

Під час розробки криптографічних систем важливо уникати дій, які використовують проміжні ключі, оскільки це може маскувати значну кількість інформації від сторонніх каналів. Критичний код має уникати використання умовних виразів та елементарних операцій, таких як І, АБО та операцій відгалуження. Наявність виконуваного коду, який залежить від вхідних даних, може легко розкрити їхні властивості, саме тому потрібно, щоб код виконувався незалежно від вхідних даних. У такому випадку дані про витрачений час та кількість ектроенергії втрачають інформативність.

Наступний метод захисту від атак SCA є відомим як "сліпота" [26]. Цей метод передбачає делегування обчислень математичних функцій певному провайдеру, який не має доступу до вхідних та вихідних даних. Це дозволяє користувачеві уникати власноручного обчислення математичних функцій. Використання гомоморфних властивостей цифрового підпису RSA є основним елементом цього методу.

Впровадження затримок виконання операцій є заходом для ускладнення часового аналізу, який має на меті забезпечення однакового часу виконання для всіх операцій. Застосування такого методу виявляється складним, бо використання таймеру може дозволити відстежити відгук системи на різні запити. Також, час виконання операцій потрібно прирівнювати до часу виконання найповільнішої операції в алгоритмі, що робить такий метод менш ефективним. Додавання випадкових затримок може призвести до збільшення

кількості необхідних шифротекстів, котрі криптоаналітик може компенсувати, за рахунок збільшення обсягу аналізованих та замірюваних даних.

Не завжди можливо реалізувати техніку впровадження балансування потужності, бо вона передбачає апаратне впровадження фіктивних регістрів, які виконують різні операції, з метою утримання споживання електроенергії на постійному рівні. Такий підхід ефективно захищає від різних видів аналізу споживання струму. Проте, його недоліком слід відзначити збільшення споживання електроенергії.

На простому рівні захист від атак, заснованих на аналізі споживання енергії, полягає у використанні логічних елементів, які мають сталий рівень споживання енергії, незалежний від оброблюваних даних та виконуваних операцій.

2.7. Висновки до розділу

У цьому розділі ми розглянули роботу різних методів симетричного та асиметричного шифрування: Шифр Цезаря, DES, TripleDES, AES, BlowFish, Twofish, Threefish, IDEA, Меркле-Геллман, RSA, Діффі-Геллман та методи з використанням еліптичних кривих. Для подальшого дослідження було вибрано алгоритм RSA.

Потім було виконано докладний аналіз математичних аспектів методу асиметричного шифрування RSA, розглянуто його криптостійкість та оцінено ефективність застосування. Було проведено аналіз потенційних вразливостей та слабких місць цього алгоритму, розглянуто методи криптоаналітичних атак, до яких він може бути вразливий.

Подальше дослідження включало вивчення існуючих методів модифікації методу RSA, які мають на меті поліпшення часових характеристик алгоритму або збільшення його стійкості до конкретних видів криптоаналітичних атак.

Найкращими для подальшого використання при розробці нових модифікацій асиметричних методів шифрування виглядають:

- метод, який використовує складений модуль;
- метод, який використовує наслідок з Китайської теореми про остачі.

Вищезгадані методи вибрано через те, що вони призводять до найбільшого збільшення швидкодії алгоритму, однак зменшуючи ускладнення.

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМІЧНО-ПРОГРАМНОГО МЕТОДУ ДЛЯ ШИФРУВАННЯ ДАНИХ ІЗ ВИКОРИСТАННЯМ АСИМЕТРИЧНИХ АЛГОРИТМІВ

3.1. Метод шифрування на основі асиметричних алгоритмів

Запропонований метод базується на традиційному методі асиметричного шифрування RSA разом з модифікаціями, що використовують складений модуль та результат китайської теореми про залишки. Його метою є збільшення швидкості роботи порівняно з існуючими алгоритмами без втрати криптографічної стійкості. На рис. 3.1 зображено схему запропонованого методу шифрування.

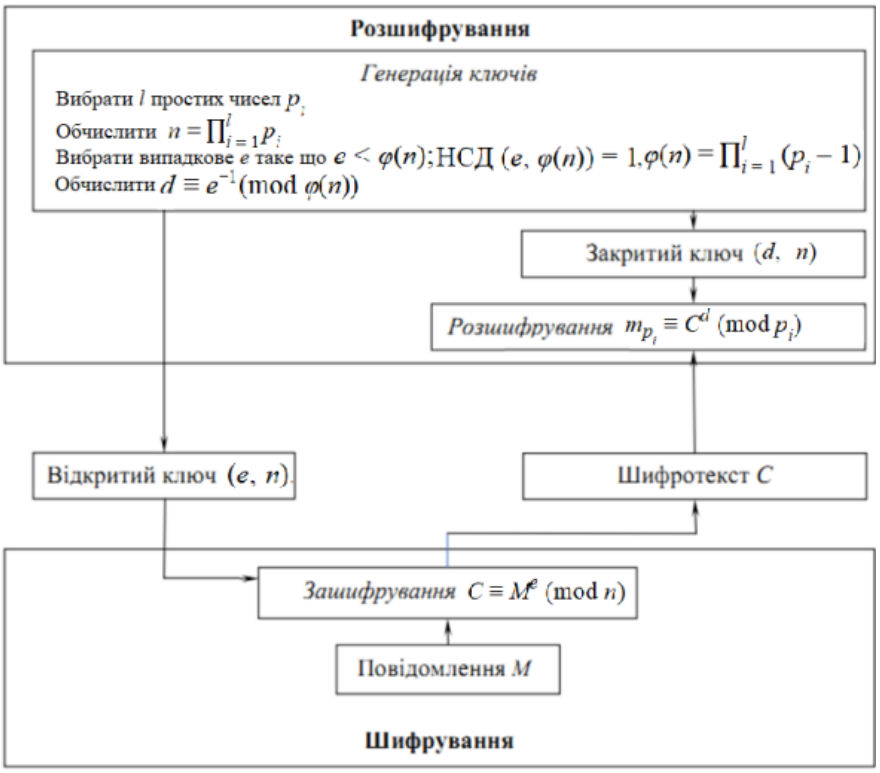


Рис. 3.1 Загальна схема запропонованого методу шифрування

3.1.1 Етап генерування ключів

Два натуральних числа (e, n) складають відкритий ключ. Число n отримується шляхом випадкової генерації l великих простих чисел p_i та обчислення їх добутку:

$$n = \prod_{i=1}^{l} p_i \tag{3.1}$$

Потім обчислюємо функцію Ейлера

$$\varphi(n) = \prod_{i=1}^{l} (p_i - 1). \tag{3.2}$$

Кількість чисел l обмежена інтервалом від $3 \leq l \leq n/4$. Це означає, кожен окремий множник має мати розрядність не менше 4 знаків, інакше тест Міллера-Рабіна для генерування простих чисел не працюватиме. Після експериментального аналізу поведінки даного методу шифрування з точки зору криптографічної стійкості та швидкості виконання буде встановлено ідеальну кількість простих чисел. Крім того, для того, щоб ускладнити атаку, рекомендується використовувати підхід рандомізації і застосовувати різну кількість простих чисел з відмінною довжиною на кілька розрядів, які будуть вибиратися випадковим чином в заздалегідь визначеному діапазоні для кожного сценарію шифрування. Так криптоаналітик не зможе заздалегідь визначити, скільки потрібно обчислити простих чисел для розкриття приватної експоненти.

Потім випадковим чином вибирається ключ шифрування, чи публічна експонента, яка є простим цілим числом e, що задовольняє наступним умовам: $e < \varphi(n)$; НСД $(e, \varphi(n)) = 1$. Пара цілих чисел (d, n), які складають ключ розшифрування, є оберненим до e, числом d, приватною експонентою:

$$d \equiv e^{-1} (\mathrm{mod}\ \varphi(n)). \tag{3.3}$$

3.1.2 Процес шифрування та розшифрування

Для отримання шифротексту C, вхідне повідомлення M шифрується шляхом використання операції піднесення до ступеня e по модулю n.

$$C \equiv M^e \pmod{n}. \tag{3.4}$$

Теорема Ейлера стверджує, що розшифрування відбувається шляхом піднесення до степеня d за модулем p_i для кожного $i \in [1; l]$:

$$m_{p_i} \equiv C^d \pmod{p_i} = (\mathrm{C} \bmod p_i)^{d \bmod (p_i - 1)} \bmod p_i. \tag{3.5}$$

Використовуючи теорему Ферма, ми можемо прискорити обчислення [27].

Далі потрібно відновити повідомлення M за допомогою m_{pi}, використовуючи китайську теорему про залишки. Для реалізації методу необхідно розв'язати систему рівнянь:

$$\begin{cases} M \equiv m_{p_1} (\mathrm{mod}\ p_1) \\ M \equiv m_{p_2} (\mathrm{mod}\ p_2) \\ \dots \\ M \equiv m_{p_l} (\mathrm{mod}\ p_l) \end{cases} \tag{3.6}$$

3.1.3 Недоліки та переваги даної методики асиметричного шифрування

Нижче наведено переваги даного алгоритму:

1. Розшифрування зашифрованого тексту вимагає не $\frac{3}{2l^3}(\log_2 n)^3$ більше бітових операцій завдяки паралельним обчисленням і наслідку китайської теореми про залишки. Тому, з кожним наступним простим числом за модулем кількість часу, необхідного для розшифрування, зменшується. Крім того, коли генерується більше простих чисел, процес стає менш складним з точки зору обчислень, що знижує складність алгоритму для генерації ключів шифрування.

