МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Ігор ЖУКОВ
« ___ » _____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"
ЗА СПЕЦІАЛЬНІСТЮ 123 "КОМП'ЮТЕРНА ІНЖЕНЕРІЯ"

Тема: Методи захисту інформації в системі дистанційного спілкування

Виконавець Тарас ІВАНОВ

Керівник: Станіслава КУДРЕНКО

Нормоконтролер: Василь МАЛЯРЧУК

Засвідчую, що у магістерській роботі немає
запозичень праць інших авторів
без відповідних посилань
Студент _____ Іванов Т.В.

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій

Кафедра комп’ютерних систем та мереж

Спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри КСМ
_____ Ігор ЖУКОВ
“_____”_____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Іванова Тараса Володимировича
(прізвище, ім'я, по батькові)

1. Тема роботи: “Методи захисту інформації в системі дистанційного спілкування”, затверджена наказом ректора від «29» серпня 2023 р. № 1521/ст.
2. Термін виконання роботи: з 02.10.2023 до 31.12.2023
3. Вихідні дані до роботи: об’єктом розробки є система дистанційного спілкування в реальному часі; система має використовувати різні методи захисту приватної інформації; розробка використовує на клієнтській стороні операційну систему “Андроїд”; вибір технологій, методів та алгоритмів захисту інформації, середовища розробки та мова написання на розсуд розробника;
4. Зміст пояснювальної записки: титульний аркуш, завдання, реферат, зміст, перелік умовних скорочень, вступ, перший розділ, висновки до першого розділу, другий розділ, висновки до другого розділу, третій розділ, висновки до третього розділу, висновки в цілому до роботи, список джерел, додаток.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: Матеріали кваліфікаційної роботи мають бути представлені у вигляді презентацій *Power Point* у форматі *.ppt* та універсальному форматі *pdf*.

6. Календарний план-графік

№ пор.	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Ознайомитись з постановкою задачі кваліфікаційної роботи	02.10.2023 - 03.10.2023	
2	Вивчити спеціальну літературу і технічну документацію	03.10.2023 - 05.10.2023	
3	Проаналізувати використані інструменти та їхні можливі замінники	06.10.2023 - 08.10.2023	
4	Написати розділ 1	09.10.2023 - 20.10.2023	
5	Проаналізувати та порівняти алгоритми та методи захисту інформації	25.10.2023 - 30.10.2023	
6	Написати розділ 2	31.10.2023 - 10.11.2023	
7	Розробити захист системи та підготувати базовий опис	11.11.2023 - 20.11.2023	
8	Написати розділ 3	21.11.2023 - 30.11.2023	
9	Оформити пояснювальну записку та пройти нормоконтроль	10.12.2023 - 22.12.2023	
10	Підготувати презентаційний матеріал та захистити роботу	10.12.2023 - 22.12.2023	

7. Дата видачі завдання «02» жовтня 2023 р.

Керівник кваліфікаційної роботи _____ Кудренко С.О.

(підпис керівника)

Завдання прийняв до виконання _____ Іванов Т.В

(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи “Методи захисту інформації в системі дистанційного спілкування ”: 92 с., 33 рис., 17 літературних джерел.

ДИСТАНЦІЙНЕ СПІЛКУВАННЯ, ЗАХИСТ ІНФОРМАЦІЇ, ОСНОВНІ ЗАГРОЗИ БЕЗПЕЦІ, МЕТОДИ ТА АЛГОРИТМИ ЗАХИСТУ

Об’єкт дослідження: алгоритми та методи захисту інформації.

Предмет дослідження: система дистанційного спілкування на базі “Андроїд”.

Мета кваліфікаційної роботи: створити захищену системи дистанційного спілкування для обміну даними.

Прогнози, припущення щодо розвитку об’єкта дослідження: підтримка, розвиток та розширення можливостей захисту приватної інформації у системі дистанційного спілкування .

Результати кваліфікаційної роботи рекомендується використовувати при розробці нових систем дистанційного спілкування або систем виключно для обміну даних, задля вибору правильного методу та/або алгоритму захисту інформації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ЗНАКІВ, СКОРОЧЕНЬ, ТЕРМІНІВ 7
ВСТУП..... 9
РОЗДІЛ 1. ОСНОВНІ ВИЗНАЧЕННЯ, ЗАГРОЗИ ТА ВАЖЛИВІСТЬ ЗАХИСТУ ІНФОРМАЦІЇ У СИСТЕМАХ ДИСТАНЦІЙНОГО СПІЛКУВАННЯ 15
1.1. Визначення системи дистанційного спілкування 15
1.2. Роль технологій у розвитку дистанційного спілкування 16
1.3. Популярні форми дистанційного спілкування..... 17
1.3.1. Відеоконференції 17
1.3.2. Месенджери 18
1.3.3. Соціальні мережі 19
1.4. Важливість захисту інформації в системі дистанційного спілкування 21
1.4.1. Зростання популярності дистанційного спілкування..... 21
1.4.2. Застосування дистанційного спілкування у різних сферах 22
1.4.3. Ризики пов'язані з незахищеною інформацією 23
1.5. Основні загрози безпеці в дистанційному спілкуванні..... 24
1.5.1. Фішингові атаки та шахрайство 24
1.5.2. Взлом пристроїв та інформаційних систем..... 26
1.5.3. Перехоплення даних та прослуховування..... 27
1.5.4. Витоки масивів інформації 27
Висновки за розділом 29
РОЗДІЛ 2. ОСНОВНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ У СИСТЕМАХ ДИСТАНЦІЙНОГО СПІЛКУВАННЯ 32
2.1. Криптографічні методи захисту інформації..... 32
2.1.1. Аналіз алгоритмів шифрування..... 34
2.1.2. Цифровий підпис та електронний цифровий ключ 47
2.1.3. Захищені протоколи передачі даних..... 48