2. За використанням наслідку з Китайської теореми про залишки, обсяг пам'яті, який потрібен для проведення всіх розрахунків дешифрування до завершального етапу рекомбінації, у $\frac{\log_2 n}{l}$ становить лише $\log_2 p_l$ біт, де p_l -

це найбільше просте число у модулі. Коли всі прості числа мають розмір приблизно рівний біт, обсяг необхідної пам'яті зменшується з кожним наступним простим числом.

Недоліком є той факт, що збільшення кількості простих множників у модулі може призвести до того, що його факторизація буде легшою. Для практичного застосування методу необхідно знайти оптимальну кількість множників, яка забезпечить оптимальний баланс між обчислювальною складністю та його стійкістю до атак методами факторизації.

3.2. Аналіз стійкості до атак запропонованого методу асиметричного шифрування

Нижче розглянуто вплив криптоаналітичних методів під час проведення атак на запропонований метод шифрування. Усі наведені атаки враховуються в контексті модулів із збалансованими простими числами.

3.2.1 Оцінка складності факторизації в залежності від кількості простих множників

Якщо припустити, що ймовірність розкладання на множники дорівнює 1, то ймовірність знайти один з двох простих множників дорівнює 0,5. Це дозволяє порівняти складність факторизації звичайного RSA-числа, яке є добутком двох чисел, з числом, яке є добутком трьох та більше чисел.

Ймовірність розкриття одного числа у випадку множника для $l > 2$ дорівнює $1 - \frac{1}{2^{l-1}}$. Іншими словами, у порівнянні з алгоритмом RSA, криптографічна стійкість для модуля фіксованого розміру зменшується вдвічі зі збільшенням множників, а ймовірність розкриття одного з них наближається до 1. У цьому випадку довжину модуля треба збільшити вдвчі, щоб зберегти той

самий ступінь криптостійкості при збільшенні на одне ціле число. Це твердження справедливе і для методів атаки грубої сили, таких як розширений алгоритм Евкліда [28].

3.2.2 Атака методу решета числового поля

На відміну від традиційних методів факторизації, метод решета числового поля дозволяє ефективно розкладати числа на прості множники, особливо коли має справу з великими числами, що є простими. Це робить його потужним інструментом для атак на криптосистеми, зокрема на асиметричні шифри, такі як RSA, якщо вони використовують ключі недостатньої довжини [29].

Атаки можуть бути успішними при використанні великих обчислювальних ресурсів та високої ефективності алгоритмів. У зв'язку з цим, для захисту від атак, ключі шифрування повинні мати достатню довжину, щоб ускладнити завдання факторизації чисел за допомогою методу решета числового поля.

3.2.3 Атака на факторизацію чисел, яка використовує метод еліптичних кривих

Використання методу еліптичної кривої для факторизації дозволяє розрахувати складений коефіцієнт p_i цілого числа n із значною швидкістю, коли коефіцієнт значно менший за $\sqrt{n}$. Найбільше факторизоване число застосовуючи цей метод мало розмір 224 біта.

За результатами дослідження Х. Ленстри [30], приблизний час для виконання методу еліптичної кривої для знаходження множника p_i цілого числа n дорівнювнює:

$$\mathrm{E}[n, p_i] = (\log_2 n)^2 \, e^{\sqrt{2}\,(\log p_i)^{1/2}(\log\log p_i)^{1/2}} \tag{3.7}$$

Отже, при використанні методу еліптичної кривої можна припускати, що час для розкладання стандартного модуля RSA буде вищий в порівнянні з часом, необхідним для розкладання складеного модуля в рекомендованому методі асиметричного шифрування при будь-якому $l > 2$. Це пояснюється використанням менших простих множників для отримання складеного модуля.

3.2.4 Атака із малою експонентою

Атака Вінера – це найбільш відома атака з використанням малої експоненти. Візьмемо відкритий ключ (n, e) та відповідний йому закритий ключ (n, d). При виконанні умови

$$d < \frac{n^{1/2l}}{\sqrt{2\,(2l-1)}}, \tag{3.7}$$

тоді приватна експонента буде обчислена за поліноміальний час, а саме $\log n$.

Наступна відома атака побудована на алгоритмі редукції на решітках. Цей метод використовує підходи для знаходження одновимірних модульних рівнянь малих розв’язків.

Атака, що використовує алгоритм редукції на решітках, має характер евристики, а її результати відносяться до випадку великих модулів RSA та значної розмірності решітки. Незважаючи на це, на практиці вона продемонструвала високу ефективність. Найбільш сильна атака Дарфі та Боне на алгоритмі редукції на решітках, використовує підгратки та впроваджує геометрично прогресивні матриці для визначення обмежень обсягу використовуваних підграток. Як результат виникає таке твердження.

Для будь-якого $\varepsilon > 0$ і будь-якого $l \geq 2$ існує таке число n_0, таке що для будь-якого $n > n_0$ має місце наступне: нехай n - складений модуль, відкритий ключ (n, e), а закритий ключ - (n, d), де $e = n^{\alpha}$ та $d = n^{\delta}$. Тоді у випадку виконання умов

$$\alpha \approx 1,$$

$$\delta \leq 1 - \sqrt{1 - \frac{1}{l}} - \varepsilon, \tag{3.8}$$

то d може бути відновлено за час $\log n$.

3.2.5 Атака з частково відомим ключем

Для RSA-криптосистем, у яких публічна або приватна експонента невелика, існує кілька основних видів атак із використанням частково відомого ключа. Їм можна поділити на два основних типи: атаки на найменш значущий біт (LSB) та атаки на найбільш значущий біт (MSB).

LSB атака виконується так: для будь-якого $\varepsilon > 0$ та $l \geq 2$ існує число n_0, що для будь-якого $n > n_0$ має місце наступне: n - складений модуль, відкритий ключ (n, e), а закритий ключ - (n, d), де $e = n^{\alpha}$ та $d = n^{\delta}$. При цьому надано публічний ключ, d_0 та M, де $d \equiv d_0 \bmod M$ і $M = n^{\beta - \delta}$. В такому випадку

$$\delta \leq \frac{2}{3} + \frac{1}{3l} - \frac{2}{3l}\sqrt{(l-1)(3\alpha l + 3\beta l - 2l - 1)} - \varepsilon \tag{3.9}$$

Атака MSB виконується так само. Щоб знайти d використовується рівняння із наближеною величиною $\hat{d}$, такою що виконується вираз $|d - \hat{d}| \leq n^{\delta}$.

У 2008 році М. Хінек провів експерименти [31], які емпірично підтвердили, що ефективність атаки із використанням вказаних методів знижується при збільшенні кількості простих множників. Таким чином, використання такого типу атаки на запропонований метод асиметричного шифрування можна вважати непрактичним.

3.3. Висновки до розділу

У цьому розділі ми розглянули введений метод асиметричного шифрування та провели його аналіз, використовуючи складений модуль та метод розшифрування на основі наслідку з Китайської теореми про лишки. Також

введено рандомізацію у виборі кількості простих множників для генерації модуля *n*. Розглянуті переваги цього методу включають:

- зменшення обчислювальної складності;

- економія використання пам'яті.

У цьому розділі було розглянуто запропонованований асиметричний метод шифрування і розглянули аналіз його особливостей з використанням складеного модуля і методу дешифрування, заснованого з наслідку з китайської теореми про залишки.

Потім розглянули, як найбільш поширені методи атак на традиційний метод асиметричного шифрування RSA застосовуються до запропонованого методу асиметричного шифрування і як зміни впливають на захист системи від цих методів.

Для модуля з трьома або чотирма простими множниками такі атаки, як метод Вінера, а також атаки з використанням відомих бітів приватної чи публічної експоненти можливі, а в деяких випадках навіть ефективні, використовуючи математичні властивості схеми шифрування RSA. Однак, для того, щоб використовувати ці атаки, необхідно виконати ряд вимог. Також, зі збільшенням кількості простих множників у модулі їхня ефективність знижується. Не рекомендується використовувати малі показники, бо це робить будь-яку асиметричну криптосистему вразливою.

З кожним додатковим фактором ймовірність успішної атаки грубої сили збільшується в 2 рази. Однак такі атаки неефективні при використанні ключів, довжина яких перевищує поточну довжину, яка становить 1024 біт або більше.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ АЛГОРИТМІЧНО-ПРОГРАМНОГО МЕТОДУ АСИМЕТРИЧНОГО ШИФРУВАННЯ

4.1. Аналіз мови програмування та бібліотек

У даному проекті буде запроваджено метод асиметричного шифрування та розролене програмне забезпечення для проведення експериментального дослідження обчислювальної складності.

Для обробки статистичних даних, отриманих під час дослідження, найчастіше використовується мова програмування Python.

4.1.1 Python

Python - це мова програмування, що знаходить широке застосування в розробці веб-додатків, програмного забезпечення та обробці даних, включаючи машинне навчання (ML) [32]. Вона користується популярністю серед розробників завдяки своїй ефективності, легкості вивчення та переносимості між різними платформами. Програми, написані на Python, доступні для безкоштовного завантаження і сумісні з різними операційними системами, що сприяє прискоренню процесу розробки.

Мова програмування Python володіє численними перевагами, які роблять її привабливою для розробників:

- Python володіє простим та зрозумілим синтаксисом, що дуже схожий на англійську мову. Це полегшує читання та розуміння коду для розробників.
- Розробники можуть бути більш ефективними завдяки можливості писати менше коду в порівнянні з іншими мовами програмування.

- Python має обширну стандартну бібліотеку, яка включає багато готових до використання рішень для різноманітних завдань, що дозволяє уникнути написання коду з нуля.
- Розробники можуть легко поєднувати Python із популярними мовами, такими як Java, C і C++.
- За Python стоїть активна спільнота, яка налічує мільйони розробників з усього світу. Це забезпечує підтримку та допомогу вирішення проблем.
- Python може бути перенесений на різні операційні системи, включаючи Windows, macOS, Linux і Unix.

4.1.2 Використані бібліотеки

Для створення програмного забезпечення, що реалізує запропонований метод асиметричного шифрування, і для виконання аналізу обчислювальної складності цього методу були використані бібліотеки NumPy і Matplotlib.

Бібліотека NumPy [33] визнана серед розробників і використовується для зручного створення та управління масивами, а також для обробки логічних операцій і виконання операцій лінійної алгебри. Крім того, NumPy взаємодіє з різними мовами програмування.

Matplotlib широко використовується розробниками [34] для відображення даних у високоякісній двовимірній та тривимірній графіці (2D і 3D). Ця бібліотека особливо популярна при розв'язанні наукових завдань. Matplotlib дозволяє візуалізувати дані за допомогою різноманітних діаграм, таких як стовпчикові і лінійні, і будувати кілька діаграм одночасно. Крім того, графіку можна легко переносити на різні платформи.

У цьому дослідженні Matplotlib використовується для створення та візуалізації графіків, які демонструють конкретні показники ефективності застосування запропонованого методу шифрування.

4.2. Архітектура програмного забезпечення

Розроблене програмне забезпечення, яке втілює запропонований метод асиметричного шифрування, реалізоване як додаток, написаний на мові програмування Python.

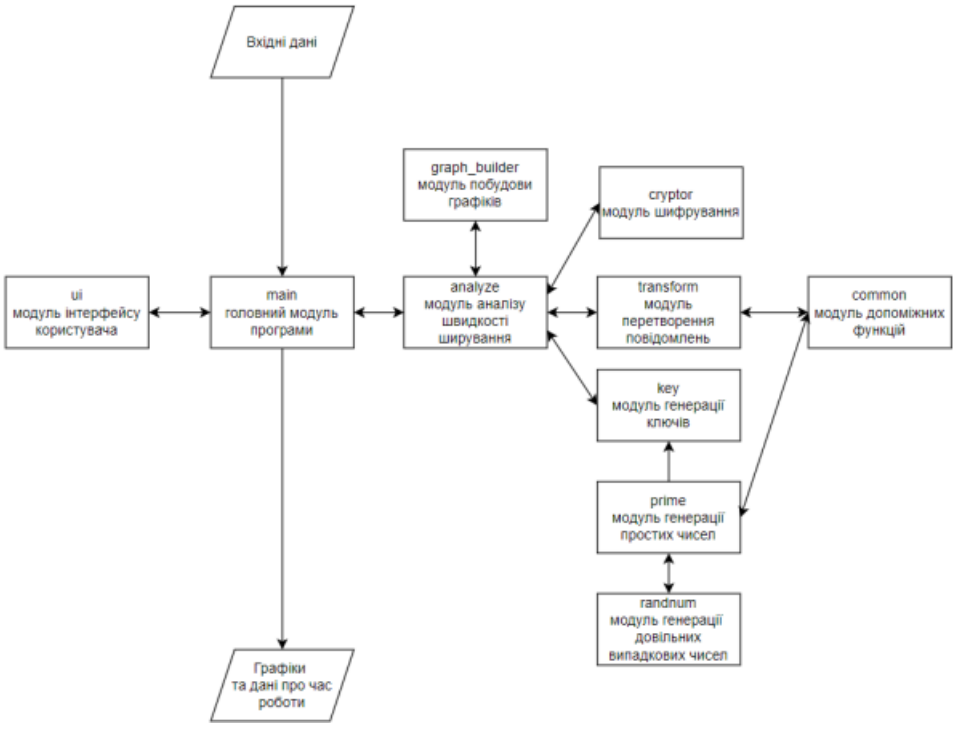


Рис. 4.1 Структурна схема програмного забезпечення

Нижче наведено розгорнутий опис кожного створеного модуля та його функціональності особливості.

- Модуль Cryptor включає в себе функціонал для проведення операцій шифрування та дешифрування. Дешифрування реалізовано у двох різних варіантах: у класичному форматі та застосуванням Китайської теореми про залишки, також за формулою.
- Модуль Transform призначений для конвертації чисел з звичайного формату у бітовий та обернено. У цьому модулі реалізовано дві функції, які виконують відповідні перетворення.
- Модуль Common включає набір математичних функцій, які використовуються у алгоритмі шифрування. До них входить функція

перевірки довжини числа у бітах та байтах, функція знаходження взаємно оберненого числа і функція перевірки знаходження найбільшого спільного дільника.

- Модуль Key містить потрібні функції для генерації ключів шифрування. Функція, яка генерує ключі, реалізована у двох варіантах - для стандартного методу RSA та для запропонованого алгоритму шифрування із використанням складеного модуля.
- Модуль Prime розроблено для створення великих простих чисел потрібної довжини.
- Модуль Randnum розроблено для створення випадкових чисел зазначеної довжини у вигляді бітів.
- Модуль Analyze включає в себе функції, які вимірюють час, потрібний для виконання методу шифрування.
- Модуль Graph_builder включає в себе функції для створення графіків на основі структури. Там зберігається інформація про точки, що використовується для побудови графіка. Крім того, модуль містить дані про назву графіка і підписи вісей.
- Модуль UI відтворює інтерфейс, який пропонує перелік доступних дій. До цих дій входить отримання графіків, що відображають залежність часу виконання реалізації запропонованого алгоритму шифрування для модуля потрібної довжини.

4.3. Аналіз швидкості алгоритму шифрування

Важливо виміряти час реалізації запропонованого методу шифрування при різній довжині модуля n, щоб знайти ідеальну кількість великих чисел l, обмежену $3 \leq l \leq n/4$, то розрядність кожної складової становить не менше 4 цифр. Для цього алгоритм виконає весь цикл створення ключа, шифрування та дешифрування для сотні випадково згенерованих повідомлень для значення l із

діапазону можливих значень. Для оцінки ефективності алгоритму буде виміряно час його роботи для різних значень *l* у діапазоні потрібних значень, а також для різних розмірів модуля *n* від 1024 біт. На основі отриманих результатів буде зображено графік залежності роботи алгоритму шифрування від значення *l* і модуля *n*.

Зі збільшенням довжини модуля n, верхня межа, як очікується, зростатиме. Однак використання надмірно великого значення змінної l, з огляду на проблеми криптографічної безпеки, є недоцільним. Виникає потреба обмеження l певним постійним значенням, що дозволяє спостерігати за поведінкою алгоритму шифрування при великих значеннях цього параметра. Отже, максимально допустимі значення l при проведенні експериментального аналізу даного алгоритму шифрування будуть обмежені точкою, в якій буде помітно уповільнення роботи. Результати дослідження представлені на рис. 4.2-4.5.