2.2. Захист мережевого сполучення та комунікації 50
2.2.1. Використання віртуальних приватних мереж *(VPN)* 50
2.2.2. Фірмові фаєрволи та інтрузійні системи виявлення 51
2.3. Захист автентифікації та авторизації 53
2.4. Захист даних та конфіденційності 54
2.5. Планування та управління інцидентами безпеки 57
2.6. Використання методу автентифікації зі сховищем даних 58
Висновки за розділом 62
РОЗДІЛ 3. ЗАБЕЗПЕЧЕННЯ НАДІЙНОГО ЗАХИСТУ У СИСТЕМАХ ДИСТАНЦІЙНОГО СПІЛКУВАННЯ 66
3.1. Середовище розробки – *"Android Studio"* 66
3.2. Технології, котрі використовуються для покращення безпеки інформації 67
3.2.1. Алгоритми *AES* 67
3.2.2. Генератор складних паролів 68
3.2.3. Обфускація коду додатку 68
3.2.4 Правила безпеки у базі даних 70
3.3. Принцип роботи системи дистанційного спілкування 71
3.4. Показники системи дистанційного спілкування 72
3.5. Практична реалізація технологій захисту інформації 73
3.5.1. Реалізація алгоритмів *AES* для шифрування повідомлень 73
3.5.2 Реалізація генерації складних паролів 75
3.5.3 Реалізація та налаштування обфускаторів 76
3.6. Тестування технологій захисту інформації 77
3.6.1. Тестування роботи алгоритму 77
3.6.2 Перевірка генерації складних паролів 78
Висновки за розділом 79
ВИСНОВКИ 83
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 89
ДОДАТОК А 91

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ПК	–	Персональний Комп’ютер
Wi-Fi	–	*Wireless Fidelity*
CSPRNG	–	*Cryptography secure pseudo-random number generators*
PBKDF2	–	*Password-Based Key Derivation Function*
AES	–	*Advanced Encryption Standard*
RSA	–	*Rivest-Shamir-Adleman*
DES	–	*Data Encryption Standard*
3DES	–	*3Data Encryption Standard*
ECB	–	*Electronic Codebook*
CBC	–	*Cipher Block Chaining*
CFB	–	*Cipher Feedback*
OFB	–	*Output Feedback*
EDE	–	*Encrypt, Decrypt, Encrypt*
EEE	–	*Encrypt, Encrypt, Encrypt*
ECC	–	*Elliptic Curve Cryptography*
NIST	–	*National Institute of Standards and Technologies*
SSL	–	*Secure Sockets Layer*
TLS	–	*Transport Layer Security*
CSPRNG	–	*Cryptography secure pseudo-random number generators*
SSL	–	*Secure Sockets Layer*
TLS	–	*Transport Layer Security*
IPsec	–	*Internet Protocol Security*
VPN	–	*Virtual Private Network*
SSH	–	*Secure Shell*
HTTPS	–	*Hypertext Transfer Protocol Secure*
HTTP	–	*HyperText Transfer Protocol*
DDoS	–	*(Distributed) Denial-of-Service attack*

Cisco ASA	–	*Cisco Adaptive Security Appliance*
NIST	–	*Національний інститут стандартів і технології*
XOR	–	*eXclusive OR*
ASCII	–	*American Standard Code for Information Interchange*

ВСТУП

Зростання кількості населення та його розподіл по всьому світу вимагає ефективних інструментів спілкування. Однією з ключових технологій, яка дозволяє нам уникнути соціальної ізоляції та забезпечити продовження міжособистого спілкування в умовах обмежень, стали системи дистанційного спілкування.

Із урахуванням подій останніх років, люди були зобов'язані оптимізувати, покращити і розширити можливості систем дистанційного спілкування. Все почалося із пандемії COVID-19, коли майже все населення Землі було прикуте до карантинних умов. Деякі люди дуже довгий час майже не виходили на вулиці, позакривалися робочі місця, деякі почали працювати вдома, призупинилося очне навчання у дитсадках, школах, університетах. Життя яке ми знали раніше, могло б зупинитись і заморозитись на певний термін. Але, необхідність комунікувати, підштовхнуло людство до більш поглибленого і широкого використання технологій та систем дистанційного спілкування.

Ці системи, допомогли знайти вихід з глухого та поодинокого життя у чотирьох стінах. Колись люди, працюючи у команді маючи живе спілкування перейшли на домашню роботу, з'явилася різка необхідність почати оптимізовувати робочі процеси, налагоджувати зв'язки та вчитись жити по-новому. У цей момент почався пік зростання та розвитку систем дистанційного спілкування. Колись начебто прості та не зовсім потрібні технології та інструменти вийшли на першу позицію у найактуальніших та необхідних для людства речей. Адже людям не тільки потрібно заново вчитись працювати, а й підтримувати зв'язки з родинами, друзями, замовляти продукти, користуватись сервісами та послугами перебуваючи більшість часу вдома.

Так само, ще одним випробуванням для нас та для систем дистанційного спілкування стала війна. Зараз відбувається війна технологій та онлайн комунікація у ній відіграє одну з провідних ролей. Адже вчасно передане та

отримане повідомлення, може як врятувати комусь життя, так і надати тактичну перевагу над ворогом.

Отже, провівши критичний аналіз проблематики предмету кваліфікаційного дослідження, стає відомо, що системи дистанційного спілкування, займають провідні позиції серед необхідних систем для людства і є актуальними та вкрай необхідними.

Для будь-якої системи необхідні пристрої, на сьогоднішній день, комп’ютери та смартфони займають лідируючі позиції у використанні користувачами для дистанційного спілкування. Проте вони мають свої унікальні характеристики та можливості, що варто розглянути у контексті їхнього використання для віддалених зв'язків.

Комп'ютери, завдяки своїй потужності та більшому екрану, зазвичай вважаються більш зручними для продуктивної роботи та великих обсягів інформації. Вони часто використовуються для відеоконференцій, великих проектів, редагування великих файлів та роботи зі складними програмами. Такі комп'ютери зазвичай мають більше обчислювальної потужності та можуть бути ефективними для бізнес-контексту, освіти та професійної роботи.

З іншого боку, смартфони відрізняються своєю мобільністю та доступністю. Вони компактні, легкі для перенесення та завжди “під рукою”. Смартфони забезпечують широкий спектр можливостей для дистанційного спілкування завдяки великому розповсюдженню месенджерів, соціальних мереж, відеодзвінків та додатків для віртуальних конференцій. Вони забезпечують зручність спілкування у будь-якому місці, де є доступ до Інтернету.

Обидва пристрої мають свої переваги та обмеження, але в контексті дистанційного спілкування смартфони, завдяки їхній мобільності та простоті використання, часто виявляються універсальнішими для повсякденних комунікаційних потреб. З іншого боку, комп'ютери залишаються незамінними у випадках, коли потрібна велика потужність обчислення та робота з великими обсягами даних.

Задля порівняння який пристрій є більш актуальним у використанні у системах дистанційного спілкування, проведемо порівняльний аналіз комп’ютерів та смартфонів за такими критеріями: вартість, легкість у використанні, портативність, швидкість у використанні, кількість додатків.

1. Вартість:

– Смартфони - є більш доступними, оскільки їхня ціна менша за комп’ютер.

– Комп’ютери - є більш вартісними і вимагають придбання додаткових вкладень у вигляді клавіатури та миші, окрім ноутбуків.

2. Легкість у використанні:

– Смартфони - система, зазвичай, інтуїтивно зрозуміла, а вбудовані датчики (сенсорний екран, вбудовані камери) роблять взаємодію простою та зручною.

– Комп’ютери - операційна система зазвичай складніша. Хоча для деяких користувачів це може бути зручним, для інших це може вимагати більшої кількості часу на вивчення та пристосування.

3. Портативність:

– Смартфони - є набагато більш портативними, оскільки вони можуть бути легко перенесені в кишені чи сумці. Це дає можливість користуватися ними будь-де і будь-коли.

– Комп’ютери - комп'ютери, навіть ноутбуки, важче та більші за розмірами, що робить їх менш портативними порівняно з смартфонами. Однак, вони все ще можуть бути перенесені з одного місця в інше.

4. Швидкість у використанні:

– Смартфони - сучасні смартфони оснащені швидкими процесорами та достатньою кількістю оперативної пам'яті, що дозволяє їм працювати досить швидко. Однак, швидкість може залежати від моделі та його рівня продуктивності.

– Комп’ютери - сучасні комп’ютери також відмінно справляються із задачами систем дистанційного спілкування.

5. Кількість додатків:

– Смартфони - на сьогоднішній день у магазинах додатків для смартфонів (таких як App Store та Google Play) є величезна кількість додатків.

– Комп’ютери - мають також достатню кількість додатків, але все ж таки меншу, ніж смартфони.

На сьогоднішній день, системи дистанційного спілкування проходять неймовірну кількість випробувань. Існує багато різних способів нашкодити користувачеві та його конфіденційній інформації. Хакери, шахраї та аферисти постійно шукають методи та способи обійти захист програмного забезпечення та заволодіти, або пошкодити, видалити певні дані, тому також розглянемо та порівняємо безпеку комп’ютерів та смартфонів, адже обидва типи пристроїв піддаються різним загрозам.

Комп'ютери, завдяки своїм складним операційним системам, можуть мати більше програмних вразливостей, а надаючи користувачам доступ до будь-яких тонких налаштувань системи, дають недосвідченим користувачам можливість власноруч нашкодити своїй безпеці. Також шанс отримати вірус з інтернету при відвідуванні сайтів та завантажень програм набагато вищій, чим у смартфонів. Обидва пристрої однаково піддаються соціальній інженерії та фішинговим атакам через месенджери та електронну пошту. З іншого боку, смартфони, які мають доступніші та простіші для використання операційні системи, мають набагато менше вразливих місць, вірусів та шкідливих програмних забезпечень, а всі додатки, що потрапляють у магазини додатків проходять сувору перевірку і користувачі можуть завантажити тільки безпечні та перевірені додатки, що мінімізує загрозу безпеці. Також варто відмітити, що будь-яка система не буде надійною, якщо користувач не володіє базовою інформацію по захисту своєї персональної інформації.

Підсумовуючи отримані результати від порівняльного аналізу комп’ютерів та смартфонів як пристроїв, для систем дистанційного спілкування та створення порівняльної таблиці, можна зробити висновок, що смартфони краще підходять на цю роль, чим персональні комп’ютери.

Таблиця 1.1

Порівняння персональний комп’ютерів та смартфонів для дистанційного спілкування

	ПК	Смартфони
Вартість	-	+
Легкість у використанні	-	+
Портативність	-	+
Швидкість у використанні	-	+
Кількість додатків	-	+
Безпека	-	+

Зі збільшенням популярності та користувацької бази збільшується ризик порушення конфіденційності та безпеки даних користувачів. Запобігання доступу до особистих інформаційних ресурсів, захист конфіденційної інформації та даних стають надзвичайно важливими завданнями для розробників та користувачів цих систем.

Загальноприйняті уявлення про безпеку та засоби захисту користувачів у дистанційному спілкуванні стали ключовими напрямками досліджень та розвитку в цій галузі. Наприклад, застосування криптографічних методів, розвиток антивірусів та фірмових захистів, а також навчання користувачів основам цифрової безпеки - усі ці аспекти стали невід'ємною частиною розвитку систем дистанційного спілкування.

Певно ніхто не хотів опинитися на місці людини, у котрої вкрали дані банківської карти, якісь паролі, або персональні дані. Тому постійний розвиток методів захисту інформації повинен бути всюди.

Таким чином, системи дистанційного спілкування стали не тільки невід'ємною частиною нашого повсякденного життя, але й ключовим інструментом для збереження нашої соціальної, професійної та особистої спрямованості у змінному світі. Їхня безпека та захист інформації в них стає пріоритетним завданням для усіх учасників цього процесу. Тому, кожен

користувач систем дистанційного спілкування повинен знати про загрози для інформації та методи її захисту, користуватися ними та не наражатись на небезпеку.

Метою даної магістерської роботи є детальний огляд, аналіз та практична реалізація методів, алгоритмів захисту інформації у системах дистанційного спілкування. Ця робота дослідила принципи захисту користувацької інформації, що передається через смартфони на базі операційної системи *“Android”* .

РОЗДІЛ 1
ОСНОВНІ ВИЗНАЧЕННЯ, ЗАГРОЗИ ТА ВАЖЛИВІСТЬ ЗАХИСТУ ІНФОРМАЦІЇ У СИСТЕМАХ ДИСТАНЦІЙОГО СПІЛКУВАННЯ

1.1. Визначення системи дистанційного спілкування

Система - це сукупність взаємопов'язаних елементів, які працюють разом для досягнення певної мети. В контексті комп'ютерних технологій, система може означати програмне забезпечення, апаратне забезпечення або комбінацію обох, які співпрацюють для виконання конкретних функцій або вирішення задач.

Елементом системи називають найпростіший компонент системи, який умовно вважають неподільним. Поняття неподільності є умовним і визначається залежно від конкретних завдань. Наприклад, розглядаючи літальний апарат як систему, немає необхідності брати до уваги атомну будову його елементів.