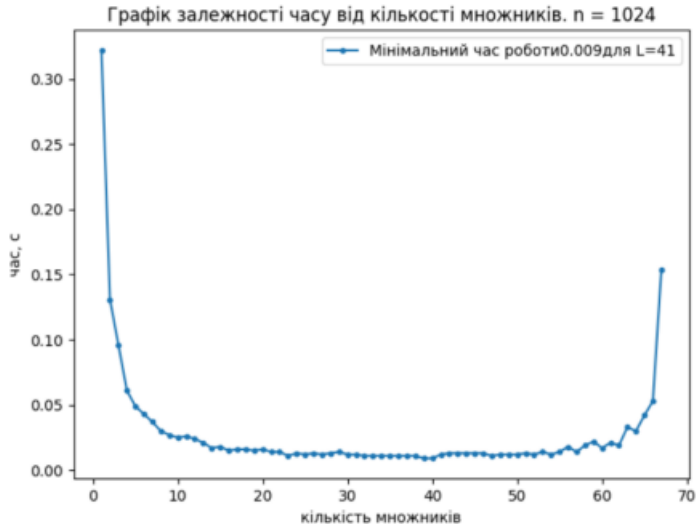


Рис. 4.2. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 1024

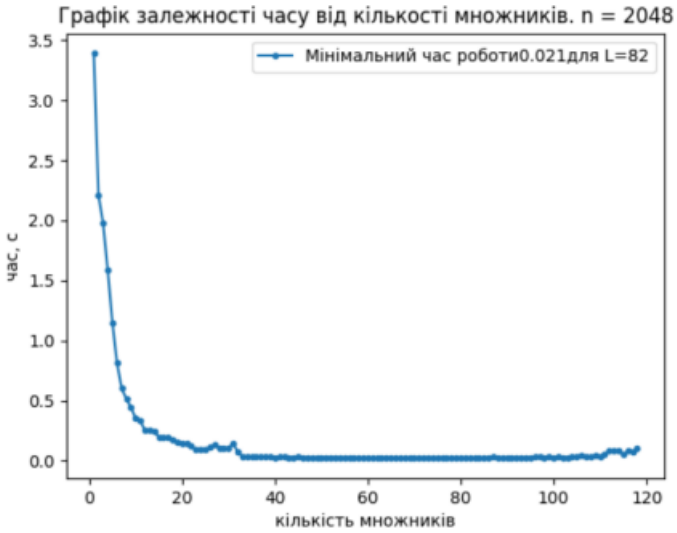


Рис. 4.3. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 2048

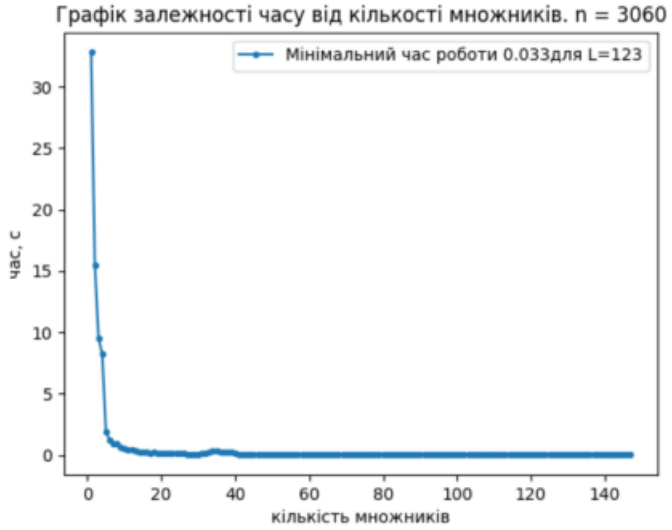


Рис. 4.4. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 3060

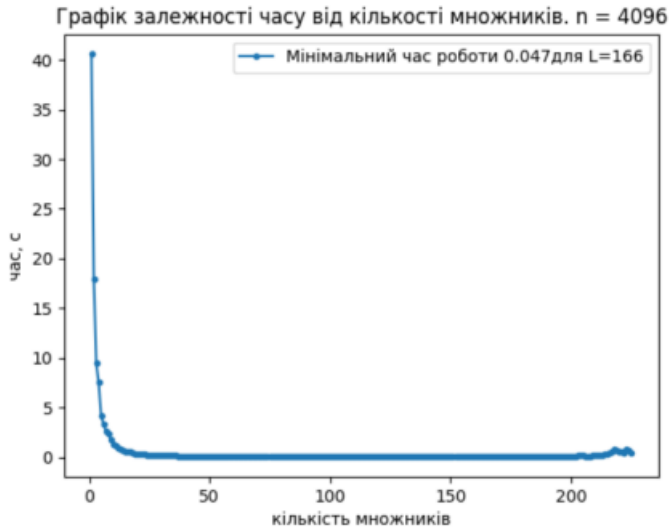


Рис. 4.5. Графік залежності часу виконання реалізації запропонованого методу шифрування для модуля n = 4096

Розглядаючи графіки, що ілюструють зміну швидкості роботи методу шифрування залежно від кількості множників у модулі, виявляється стійка закономірність мінімізації часу виконання алгоритму в інтервалі, пропорційному довжині модуля. Використання отриманих значень дозволяє сформулювати наступну залежність:

$$\frac{n}{l} = \frac{1024}{41} \approx \frac{2048}{82} \approx \frac{3060}{123} \approx \frac{4096}{166} \approx 25 \tag{4.1}$$

Потрібно зазначити, що оптимальний час виконання методу досягається при виборі кількості простих чисел l, як

$$l = \frac{n}{25}. \tag{4.2}$$

Щоб внести елемент випадковості та підвищення стійкості до атак, рекомендується обирати кількість простих чисел l, як

$$l = \frac{n}{m},\ m \in [20, 30]. \tag{4.3}$$

Щоб продемонструвати ефективність запропонованого підходу до вибору кількості простих чисел, позначених через l, ми порівняємо графік часу виконання алгоритму з використанням запропонованого методу з графіком, побудованим з використанням мінімальних значень часу виконання, отриманих раніше, доповнених проміжними значеннями. Графік порівняння зображено на рис. 4.6.

Рис. 4.6. Графіки часу виконання реалізації запропонованого методу шифрування та часу мінімального виконання

Графіки демонструють, що використання даного підходу до вибору кількості чисел *l* дозволяє досягти часу виконання алгоритму для методу шифрування, що впритул наближається до мінімального. Хоча теоретично існує можливість підвищення продуктивності за рахунок зменшення діапазону числа m, такі дії не рекомендуються, оскільки це може потенційно полегшити виконання криптоатак на алгоритм.

4.4. Оцінка практичності застосування наслідку з Китайської теореми про залишки.

Включення наслідку з Китайської теореми про залишки в будь-яку варіацію методу RSA призводить до скорочення обчислювальної складності, що у свою чергу призводить до скорочення часу виконання алгоритму. Застосування цієї теореми до звичайної реалізації методу дає скорочення часу виконання в 4 рази. Однак необхідно знайти доцільність комбінування стратегій для мінімізації часу виконання алгоритму, зокрема, збільшення кількості простих чисел у модулі та використання наслідків Китайської теореми про залишки (КТЗ). Це дослідження є дуже важливим, оскільки існує ймовірність того, що використання

теореми може надмірно ускладнити обчислення при роботі зі значною кількістю простих множників.

Щоб оцінити доречність інтеграції різних стратегій для підвищення ефективності алгоритму, ми порівняємо часовий графік алгоритму, який використовує повну форму запропонованого методу, що включає використання згаданої теореми, з часовим графіком алгоритму, який не використовує її. Графік порівняння зображено на рис. 4.7.

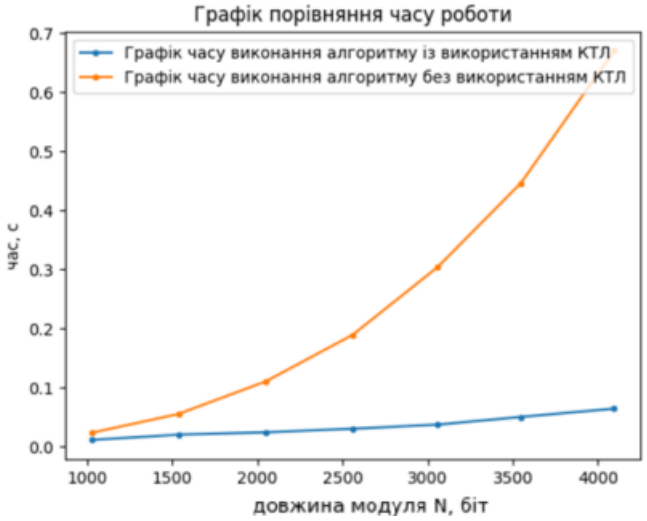


Рис. 4.7. Графіки часу виконання алгоритму із використанням КТЗ та без використання КТЗ

Графіки часу виконання демонструють ефективність інтеграції стратегій, спрямованих на покращення його продуктивності, зокрема, збільшення простих чисел у модулі та використання Китайської теореми про залишки. Аналіз цих графіків відображає, що прискорення роботи алгоритму пропорційне збільшенню розміру модуля.

4.5. Доречність використання модифікованого методу шифрування порівняно з традиційним алгоритмом RSA.

Щоб оцінити практичність застосування запропонованого методу перед традиційним методом RSA побудуємо і ретельно проаналізуємо графіки часу виконання для кожного відповідного підходу. Графік зображено на рис. 4.8.

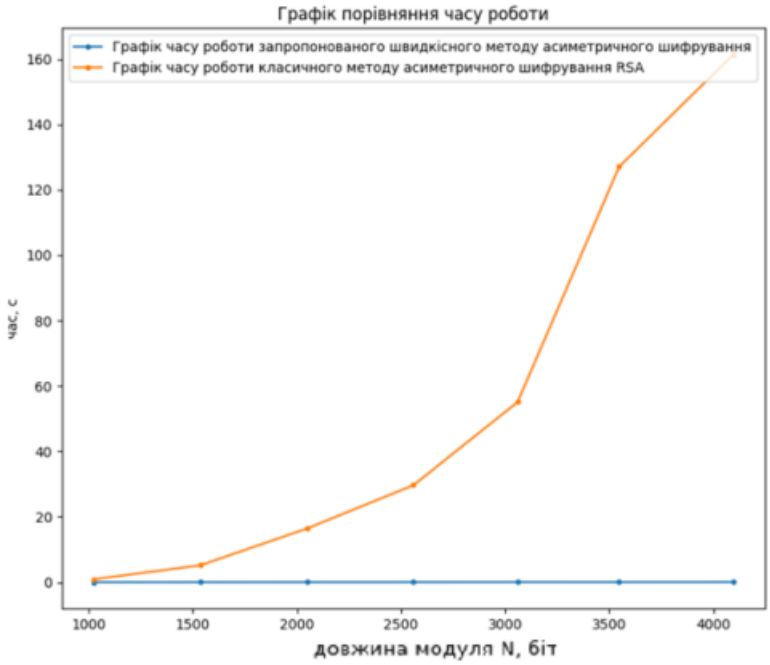


Рис. 4.8. Графіки часу виконання класичного методу RSA та запропонованого методу шифрування

Для визначення точного приросту швидкості запропонованого асиметричного методу порівняно з класичним методом RSA щодо довжини модуля був проведений аналіз графіків, наведених на рисунку 4.8. На основі отриманих значень було розраховано співвідношення часу роботи між цими методами. Після цього було побудовано графік зміни співвідношення обчислювальної складності між класичним алгоритмом RSA та запропонованим асиметричним методом шифрування.

На основі отриманих значень було побудовано функцію, що зменшує обчислювальну складність алгоритму:

$$f(x) = -1{,}275 \times 10^{-13}x^5 + 1{,}72 \times 10^{-9}x^4 - 8{,}508 \times 10^{-6}x^3 + 1{,}9609 \times \\ \times 10^{-2}x^2 - 20{,}606\,x + 8009 \qquad (4.4)$$

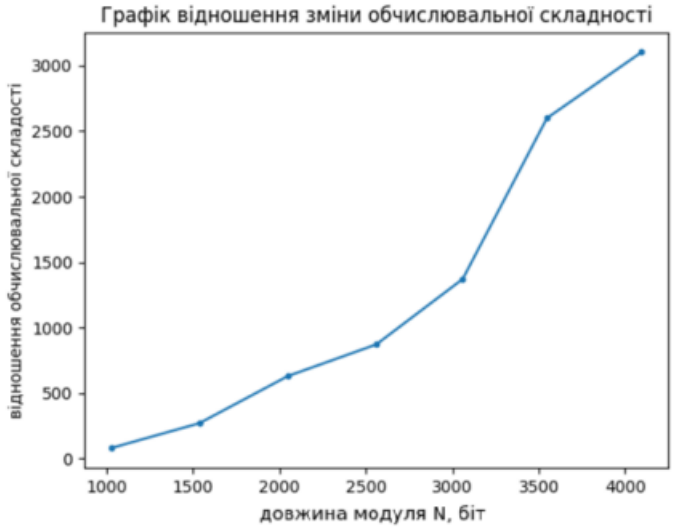


Рис. 4.9. Графік зміни значення відношення обчислювальної складності класичного алгоритму RSA до обчислювальної складності запропонованого методу асиметричного шифрування

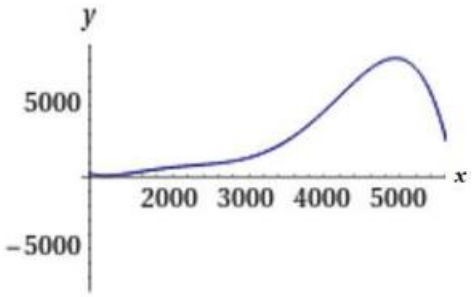


Рис. 4.10. Графік функції зменшення обчислювальної складності роботи алгоритму

Проаналізувавши функцію, можна помітити що при збільшенні довжини модуля більше ніж на 5000 біт, часова диспропорція поступово зменшується. Оптимальний результат для зменшення складності обчислювання досягається в діапазоні [4000; 5000] для запропонованого методу шифрування.

4.6. Аналіз обчислювальної складності алгоритму

З метою оцінки обчислювальної складності запропонованого методу шифрування при великих значеннях модуля створено графік, на якому

відображено залежність часу роботи методу від довжини модуля *n* на інтервалі [1024; 10240] біт. Графік зображено на рис. 4.11.

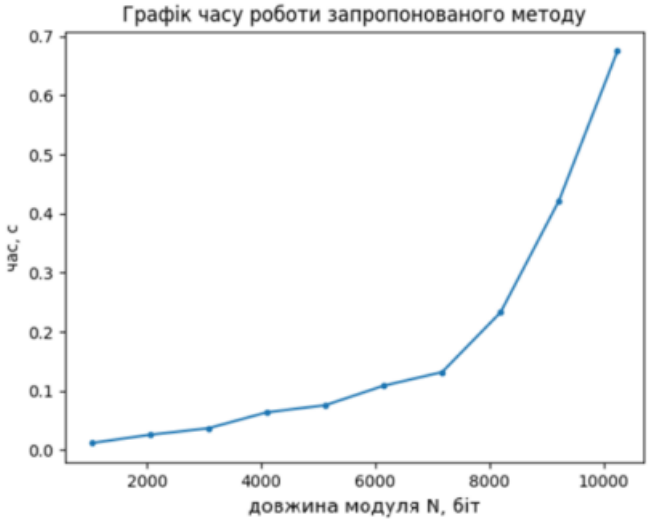


Рис. 4.11. Графік залежності часу роботи методу від довжини модуля n на інтервалі [1024; 10240] біт

Побудований графік демонструє, що час виконання алгоритму шифрування збільшується пропорційно збільшенню довжини модуля n, іншими словами, спостерігається експоненційний ріст.

Графік ілюструє, що час виконання алгоритму шифрування зростає експоненціально.

4.7. Висновки до розділу

У цьому розділі було проаналізовано мову програмування – Python та використані для програмного забезпечення бібліотеки NumPy і Matplotlib, які потрібні для того, щоб розраховувати складні операції та візуалізувати отримані дані.

Далі проаналізовано структуру програмного забезпечення, описані модулі і зв’язки між частинами додатку.

Проведено практичне дослідження для оцінки ефективності створеного алгоритмічно-програмного підходу до асиметричного шифрування. Для цього було проведено порівняння часу роботи з традиційним методом RSA.

На основі аналізу ми сформулювали рівняння для визначення оптимальної кількості великих простих чисел у складеному модулі. Використання цієї формули дозволяє досягти найбільш ефективного часу виконання алгоритму, що дає оптимальні результати у зменшенні обчислювальної складності.

РОЗДІЛ 5. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА ФУНКЦІЇ ПРИРОДНО-ЗАПОВІДНИХ ТЕРИТОРІЙ В УКРАЇНІ.

Заповідні території виконують багато різноманітних функцій, серед яких основними є:

- **Відновлення екосистем та їх компонентів.**

Діяльність людини призвела до трансформації природних екосистем і ландшафтів практично на всій території України. Заповідні території, хоч і меншою мірою, але також антропогенно змінені. Отож, крім режимних заходів збереження природних екосистем і ландшафтів, одним із важливих напрямків природоохоронної діяльності територій ПЗФ є відновлення природних екосистем і їх компонентів.

Відновлювальні заходи відрізняються від власне режимної діяльності тим, що вони обмежені у часі, мають переважно одноразовий характер.

Заходи по відновленню природних екосистем, ландшафтів та їх компонентів можуть бути дуже різноманітними. Склад цих заходів і методи їх реалізації визначаються станом природних екосистем, їх розміром, ступенем антропогенної змінності, пріоритетами їх охорони та багатьма іншими чинниками.

В Україні в найбільшій мірі потребують відновлення степові екосистеми. За розораністю території й площею орних земель на одного жителя Україна посідає одне з перших місць у світі.

Як одну із форм ренатуралізації природних комплексів можна розглядати і роботи по реінтродукції видів. Дещо іншою формою ренатуралізації є введення в екосистему ключових видів, які необхідні для підтримання її структури і функцій, зокрема трофічних зв’язків.

- **Рекреаційна функція.**

Одним із видів використання територій та об’єктів ПЗФ є їх використання в оздоровчих та інших рекреаційних цілях. Це можливе за умови дотримання природоохоронного режиму, встановленого цим Законом та іншими актами чинного законодавства.

До установ ПЗФ України, які організовують і здійснюють рекреаційну діяльність, відносяться національні природні парки, біосферні заповідники, регіональні ландшафтні парки, парки-пам’ятки садово-паркового мистецтва, ботанічні сади, зоопарки, дендропарки.

Рекреаційна діяльність на територіях установ, інших територіях та об’єктах ПЗФ полягає в забезпеченні попиту населення в оздоровленні та відпочинку, туризмі, санаторно-курортному лікуванні, полюванні тощо.

Екскурсійна діяльність у межах ПЗФ України становить собою різновид рекреаційної діяльності щодо організації подорожей, які не перевищують 16 годин, тобто без ночівлі, у супроводі фахівця-екскурсовода заздалегідь складеними маршрутами для ознайомлення з визначними місцями, пам’ятками природи, історії, культури тощо.

Важливою формою рекреації є туризм. Розрізняють багато видів туризму, але серед них для туристичних форм діяльності в межах заповідних територій є екотуризм. Стратегічна мета екотуризму − лімітована потребами збереження довкілля рекреаційна діяльність. Загальним правилом екотуризму, що відрізняє його від «традиційного» туризму є наявність досить жорстких правил поведінки, значно жорсткіших, ніж на звичайних туристських маршрутах.

В умовах розвитку товарно-грошових відносин суб’єкти рекреаційної діяльності на території ПЗФ України мають сприйняти та «вписатися» у ринкове середовище. Тому, нагальним є реструктуризація існуючої системи рекреаційного господарювання, підвищення ефективності використання рекреаційного «продукту» ПЗФ, який має певну споживну вартість.

- **Функцію збереження та відновлення етнічних традицій природокористування.**

На стадії формування етносу, за незначної густини населення, спорадичні негативні результати природокористування не могли призвести до катострофічних наслідків, але збагачували суспільство досвідом, який часом виливався в систему усталених традицій, правил і регламентацій. Внаслідок цих

адаптивних процесів традиційне (етнічне) природокористування набуло ознак, що цілком відповідають сучасним науковим принципам охорони довкілля.

На відміну від загального поняття природокористування, традиційне, або етнічне природокористування − це процес взаємодії етносу з етнічною територією і є результатом їх багатовікової коеволюції та взаємоадаптації.

Традиційне етнічне природокористування, окрім того що є вагомим чинником збереження національної едентичності та звичного стилю життя місцевого населення, також суттєво сприяє збереженню біологічного та ландшафтного різноманіття, збалансованому використанню природних ресурсів й охороні природи взагалі.

- **Функцію бази для наукових досліджень територій та об'єктів ПЗФ.**

Природні заповідники та національні природні парки є науковими установами, де проводяться регулярні наукові дослідження. Програма наукової роботи в заповідниках отримала назву "Літопис природи".