Підсистема — це компонент системи, який сам є системою.

У сукупності елементи та підсистеми називаються компонентами системи. Поділ системи на окремі елементи та підсистеми є неоднозначним і залежить від мети та конкретних завдань дослідження.

Система дистанційного спілкування - це система, що забезпечує комунікацію між віддаленими користувачами за допомогою різних технологій та засобів. Вона дозволяє людям обмінюватися повідомленнями, аудіо- та відеоданими, спільно працювати над проектами та інтерактивно взаємодіяти незалежно від свого фізичного розташування.

Системи дистанційного спілкування можуть включати такі компоненти, як електронна пошта, миттєві повідомлення, відеоконференції, спільні робочі простори, вебінари та інші інструменти для забезпечення ефективного спілкування на відстані.

1.2. Роль технологій у розвитку дистанційного спілкування

Технології відіграють надзвичайно важливу роль у розвитку дистанційного спілкування. Вони надають засоби та інфраструктуру для забезпечення ефективного зв'язку між людьми, які знаходяться на відстані. Ось деякі ролі, які технології відіграють у розвитку дистанційного спілкування:

– Технології забезпечення зв’язку, такі як Інтернет, мобільна телефонія та супутникові зв'язок, дозволяють людям спілкуватися у режимі реального часу, незалежно від фізичного розташування. Вони забезпечують передачу повідомлень, голосу, відео та інших форматів даних, що дозволяє людям спілкуватися безпосередньо через різні пристрої.

– Технології відеозв'язку, такі як веб-камери, мікрофони та програмне забезпечення для відеоконференцій, дозволяють людям бачити та чути одне одного в реальному часі. Це робить дистанційне спілкування більш особистим та взаємодійним, а також сприяє покращенню комунікації та розумінню між віддаленими учасниками.

– Технології спільної роботи, такі як хмарні сервіси та онлайн-інструменти для спільного редагування документів, дозволяють групам людей працювати разом над одними й тими ж документами чи проектами незалежно від їх фізичного розташування. Це полегшує співпрацю та обмін ідеями на відстані.

– Технології миттєвих повідомлень (наприклад, чати та месенджери) та електронної пошти дозволяють людям обмінюватися повідомленнями та інформацією у вигляді тексту, документів, зображень тощо. Вони забезпечують швидкий та зручний спосіб комунікації, який може бути використаний у будь-який час та в будь-якому місці.

В цілому, технології розвиваються та вдосконалюються, щоб забезпечити більш зручне, ефективне та реалістичне дистанційне спілкування. Вони допомагають зменшити географічні обмеження, збільшити доступність та покращити якість комунікації між віддаленими людьми [1].

1.3. Популярні форми дистанційного спілкування

1.3.1. Відеоконференції

Відеоконференції - це форма дистанційного спілкування, яка дозволяє людям обмінюватися аудіо- та відеоданими в реальному часі за допомогою веб-камер, мікрофонів та спеціального програмного забезпечення або платформ.

Основна ідея відеоконференцій полягає в тому, щоб дозволити учасникам бачити одне одного та спілкуватися як у реальному житті, хоча вони можуть знаходитися в різних місцях. Вони використовуються в багатьох сферах, включаючи бізнес, освіту, медицину, соціальні зустрічі та інші області.

Для проведення відео конференцій використовуються різні платформи та інструменти, такі *як Zoom* (рис. 1.1), *Microsoft Teams*, *Google Meet* (рис. 1.2), *Skype* та інші. Ці платформи надають можливість створювати віртуальні зустрічі, запрошувати учасників, демонструвати презентації, проводити обговорення та співпрацювати над спільними проектами.

Рис. 1.1. Логотип додатку для відеконференції “*Zoom*”

Рис. 1.2. Приклад відеоконференції *Google Meet* від *Google*

Відеоконференції мають багато переваг, включаючи зменшення необхідності у фізичних зустрічах, збереження часу і витрат на подорожі, покращення комунікації та співпраці між віддаленими учасниками, можливість проведення зустрічей з великою кількістю людей тощо.

Однак, варто враховувати деякі недоліки відеоконференцій, такі як можливе нестабільне Інтернет-підключення, що призведе до проблем зі звуком або зображенням, а також втрату непрямого контакту та невербальної комунікації, яка може бути присутня під час особистої зустрічі.

1.3.2. Месенджери

Миттєві повідомлення (чати), часто називані чатами, є популярною формою дистанційного спілкування, яка дозволяє людям обмінюватися текстовими повідомленнями в режимі реального часу. Ця форма спілкування використовується як для особистих розмов, так і для комунікації в робочих та групових контекстах.

Текстові повідомлення в режимі реального часу дозволяють вам обмінюватися текстовими повідомленнями безпосередньо з іншими користувачами у реальному часі. Ви можете надсилати повідомлення, які миттєво з'являються на екрані отримувача, що забезпечує швидку комунікацію.

Один на один та групове спілкування можуть бути використані для приватних розмов між двома користувачами або для комунікації в групах. Ви можете створювати групові чати, в яких кілька осіб може спілкуватися одночасно.

Мультимедійний контент у багатьох миттєвих повідомленнях ви можете надсилати не тільки текст, але й різні типи мультимедійного контенту, такі як фотографії, відео, аудіозаписи, документи тощо. Це дозволяє більш розширено виражати свої думки та ідеї.

Синхронна та асинхронна комунікація дозволяє вам вести розмови у режимі реального часу, що сприяє швидкій обміну повідомленнями. Водночас, ви також можете залишати повідомлення відсутнім користувачам, які побачать їх, коли знову будуть в мережі.

Миттєві повідомлення широко використовуються в різних сферах, включаючи особисте спілкування, роботу в команді, викладання та багато інших. Вони стали невід'ємною частиною нашого цифрового життя.

Популярні месенджери (рис. 1.3) та платформи для миттєвих повідомлень включають *WhatsApp, Telegram* (рис.1.4), *Facebook Messenger, Signal, Slack, Microsoft Teams, WeChat* та інші. Кожен з них має свої особливості та можливості, але загальна мета полягає в забезпеченні швидкого та зручного спілкування у віртуальному просторі.