Законом України "Про природно-заповідний фонд України" (1994) передбачено, що Літопис природи є основною формою узагальнення наукових досліджень в заповідниках та національних природних парках.

Для проведення наукових досліджень в заповідниках та природних парках обладнуються наукові полігони. Саме на цих наукових полігонах згідно програми Літопису і проводиться наукова робота в згаданих категоріях ПЗФ.

Протягом всього періоду існування заповідників та національних природних парків вивчається їх флора та фауна, проводиться її інвентаризація. Особлива увага приділяється раритетній компоненті флори – рідкісним видам.

Традиційним для літопису природи є ведення календаря природи. Це – періодизація річного циклу природи, яка дає можливість відобразити характерні біокліматичні риси поточного року та сезонів.

На наукових установах в природі лежить складне і важливе завдання збирання та аналізу інформації, отриманої безпосередньо в природних умовах. Нині актуальним є питання про більш широке використання цієї інформації.

- **Освітньо-виховна діяльність.**

Сучасне суспільство стало перед проблемою гармонійних взаємовідносин з природою та оптимізацією довкілля, єдністю пізнання законів природи і їх використання для блага Homo sapiens. Вирішення цих проблем можливе за досягнення високого рівня свідомості, культури, освіти та виховання. При цьому екологічна освіта та виховання сприяють формуванню громадянина, його свідомості, патріотизму та професіоналізму.

В нашій державі створено державні природоохоронні заклади, які активно займаються проблемами екологічної освіти та виховання. До них передусім відносяться такі установи, як заповідники, національні природні парки, ландшафтні парки, природничі музеї. Такі організації є місцем впровадження формальної і неформальної екологічної освіти, а також самостійно організовують і проводять освітньо-виховну діяльність.

ВИСНОВКИ

Алгоритми шифрування відіграють ключову роль у забезпеченні безпеки в сучасному цифровому обміні даними. Існують різні способи порівняти алгоритми шифрування та продемонструвати їхні сильні та слабкі сторони. Для того, щоб вибрати відповідний алгоритм шифрування, користувачі можуть враховувати різні фактори, такі як швидкість, пропускна здатність, складність, завантаження процесора, рівень безпеки тощо. У даній роботі було декількох алгоритмів шифрування з детальним описом внутрішньої роботи кожного з них. Крім того, ми порівняли та проаналізували результати, отримані десятьма алгоритмами шифрування: AES, BlowFish, DES, IDEA з точки зору часу шифрування, пропускної здатності.

Було проведено аналіз функціонування методу RSA для асиметричного шифрування, його стійкості до криптографічних атак і ефективності в застосуванні. Були детально розглянуті вразливості цього методу, зокрема атаки на основі факторизації, непрямі методи атаки і атаки через сторонні канали. Було досліджено існуючі модифікації методу RSA, спрямовані на збільшення його стійкості до конкретних криптоаналітичних атак.

Запропоновано метод асиметричного шифрування, який ґрунтується на традиційному методі RSA. Для модифікації було введено складений модуль та використано наслідок з Китайської теореми про залишки. Проведено аналіз використання різних методів криптографічних атак, таких як атака, атака із використанням малої експоненти, атака Вінера, атака на основі еліптичної кривої, атака з використанням частково відомих біт. Також досліджено вплив введених модифікацій на стійкість до цих атак. Моделювання ефективності атаки в залежності від кількості простих множників у складеному модулі показало, що всі атаки втрачають ефективність при збільшенні кількості простих множників або залишаються нечутливими до цього параметра. Аналіз атаки грубою силою підтвердив, що її успішність зростає пропорційно кількості

простих множників у модулі, але такі атаки залишаються неефективними для ключів задовженням 1024 біта і більше.

Був проведений огляд відповідних інструментів для розроблення програмного продукту, який втілює метод асиметричного шифрування та надає можливість проводити його аналіз. Вирішено розробляти це програмне забезпечення на мові програмування Python та з використанням бібліотек NumPy і Matplotlib.

Було проведено дослідження ефективності розробленого методу шифрування, використовуючи складений модуль різної довжини. У ході цього експерименту була визначена формула для отримання оптимальної кількості великих чисел у складеному модулі. Використання цієї формули дозволяє досягти найкращого результату з точки зору зменшення обчислювальної складності.

Порівняльний аналіз часу роботи звичайного алгоритму RSA та часу роботи представленого методу шифрування підтвердив, що запропонований метод виявляє меншу обчислювальну складність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. F. Zhang, Z.-y. Liang, B.-l. Yang, X.-j. Zhao, S.-z. Guo, and K. Ren, “Survey of design and security evaluation of authenticated encryption algorithms in the caesar competition,” Frontiers of Information Technology & Electronic Engineering, vol. 19, no. 12. pp. 1475–1499, 2018.

2. Diffie W., Hellman M.E. New Directions in Cryptography IEEE Trans. Inf. Theory F. Kschischang — IEEE, 1976.

3. O. G. Abood and S. K. Guirguis, “A survey on cryptography algorithms,” International Journal of Scientific and Research Publications, vol. 8, no. 7, pp. 410–415, 2018.

4. R. Bhanot and R. Hans, “A review and comparative analysis of various encryption algorithms,” International Journal of Security and Its Applications, vol. 9, no. 4, pp. 289–306, 2015.

5. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM — New York City: ACM, 1978.

6. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходный код на С / Б. Шнайер. - М.: Вильямс, 2016 – 842 с.

7. G. Kaur and M. Mahajan, “Evaluation and comparison of symmetric key algorithms,” International Journal of Science, Engineering and Technology Research (IJSETR), vol. 2, no. 10, pp. 1960–1962, 2013.

8. N. Tyagi and A. Ganpati, “Comparative analysis of symmetric key encryption algorithms,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 8, pp. 63–70, 2014.

9. A. Y. Hendi, M. O. Dwairi, Z. A. Al-Qadi, and M. S. Soliman, “A novel simple and highly secure method for data encryption-decryption,” International Journal

of Communication Networks and Information Security, vol. 11, no. 1, pp. 232–238, 2019.

10. M. Mathur and A. Kesarwani, “Comparison between des, 3des, rc2, rc6, BlowFish and aes,” in Proceedings of National Conference on New Horizons in IT-NCNHIT, vol. 3, 2013, pp. 143–148.

11. Bakhtiari M., Maarof M.A. Serious security weakness in RSA cryptosystem, International Journal of Computer Science Issues, 2012 –сс. 175-178.

12. RSA-числа [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/RSA

13. Pomerance C., Lenstra H., Analysis and comparision of some integer factoring algorithms, Computational methods in number theory, №1, 1982 – сс. 89-139.

14. P. J. Sun, “Privacy protection and data security in cloud computing: a survey, challenges, and solutions,” IEEE Access, vol. 7, pp. 147 420–147 452, 2019.

15. Wiener M.J., Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, 1990.

16. Vipul Ved Prakash, Benjamin Trott, “Asymmetric Cryptography in Perl”, O'Reilly, 2001.

17. Joye M., Olivier F. Side-channel analysis, Encyclopedia of Cryptography and Security, 2005, сс. 571–576.

18. Gutmann P., Naccache D., Palmer C.C. Side Channel attacks on cryptographic software, copublished by the IEEE computer and reliability societies, 2009.

19. J. Katz and Y. Lindell, Introduction to Modern Cryptography, ser. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press, 2014.

20. Großschadl J. The Chinese Remainder Theorem and its Application in a High-Speed RSA Crypto Chip.

21. W. Stallings, “The principles and practice of cryptography and network security 7th edition, isbn-10: 0134444280,” Pearson Education, vol. 20, no. 1, p. 7, 2017.

22. S. Basu, “International data encryption algorithm (idea)– a typical illustration,” Journal of global research in Computer Science, vol. 2, no. 7, pp. 116–118, 2011.

23. Koyama K. New public-key schemes based on elliptic curves over the ring. Annual International Cryptology Conference. Berlin, Heidelberg, 1991 – cc. 252-266.

24. Demytko N. A new elliptic curve based analogue of RSA. Workshop on the Theory and Application of of Cryptographic Techniques. Berlin, Heidelberg, 1993

25. W.Diffie and M.E.Hellman, “New directions in cryptograpgy”, IEEE Trans.Inf.Theory, vol.IT-22, N6, p.644-654, Nov. 1976.

26. Takagi T. Fast RSA-type cryptosystems using n-adic expansion. Annual International Cryptology Conference, Berlin, Heidelberg, 1997 – cc. 372-384.

27. Collins T. et al. Public key cryptographic apparatus and method, пат. 5848159 США, 1998.

28. Schindler, Werner. A Timing Attack against RSA with the Chinese Remainder Theorem, Cryptographic Hardware and Embedded Systems CHES 2000, volume 1965 of Lecture Notes in Computer Science, Berlin, Heidelberg, 2000 – cc.109–124.

29. Stinson D.R. Cryptography : Theory and Practice. CRC Press LLC, 1995.

30. Lenstra H.W. Elliptic curves and number-theoretic algorithms, International Congress of Mathematicians, 1986 – cc. 99-120.

31. Lehman R.S., Factoring large integers, Math. Comp., 1976 – cc. 637-646.

32. Boneh D., Durfee G. Cryptanalysis of RSA with private key d less than x 0,292, IEEE transactions on Information Theory, 2000 –cc. 1339-1349.

33. Ernst M., Jochemsz E., May A., de Weger B. Partial Key Exposure Attacks on RSA up to Full Size Exponents. Advances in Cryptology – Proceedings of EURO-CRYPT, 2005, cc. 371–387.