Рис. 1.3. Приклад месенджерів

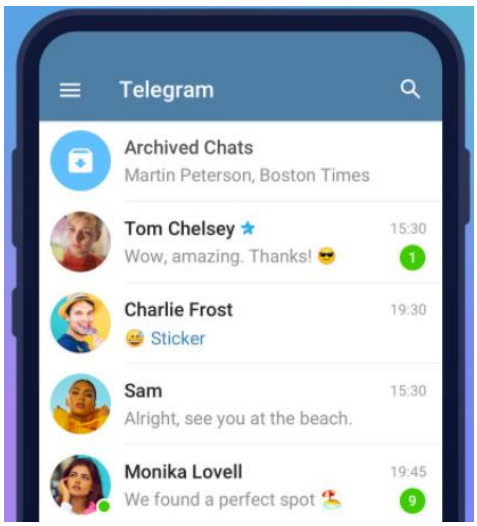

Рис. 1.4. Приклад месенджера *Telegram*

1.3.3. Соціальні мережі

Соціальні мережі - це онлайн-платформи, які дозволяють людям створювати профілі, спілкуватися з іншими користувачами, ділитися контентом (текстовими повідомленнями, фотографіями, відео), виражати свої думки та інтереси, підтримувати зв'язки та взаємодіяти в онлайн-спільнотах. Також деякі соціальні мережі використовуються для бізнесу, користувачі ведуть в них свої магазини соціальні мережі стали надзвичайно популярними і впливають на спосіб, яким люди спілкуються та сприймають інформацію.

Основними аспектами соціальних мереж (рис. 1.5), (рис. 1.6), є профілі та мережі зв’язків, публікації контенту, спілкування, онлайн спільноти та групи, реклама та комерційна діяльність.

Користувачі створюють профілі, де можуть надавати особисту інформацію, таку як ім'я, фотографію, інтереси, дату народження тощо. Вони можуть зв'язуватися з іншими користувачами, додавати їх до своєї мережі зв’язків та обмінюватися повідомленнями.

Користувачі можуть публікувати різноманітний контент, такий як текстові статуси, фотографії, відео, посилання тощо. Інші користувачі можуть коментувати, вподобати та ділитися цим контентом.

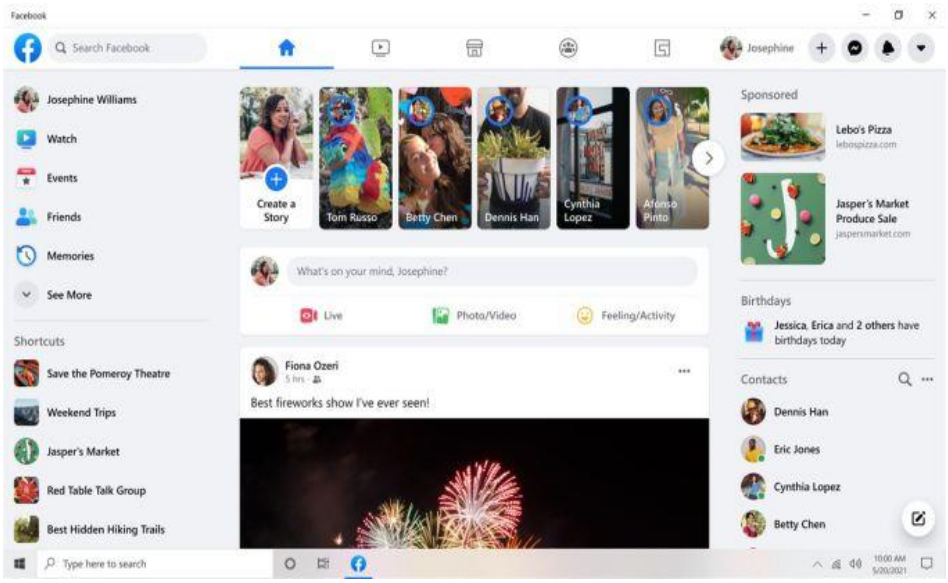

Рис. 1.5. Приклад соціальної мережі *Facebook*

Рис. 1.6. Логотипи популярних соціальних мереж

Соціальні мережі надають можливість спілкуватися з іншими користувачами за допомогою коментарів, приватних повідомлень або групових чатів. Це дозволяє людям обговорювати теми, висловлювати свою думку та взаємодіяти один з одним.

Соціальні мережі дозволяють користувачам приєднуватися до спільнот та груп інтересів, де вони можуть обмінюватися інформацією, обговорювати теми та знаходити спільну мову з людьми з подібними інтересами.

Багато соціальних мереж надають можливість рекламувати товари та послуги, створювати бізнес-сторінки та розміщувати рекламні матеріали, що дозволяє компаніям взаємодіяти зі своїми клієнтами та залучати нових користувачів.

Деякі з найпопулярніших соціальних мереж включають *Facebook Instagram, Twitter, LinkedIn, Snapchat, TikTok* та інші. Кожна соціальна мережа має свої унікальні особливості та аудиторію, і вони можуть бути використані для різних цілей, включаючи особисте спілкування, маркетингові кампанії, професійне мережування тощо.

1.4. Важливість захисту інформації в системі дистанційного спілкування

1.4.1. Зростання популярності дистанційного спілкування

Зростання популярності дистанційного спілкування є очевидним явищем, особливо в контексті сучасних технологій.

За останні роки відбулося значне покращення технологій спілкування, зокрема відеоконференцій, миттєвих повідомлень та соціальних мереж. Висока доступність швидкого Інтернету, покращення якості зображення та звуку, а також зручність використання додали зручності та ефективності до дистанційного спілкування.

За допомогою дистанційного спілкування люди можуть спілкуватися і співпрацювати в будь-якій точці світу без значних перешкод. Це особливо важливо для міжнародних бізнес-комунікацій, віддалених робочих груп і зв'язку з клієнтами з усього світу.

Дистанційне спілкування дозволяє зв'язуватися з іншими людьми без фізичного присутності. Це зберігає час і зусилля, оскільки не потрібно

подорожувати або витрачати час на організацію зустрічей. Крім того, дистанційне спілкування дозволяє багатофункціональність, включаючи спільну роботу над проектами, обмін документами і інше.

Зростання тренду віддаленої роботи також сприяє популярності дистанційного спілкування. Завдяки технологіям дистанційного спілкування компанії можуть забезпечити комунікацію та співпрацю між співробітниками, які працюють з різних місць, що дає більшу гнучкість і можливість працювати з будь-якої точки світу.

Пандемія *COVID-19* суттєво змінила наш спосіб життя і роботу, змушуючи багатьох людей переходити на дистанційний режим роботи та спілкування. Це призвело до збільшення використання дистанційних засобів спілкування, таких як відеоконференції та миттєві повідомлення.