34. Hinek M.J. On the security of multi-prime RSA //Journal of Mathematical Cryptology, 2008 – cc. 117-147.

Додаток А

Код програмного продукту

```
class NotRelativePrimeError(ValueError):
 def __init__(self, a: int, b: int, d: int, msg: str = '') -> None:
 super().__init__(msg or "%d and %d are not relatively prime,
divider=%i" % (a, b, d))
 self.a = a
 self.b = b
 self.d = d
def bit_size(num: int) -> int:
 """
 Number of bits needed to represent a integer excluding any prefix
 0 bits.
 Usage::
 >>> bit_size(1023)
 10
 >>> bit_size(1024)
 11
 >>> bit_size(1025)
 11
 :param num:
 Integer value. If num is 0, returns 0. Only the absolute value of
the
 number is considered. Therefore, signed integers will be abs(num)
 before the number's bit length is determined.
 :returns:
 Returns the number of bits in the integer.
 """
```

```
 try:
 return num.bit_length()
 except AttributeError as ex:
 raise TypeError('bit_size(num) only supports integers, not %r' %
type(num)) from ex
def byte_size(number: int) -> int:
 """
 Returns the number of bytes required to hold a specific long number.
 The number of bytes is rounded up.
 Usage::
 >>> byte_size(1 << 1023)
 128
 >>> byte_size((1 << 1024) - 1)
 128
 >>> byte_size(1 << 1024)
 129
 :param number:
 An unsigned integer
 :returns:
 The number of bytes required to hold a specific long number.
 """
 if number == 0:
 return 1
 return ceil_div(bit_size(number), 8)
def ceil_div(num: int, div: int) -> int:
 """
 Returns the ceiling function of a division between `num` and `div`.
 Usage::
 >>> ceil_div(100, 7)
 15
```

```
>>> ceil_div(100, 10)
10
>>> ceil_div(1, 4)
1
:param num: Division's numerator, a number
:param div: Division's divisor, a number
:return: Rounded up result of the division between the parameters.
"""
quanta, mod = divmod(num, div)
if mod:
quanta += 1
return quanta
def extended_gcd(a: int, b: int):
"""Returns a tuple (r, i, j) such that r = gcd(a, b) = ia + jb
"""
# r = gcd(a,b) i = multiplicitive inverse of a mod b
# or j = multiplicitive inverse of b mod a
# Neg return values for i or j are made positive mod b or a
respectively
# Iterateive Version is faster and uses much less stack space
x = 0
y = 1
lx = 1
ly = 0
oa = a # Remember original a/b to remove
ob = b # negative values from return results
while b != 0:
q = a // b
(a, b) = (b, a % b)
(x, lx) = ((lx - (q * x)), x)
```

```
 (y, ly) = ((ly - (q * y)), y)
 if lx < 0:
 lx += ob # If neg wrap modulo orignal b
 if ly < 0:
 ly += oa # If neg wrap modulo orignal a
 return a, lx, ly # Return only positive values
def inverse(x: int, n: int) -> int:
 """Returns the inverse of x % n under multiplication, a.k.a x^-1 (mod
n)
 >>> inverse(7, 4)
 3
 >>> (inverse(143, 4) * 143) % 4
 1
 """
 (divider, inv, _) = extended_gcd(x, n)
 if divider != 1:
 raise NotRelativePrimeError(x, n, divider)
 return inv
from functools import reduce
def encrypt(message: int, pubkey: int) -> int:
 (n, ekey) = pubkey
 if message < 0:
 raise ValueError('Only non-negative numbers are supported')
 if message > n:
 raise OverflowError("The message %i is too long for n=%i" %
(message, n))
 return pow(message, ekey, n)
def decrypt(cyphertext: int, privkey: int) -> int:
 (n, dkey) = privkey
 return pow(cyphertext, dkey, n)
```

```
def chinese_remainder_decrypt(cyphertext: int, privkey: int) -> int:
 (primes, dkey) = privkey
 messages = []
 for prime in primes:
 messages.append(pow(pow(cyphertext, 1, prime), pow(dkey, 1, prime -
1), prime))
 return chinese_remainder(primes, messages)
def chinese_remainder(modulo, cyphers):
 sum = 0
 prod = reduce(lambda acc, b: acc * b, modulo)
 for n_i, a_i in zip(modulo, cyphers):
 p = prod // n_i
 sum += a_i * mul_inv(p, n_i) * p
 return sum % prod
def mul_inv(a, b):
 b0 = b
 x0, x1 = 0, 1
 if b == 1:
 return 1
 while a > 1:
 q = a // b
 a, b = b, a % b
 x0, x1 = x1 - q * x0, x0
 if x1 < 0:
 x1 += b0
 return x1
import random
import analyse
MIN_SIZE = 2
MAX_SIZE = 1000
```

```
DEFAULT_BITS = 1024
MESSAGE_COUNT = 100
class Demo:
 def __init__(self, scenario_number, nbits=DEFAULT_BITS,
message_count=MESSAGE_COUNT, min_size=MIN_SIZE,
 max_size=MAX_SIZE):
 self.messages = self.message_generator(message_count, min_size,
max_size)
 if scenario_number == 1:
 self.scenario1(nbits)
 elif scenario_number == 2:
 self.scenario2()
 elif scenario_number == 3:
 self.scenario3()
 elif scenario_number == 4:
 self.scenario4()
 elif scenario_number == 5:
 self.scenario5()
 @staticmethod
 def message_generator(n, min_size, max_size):
 messages = []
 for i in range(1, n + 1):
 messages.append(random.randrange(min_size, max_size))
 return messages
 def scenario1(self, nbits):
 times = [analyse.all_prime_count_test(self.messages, nbits)]
 title = "Графік залежності часу від кількості множників. n = " +
str(nbits)
 analyse.draw_graphs(times, title)
 def scenario2(self):
```

```
dict_list = []
dict_list.append(MIN_TIME)
speed_rsa_times = {}
dict_list.append(speed_rsa_times)
analyse.draw_graphs_by_dict(speed_rsa_times, MIN_TIME)
def scenario3(self):
speed_rsa_times = {}
no_crt_times = {}
for modulo in MODULO:
speed_rsa_times.update({modulo:
analyse.average_time_speed_rsa(self.messages, modulo)})
no_crt_times.update({modulo:
analyse.average_time_no_crt_rsa(self.messages, modulo)})
analyse.draw_graphs_by_dict2(speed_rsa_times, no_crt_times)
def scenario4(self):
speed_rsa_times = {}
classic_rsa_times = {}
for modulo in MODULO:
print(MODULO)
speed_rsa_times.update({modulo:
analyse.average_time_speed_rsa(self.messages, modulo)})
classic_rsa_times.update({modulo:
analyse.average_time_classic_rsa(self.messages, modulo)})
print(speed_rsa_times)
print(classic_rsa_times)
analyse.draw_graphs_by_dict3(speed_rsa_times, classic_rsa_times)
def scenario5(self):
speed_rsa_times = {}
for i in range(1, 2, 1):
k = i * 1024
```

```
 print(k)
 speed_rsa_times.update({k:
analyse.average_time_speed_rsa(self.messages, k)})
 print(speed_rsa_times)
 analyse.draw_graphs_by_dict(speed_rsa_times)
import math
import random
from rsa import common, prime
DEFAULT_EXPONENT = 65537
MIN_LIMIT = 20
MAX_LIMIT = 30
def keys_crt(nbits: int,
 prime_num: int = 2,
 accurate: bool = False,
 exponent: int = DEFAULT_EXPONENT,
 prime_calculation: bool = True):
 if nbits < 16:
 raise ValueError('Key too small')
 if prime_calculation:
 prime_num = prime_number_generator(nbits)
 if prime_num < 2:
 raise ValueError('prime number too small')
 if prime_num > nbits / 4:
 raise ValueError('prime number too large')
 # Generate the key components
 (n, e, d, primes) = gen_keys(nbits, prime_num, accurate=accurate,
exponent=exponent)
 return (
 (n, e),
 (primes, d)
```

```
 )
def keys(nbits: int,
 prime_num: int,
 accurate: bool = False,
 exponent: int = DEFAULT_EXPONENT,
 prime_calculation: bool = True):
 if nbits < 16:
 raise ValueError('Key too small')
 if prime_calculation:
 prime_num = prime_number_generator(nbits)
 if prime_num < 2:
 raise ValueError('prime number too small')
 if prime_num > nbits / 4:
 raise ValueError('prime number too large')
 # Generate the key components
 (n, e, d, primes) = gen_keys(nbits, prime_num, accurate=accurate,
exponent=exponent)
 return (
 (n, e),
 (n, d)
 )
def gen_keys(nbits: int,
 prime_num: int,
 accurate: bool = False,
 exponent: int = DEFAULT_EXPONENT):
 # Regenerate p and q values, until calculate_keys doesn't raise a
ValueError.
 while True:
 primes = find_primes(nbits, prime_num, accurate)
 try:
```

```
 (e, d) = calculate_keys_custom_exponent(primes,
exponent=exponent)
 break
 except ValueError:
 pass
 n = math.prod(primes)
 return n, e, d, primes
def find_primes(total_bits: int, counter: int = 2, accurate: bool = False):
 nbits = total_bits // counter
 primes = []
 # Keep choosing other primes until they match our requirements.
 while not is_acceptable(primes, total_bits, accurate):
 primes.clear()
 for i in range(counter):
 primes.append(prime.get_prime(nbits))
 primes.sort(reverse=True)
 return primes
def is_acceptable(primes, total_bits: int, accurate: bool) -> bool:
 if len(primes) != len(set(primes)) or len(primes) == 0:
 return False
 if not accurate:
 return True
 # Make sure we have just the right amount of bits
 found_size = common.bit_size(math.prod(primes))