1.4.2. Застосування дистанційного спілкування у різних сферах

Дистанційне спілкування знайшло своє застосування у багатьох сферах життя і діяльності. Ось кілька прикладів його використання в різних сферах:

– Для бізнесу і роботи дистанційне спілкування дозволяє ведення відеоконференцій, онлайн-засідань, обмін документами і спільну роботу над проектами. Воно допомагає зв'язати команди, що працюють з різних місць, спілкуватися з клієнтами і партнерами з усього світу і ефективно організовувати віддалену роботу. У додачу введення всіх робочих конференцій і виконання завдань за допомогою дистанційного спілкування набагато скорочує витрати компаній на аренду офісів.

– Для освіти дистанційне спілкування використовується в онлайн-навчанні, віртуальних лекціях, вебінарах та дистанційних курсах. Воно дозволяє студентам і викладачам зв'язуватися, обмінюватися матеріалами, виконувати завдання і проводити віддалені екзамени у раз виникнення надзвичайних ситуацій, котрі не дають можливості проводити навчання в навчальних установах.

– Для медицини дистанційне спілкування дозволяє пацієнтам консультуватися з лікарями в режимі онлайн, проводити віддалені консультації,

моніторинг стану пацієнтів та передавати медичні дані. Це особливо корисно для людей, які мешкають віддалено або мають обмежену можливість пересування.

– Соціальні мережі та миттєві повідомлення дозволяють людям з усього світу спілкуватися, обмінюватися новинами, фотографіями та відео, знаходити старих друзів і знайомитися з новими людьми.

– Дистанційне спілкування дозволяє організовувати віртуальні конференції, семінари, виставки, концерти та інші події. Воно забезпечує можливість участі людей з різних країн без фізичного присутності.

Це лише декілька прикладів застосування дистанційного спілкування. Завдяки постійному розвитку технологій, його можливості стають все більш широкими і доступними в різних сферах життя.

1.4.3. Ризики пов'язані з незахищеною інформацією

Захист інформації в системі дистанційного спілкування є надзвичайно важливим аспектом. Оскільки дистанційне спілкування відбувається через мережу Інтернет, інформація, що передається та зберігається, може бути під загрозою безпеки і підпадати під ризик втрати конфіденційності, цілісності та доступності даних. Ось деякі аспекти важливості захисту інформації в системі дистанційного спілкування:

– Захист конфіденційної інформації є ключовим аспектом. Під час дистанційного спілкування передаються особисті дані, комерційна інформація, медичні записи, банківські дані та інші конфіденційні відомості. Необхідно забезпечити, щоб ці дані були захищені від несанкціонованого доступу та перехоплення.

– Цілісності інформації означає забезпечення того, що дані не підлягають незаконним змінам, втратам чи спотворенням під час передачі або зберігання. Важливо, щоб інформація, яку ви обмінюєте через дистанційне спілкування, була надійно захищена від несанкціонованого втручання.

– Забезпечення автентифікації важливо для підтвердження ідентичності користувачів, які беруть участь у дистанційному спілкуванні. Це допомагає уникнути підробки або несанкціонованого доступу до системи.

– Дистанційне спілкування може піддаватися різноманітним зловмисним атакам, таким як фішинг, віруси, шпигунське програмне забезпечення тощо. Захист від таких атак є критичним для збереження безпеки та приватності в системі дистанційного спілкування.

– Забезпечення надійного з'єднання та захисту від технічних перешкод є також важливим аспектом. Надійна мережева інфраструктура та захист від відмови сервісу (*DDoS*) допомагають забезпечити доступність та продуктивність системи дистанційного спілкування.

Усі ці аспекти підкреслюють важливість захисту інформації в системі дистанційного спілкування. Організації та користувачі повинні приділяти належну увагу захисту своїх даних, використовуючи захисні механізми, шифрування, автентифікацію та інші методи, які допоможуть забезпечити безпеку та конфіденційність у процесі дистанційного спілкування (рис. 1.7) [2].

Рис. 1.7. Ілюстрація умовно захищеної приватної інформації

1.5. Основні загрози безпеці в дистанційному спілкуванні

1.5.1. Фішингові атаки та шахрайство

Фішинг - це метод соціального інженерінгу, при якому зловмисник представляється довіреною особою або організацію (наприклад, банк, соціальну мережу або електронну платформу) і намагається залучити людей до надання своїх особистих даних, таких як паролі, номери кредитних карток, соціальний номер тощо. Це може бути зроблено через підроблені електронні листи, веб-сайти або повідомлення.

Спам-листими та фішинговими веб-сайти зловмисники можуть використовувати спам-листи, щоб розсилати масові електронні листи з

фішинговими повідомленнями, що містять підроблені логотипи та вигляд, що схожий на офіційні повідомлення від відомих організацій. Вони також можуть створювати фішингові веб-сайти, що схожі на офіційні сторінки, з метою отримання особистої інформації.

Фішингові атаки зазвичай використовують соціальний інженерінг, тобто маніпуляцію психологічних факторів та емоцій людей, щоб вони відкрили свою конфіденційну інформацію. Зловмисники можуть створювати ситуації, які стимулюють поспіх, страх або небезпеку, щоб змусити людей реагувати швидко та бездумно, надаючи свої особисті дані. Вони також можуть створювати фішингові веб-сайти, що схожі на офіційні сторінки, з метою отримання особистої інформації. Крім фішингових атак, існує також широкий спектр інших шахрайських дій, які можуть бути виконані через дистанційне спілкування. Це включає шахрайство з використанням підроблених профілів, підроблення продуктів або послуг, маніпуляцію фінансовою інформацією, вимагання викупу (рансому) та інші шахрайські схеми. Фішингові атаки (рис. 1.8) та шахрайство є типами шахрайських дій, які спрямовані на отримання конфіденційної інформації, такої як паролі, особисті дані, фінансова інформація, шляхом підробки або маніпуляцій з метою обману людей.

Рис. 1.8. Ілюстрація фішингової атаки

1.5.2. Взлом пристроїв та інформаційних систем

Взлом пристроїв та інформаційних систем є серйозною загрозою для безпеки та конфіденційності даних. Він може призвести до несанкціонованого доступу до особистої інформації, крадіжки даних, фінансових втрат,

розповсюдження шкідливого програмного забезпечення та інших небажаних наслідків. Ось деякі важливі аспекти взлому пристроїв та інформаційних систем:

– Багато вразливого програмного забезпечення та випадків взлому пристроїв та систем пов'язані з використанням вразливостей у програмному забезпеченні. Зловмисники можуть використовувати вразливості операційних систем, додатків, браузерів або інших компонентів для здійснення несанкціонованого доступу.