 return total_bits == found_size
def prime_number_generator(nbits):
 return nbits // random.randint(MIN_LIMIT, MAX_LIMIT)
def calculate_keys_custom_exponent(primes, exponent: int):
 phi_n = 1
 for _prime in primes:
```

```
 phi_n *= (_prime - 1)
 try:
 d = common.inverse(exponent, phi_n)
 except common.NotRelativePrimeError as ex:
 raise common.NotRelativePrimeError(
 exponent, phi_n, ex.d,
 msg="e (%d) and phi_n (%d) are not relatively prime
(divider=%i)" %
 (exponent, phi_n, ex.d))
 if (exponent * d) % phi_n != 1:
 raise ValueError("e (%d) and d (%d) are not mult. inv. modulo "
 "phi_n (%d)" % (exponent, d, phi_n))
 return exponent, d
from rsa import common, randnum
def get_prime(nbits):
 assert nbits > 3 # the loop wil hang on too small numbers
 while True:
 integer = randnum.read_random_odd_int(nbits)
 # Test for primeness
 if is_prime(integer):
 return integer
def is_prime(number):
 # Check for small numbers.
 if number < 10:
 return number in {2, 3, 5, 7}
 # Check for even numbers.
 if not (number & 1):
 return False
 # Calculate minimum number of rounds.
 k = get_primality_testing_rounds(number)
```

```
 # Run primality testing with (minimum + 1) rounds.
 return miller_rabin_primality_testing(number, k + 1)
def get_primality_testing_rounds(number):
 # Calculate number bitsize.
 bitsize = common.bit_size(number)
 # Set number of rounds.
 if bitsize >= 1536:
 return 3
 if bitsize >= 1024:
 return 4
 if bitsize >= 512:
 return 7
 # For smaller bitsizes, set arbitrary number of rounds.
 return 10
def miller_rabin_primality_testing(n, k):
 # prevent potential infinite loop when d = 0
 if n < 2:
 return False
 # Decompose (n - 1) to write it as (2 ** r) * d
 # While d is even, divide it by 2 and increase the exponent.
 d = n - 1
 r = 0
 while not (d & 1):
 r += 1
 d >>= 1
 # Test k witnesses.
 for _ in range(k):
 # Generate random integer a, where 2 <= a <= (n - 2)
 a = randnum.randint(n - 3) + 1
 x = pow(a, d, n)
```

```
 if x == 1 or x == n - 1:
 continue
 for _ in range(r - 1):
 x = pow(x, 2, n)
 if x == 1:
 # n is composite.
 return False
 if x == n - 1:
 # Exit inner loop and continue with next witness.
 break
 else:
 # If loop doesn't break, n is composite.
 return False
 return True
import os
import struct
from rsa import common, transform
def read_random_bits(nbits: int) -> bytes:
 """Reads 'nbits' random bits.
 If nbits isn't a whole number of bytes, an extra byte will be appended
with
 only the lower bits set.
 """
 nbytes, rbits = divmod(nbits, 8)
 # Get the random bytes
 randomdata = os.urandom(nbytes)
 # Add the remaining random bits
 if rbits > 0:
 randomvalue = ord(os.urandom(1))
 randomvalue >>= (8 - rbits)
```

```
 randomdata = struct.pack("B", randomvalue) + randomdata
 return randomdata
def read_random_int(nbits: int) -> int:
 """Reads a random integer of approximately nbits bits.
 """
 randomdata = read_random_bits(nbits)
 value = transform.bytes2int(randomdata)
 # Ensure that the number is large enough to just fill out the required
 # number of bits.
 value |= 1 << (nbits - 1)
 return value
def read_random_odd_int(nbits: int) -> int:
 """Reads a random odd integer of approximately nbits bits.
 >>> read_random_odd_int(512) & 1
 1
 """
 value = read_random_int(nbits)
 # Make sure it's odd
 return value | 1
def randint(maxvalue: int) -> int:
 """Returns a random integer x with 1 <= x <= maxvalue
 May take a very long time in specific situations. If maxvalue needs N
bits
 to store, the closer maxvalue is to (2 ** N) - 1, the faster this
function
 is.
 """
 bit_size = common.bit_size(maxvalue)
 tries = 0
 while True:
```

```
 value = read_random_int(bit_size)
 if value <= maxvalue:
 break
 if tries % 10 == 0 and tries:
 # After a lot of tries to get the right number of bits but
still
 # smaller than maxvalue, decrease the number of bits by 1.
That'll
 # dramatically increase the chances to get a large enough
number.
 bit_size -= 1
 tries += 1
 return value
import math
def bytes2int(raw_bytes: bytes) -> int:
 r"""Converts a list of bytes or an 8-bit string to an integer.
 When using unicode strings, encode it to some encoding like UTF8 first.
 >>> (((128 * 256) + 64) * 256) + 15
 8405007
 >>> bytes2int(b'\x80@\x0f')
 8405007
 """
 return int.from_bytes(raw_bytes, 'big', signed=False)
def int2bytes(number: int, fill_size: int = 0) -> bytes:
 """
 Convert an unsigned integer to bytes (big-endian)::
 Does not preserve leading zeros if you don't specify a fill size.
 :param number:
 Integer value
 :param fill_size:
```

```
 If the optional fill size is given the length of the resulting
 byte string is expected to be the fill size and will be padded
 with prefix zero bytes to satisfy that length.
 :returns:
 Raw bytes (base-256 representation).
 :raises:
 ``OverflowError`` when fill_size is given and the number takes up
more
 bytes than fit into the block. This requires the ``overflow``
 argument to this function to be set to ``False`` otherwise, no
 error will be raised.
 """
 if number < 0:
 raise ValueError("Number must be an unsigned integer: %d" % number)
 bytes_required = max(1, math.ceil(number.bit_length() / 8))
 if fill_size > 0:
 return number.to_bytes(fill_size, 'big')
 return number.to_bytes(bytes_required, 'big')
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
from rsa import key_classic as kc, key_mod as km, cryptor
def run_classic_rsa(message, nbits, prime_count):
 (pubkey, privkey) = kc.newkeys(nbits)
 crypto = cryptor.encrypt(message, pubkey)
 return cryptor.decrypt(crypto, privkey)
def run_multiprime_rsa(message, nbits, prime_count):
 (pubkey, privkey) = km.keys(nbits, prime_count)
 crypto = cryptor.encrypt(message, pubkey)
 return cryptor.decrypt(crypto, privkey)
```

```
def run_multiprime_crt_rsa(message, nbits, prime_count):
 (pubkey, privkey) = km.keys_crt(nbits, prime_count)
 crypto = cryptor.encrypt(message, pubkey)
 return cryptor.chinese_remainder_decrypt(crypto, privkey)
def run_speed_rsa(message, nbits, prime_count):
 (pubkey, privkey) = km.keys_crt(nbits)
 crypto = cryptor.encrypt(message, pubkey)
 return cryptor.chinese_remainder_decrypt(crypto, privkey)
def time_buffer(messages, nbits, rsa, prime_count: int = 2):
 times = []
 for m in messages:
 start_time = datetime.now()
 rsa(m, nbits, prime_count)
 times.append((datetime.now() - start_time).total_seconds())
 return times
def average_time(times):
 return round(np.mean(times).item(), 3)
def average_time_speed_rsa(messages, modulo):
 return average_time(time_speed_rsa(messages, modulo))
def average_time_no_crt_rsa(messages, modulo):
 return average_time(time_multiprime_rsa(messages, modulo, 2))
def average_time_classic_rsa(messages, modulo):
 return average_time(time_classic_rsa(messages, modulo))
def time_classic_rsa(messages, nbits):
 return time_buffer(messages, nbits, run_classic_rsa)
def time_multiprime_rsa(messages, nbits, prime_count):
 return time_buffer(messages, nbits, run_multiprime_rsa, prime_count)
def time_multiprime_crt_rsa(messages, nbits, prime_count):
 return time_buffer(messages, nbits, run_multiprime_crt_rsa,
prime_count)
```

```
def time_speed_rsa(messages, nbits):
 return time_buffer(messages, nbits, run_speed_rsa)
def all_prime_count_test(messages, nbits):
 times = []
 for i in range(100, 193):
 print(i)
 times.append(average_time(time_multiprime_crt_rsa(messages, nbits,
i)))
 return times
def draw_graphs(times, title):
 x = np.arange(1, len(times[0]) + 1, 1)
 for time in times:
 plt.plot(x, time, '-o', markersize=3, label="Мінімальний час роботи
" + str(min(time)) +
 "для L=" +
str(time.index(min(time)) + 3))
 plt.title(title)
 plt.ylabel('час, с')
 plt.xlabel('кількість множників')
 plt.legend(loc="upper right")
 plt.show()
def draw_graphs_by_dict(dictionary1):
 x = dictionary1.keys()
 plt.plot(x, dictionary1.values(), '-o', markersize=3, label="Графік
мінімального часу виконання алгоритму")
 plt.plot(x, dictionary1.values(), '-o', markersize=3, label="Графік
часу при L = N / M, 20 < M < 30")
 plt.title("Графік часу роботи запропонованого методу")
 plt.ylabel('час, с')
 plt.xlabel('довжина модуля N, біт')
```

```
plt.legend(loc="upper right")
plt.show()
```