– Інколи зловмисники можуть отримати віддалений доступ до пристроїв або систем через використання слабких паролів, недостатньої захисту мережі або недостатньо оновленого програмного забезпечення.

Для захисту від взлому пристроїв та інформаційних систем рекомендується:

– Встановлювати оновлення програмного забезпечення, включаючи операційні системи та додатки.

– Використовувати сильні та унікальні паролі для всіх облікових записів і забезпечувати їх регулярну зміну.

– Увімкнути двоетапну автентифікацію для своїх облікових записів, де це можливо.

– Бути пильними стосовно спаму, фішингових повідомлень та підозрілих веб-сайтів. Не надавати особисту інформацію або натискати на посилання, якщо вони виглядають сумнівно.

– Використовувати надійне антивірусне програмне забезпечення та брандмауери для захисту від шкідливого програмного забезпечення та несанкціонованого доступу.

– Бути обережним при завантаженні файлів або відкритті прикріплених файлів у електронних листах, особливо від невідомих джерел.

Враховуючи ці заходи безпеки, можна зменшити ризик взлому пристроїв та інформаційних систем і зберегти свою інформацію в безпеці.

1.5.3. Перехоплення даних та прослуховування

Перехоплення даних та прослуховування є методами незаконного отримання доступу до конфіденційної інформації, яка передається через мережі.

Ці методи можуть бути використані зловмисниками з метою зламу або викрадення даних, які передаються між пристроями або через Інтернет. Ось деякі важливі аспекти перехоплення даних та прослуховування:

– Зловмисники можуть використовувати програмне забезпечення або пристрої для перехоплення і аналізу пакетів даних, що передаються по мережі, це називається *Packet Sniffing* (прослуховування пакетів). Вонии можуть отримувати доступ до цих пакетів, що містять конфіденційну інформацію, таку як логіни, паролі, фінансові дані і т.д.

– При атаках *Man-in-the-Middle* (*MITM*) зловмисники розміщуються між комунікуючими сторонами і перехоплюють або змінюють передану інформацію. Зловмисники можуть відтворити з'єднання між двома сторонами і перехоплювати всю інформацію, яка надсилається між ними, включаючи паролі, фінансові дані тощо.

– У випадку використання незахищених *Wi-Fi* мереж або мереж зі слабким шифруванням, зловмисники можуть перехоплювати інформацію, яка надсилається через такі мережі, це називається *Wi-Fi* Інтерцептування. Вони можуть отримати доступ до приватних даних, які передаються між пристроями, таких як паролі, електронні листи, фінансові дані і т.д.

– Зловмисники можуть використовувати віруси, троянські програми.

1.5.4. Витоки масивів інформації

Зливи баз даних з користувацькою конфіденційною інформацією є серйозним порушенням безпеки даних, коли конфіденційна інформація, така як імена, адреси, номери банківських рахунків, фінансові дані або інші особисті дані, незаконно витікають з бази даних і потрапляють у недозволений доступ.

Причини зливів баз даних можуть бути різноманітними починаючи від кібератак, коли зловмисники можуть намагатися проникнути в систему або базу даних, використовуючи різні методи, такі як *SQL*-ін'єкції, вторгнення з використанням вразливостей програмного забезпечення. Іноді зливи баз даних відбуваються через зловживання працівниками, які мають прямий доступ до конфіденційної інформації. Це може бути ненавмисна помилка. Втрата або

крадіжка фізичних носіїв таких як сервери, комп'ютери або зовнішні жорсткі диски, втрачаються або крадуться, це може призвести до неправомірного доступу до збереженої на них інформації. Наслідки зливів баз даних можуть бути серйозними, включаючи крадіжку особистої ідентичності, фінансові збитки, порушення конфіденційності та втрату довіри з боку користувачів. Задля запобігання зливам баз даних рекомендуються регулярно оновлювати програмного забезпечення, щоб запобігти використанню вразливостей зловмисниками, забезпечувати належний контроль доступу до баз даних, використовуючи сильні паролі, двоетапну перевірку або інші методи автентифікації. Для нових користувачів треба проводити аудит безпеки для виявлення можливих слабких місць у системі та базах даних, навчати співробітників щодо кібербезпеки, соціального інжинірингу та правил безпеки даних [3].

Важливо приділяти належну увагу захисту баз даних та приймати відповідні заходи безпеки для запобігання зливам і витокам конфіденційної інформації.

Висновки за розділом

Перший розділ надає детальну теоретичну інформацію про визначення терміну система, системи дистанційного спілкування, її актуальність, популярні форми та про важливість захист інформації що надходить і виходить з таких систем.

Система - це сукупність взаємопов'язаних елементів, які працюють разом для досягнення певної мети.

Система дистанційного спілкування - це система, що забезпечує комунікацію між віддаленими користувачами за допомогою різних технологій та засобів. Вона дозволяє людям обмінюватися повідомленнями, аудіо - та відеоданими, спільно працювати над проектами та інтерактивно взаємодіяти незалежно від свого фізичного розташування.

Технології відіграють надзвичайно важливу роль у розвитку дистанційного спілкування. Вони надають засоби та інфраструктуру для забезпечення ефективного зв'язку між людьми, які знаходяться на відстані.

Популярні форми дистанційного спілкування: відеоконференції, месенджери та соціальні мережі.

Відеоконференція – це форма дистанційного спілкування, яка дозволяє людям бачити і чути одне одного за допомогою веб-камер та мікрофонів. Відеоконференції використовуються для проведення онлайн-зустрічей, вебінарів, телеконференцій та інших подібних заходів. Для проведення відеоконференцій використовуються різні платформи та інструменти, такі як *Zoom, Microsoft Teams, Google Meet, Skype* та інші.

Месенджери (чати) – чати та месенджери, які дозволяють надсилати текстові повідомлення у реальному часі. Популярні месенджери, такі як *WhatsApp, Telegram, Slack, Microsoft Teams*, дозволяють спілкуватися один на один або у групах. Ця форма спілкування використовується як для особистих розмов, так і для комунікації в робочих та групових контекстах. Чати дозволяють вести розмови у режимі реального часу, що сприяє швидкій обміну повідомленнями. В той же час, ви також можете залишати повідомлення відсутнім користувачам, які побачать їх, коли знову будуть в мережі.

Соціальні мережі - платформи соціальних мереж, такі як *Facebook, Instagram, Twitter, LinkedIn*, дозволяють людям створювати профілі, спілкуватися з іншими користувачами, ділитися контентом (текстовими повідомленнями, фотографіями, відео), виражати свої думки та інтереси, підтримувати зв'язки та взаємодіяти в онлайн-спільнотах. Соціальні мережі стали надзвичайно популярними і впливають на спосіб, яким люди спілкуються та сприймають інформацію.

За останні роки відбулося значне покращення технологій спілкування, зокрема відеоконференцій, миттєвих повідомлень та соціальних мереж. Висока доступність швидкого Інтернету, покращення якості зображення та звуку, а також зручність використання додали значущості та ефективності до дистанційного

спілкування. Зростання тренду віддаленої роботи також сприяє популярності дистанційного спілкування.

Дистанційне спілкування знайшло своє застосування у багатьох сферах життя і діяльності. У таких сферах як: бізнес і робота, освіта і наука, медицина, соціальні зв’язки, організації заходів, від управління невеличкими процесами до керування компаніями та вирішенням політичних питань. Тому системи дистанційного спілкування зараз на часі і є неймовірно актуальними і вже встигли переплестись з багатьма сферами життя.

Основними загрозами інформації є:

– Вразливості програмного забезпечення.

– Віддалений або фізичний доступ до пристроїв або носіїв інформації.

– Фішингові атаки, кібератаки та шахрайство.

– Соціальний інжиніринг.

– *Packet Sniffing* (прослуховування пакетів).

– *Man-in-the-Middle (MITM)* атаки.

– *Wi-Fi* Інтерцептування.

– Віруси та шпигунське програмне забезпечення.

Захист інформації в системі дистанційного спілкування є надзвичайно важливим аспектом. Оскільки дистанційне спілкування відбувається через мережу Інтернет, інформація, що передається та зберігається, може бути під загрозою безпеки і підпадати під ризик втрати, викраденню конфіденційності інформації, її цілісності та доступності.

РОЗДІЛ 2
ОСНОВНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ У СИСТЕМАХ ДИСТАНЦІЙНОГО СПІЛКУВАННЯ

2.1. Криптографічні методи захисту інформації

Криптографія - це наука про захист інформації шляхом шифрування і розшифрування даних. Головні принципи криптографії є забезпечення конфіденційності даних, це означає що лише авторизовані особи повинні мати доступ до зашифрованих даних, іншим недозволено їх переглядати чи отримувати зрозумілу інформацію з них. Криптографія зберігає цілісність даних, забезпечує недоступність, зміну або порушення даних під час передачі. Це досягається за допомогою хеш-функцій, цифрових підписів та кодів перевірки цілісності. Методи автентифікація встановлює ідентичність сторін, що комунікують між собою, для підтвердження їх відповідності встановленим правилам. Вона гарантує, що сторони, що обмінюються даними, є тими, за кого вони себе видають. Принципи незаперечності забезпечує доказову силу ідентифікації та авторства повідомлення. Це означає, що відправник не може відкинути факт відправки повідомлення, а отримувач не може відкинути факт його отримання. А принцип Криптографія використовує ключі для шифрування та розшифрування даних. Ключі можуть бути симетричними (коли один і той же ключ використовується для шифрування та розшифрування) або асиметричними (коли використовуються два ключі: публічний для шифрування та приватний для розшифрування). Криптографічні алгоритми повинні бути відмовостійкими, тобто відомий шифротекст чи ключ не повинні допомагати зловмисникам відкрити інші повідомлення чи ключі.

Процес шифрування персональних даних починається з вибору шифрувального алгоритму. Існує багато різних алгоритмів, таких як *AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), DES (Data*

Encryption Standard) тощо. Вибір алгоритму залежить від рівня безпеки, швидкодії та вимог конкретної системи. Після обрання потрібного шифрувального алгоритму необхідно згенерувати ключ для шифрування та розшифрування даних. Ключ може бути симетричним або асиметричним. Для генерації симетричного ключа рекомендується використовувати генератор випадкових чисел або криптографічно стійку функцію хешування для створення випадкового набору символів для ключа. Найбільш розповсюдженим методом є використання криптографічно стійкого генератора випадкових чисел (*CSPRNG - Cryptography secure pseudo-random number generators)*. Також можна використовувати пароль або фразу для генерації ключа шифрування за допомогою функції хешування або ітеративними методами, такими як *PBKDF2* або *bcrypt*. Для асиметричного шифрування, необхідно згенерувати ключову пару, що складається з приватного та публічного ключів. Для цього використовується криптографічний алгоритм, який генерує ці ключі, наприклад, *RSA keypair generation algorithm*, або ж можна використати методи, котрі описані для генерації синхронного ключа, але створити відповідно вже два різні ключі. Зверніть увагу, що генерація ключів вимагає використання безпечних і надійних методів. Деякі мови програмування та криптографічні бібліотеки надають спеціалізовані функції для генерації ключів. Рекомендується використовувати такі інструменти та дотримуватися кращих практик безпеки, які встановлені для вашої конкретної платформи та середовища. Після вибору алгоритму та генерації ключа можна розпочати процес шифрування. Вихідні дані (персональні дані) проходять через шифрувальний алгоритм за допомогою ключа, що перетворює їх у шифротекст, а далі зашифровані данні передаються чи зберігаються. Шифрування може включати послідовне застосування різних операцій, таких як перестановки, заміни символів, бітові операції, повторення або чергування одних і тих же операцій одна за одною, тощо, для забезпечення максимальної безпеки даних. Після шифрування дані можуть бути збережені на носії інформації або передані по мережі. Зберігання або передача шифрованих даних забезпечує їх конфіденційність, оскільки без правильного ключа неможливо розшифрувати дані і отримати зрозумілу

інформацію, що дає можливість безпечно передавати дані. Для отримання зрозумілих даних з шифротексту потрібно виконати процес розшифрування. Цей процес включає застосування шифрувального алгоритму до шифротексту з використанням відповідного ключа. Якщо ключ правильний, шифротекст перетворюється у вихідні персональні дані (рис. 2.1). [4].

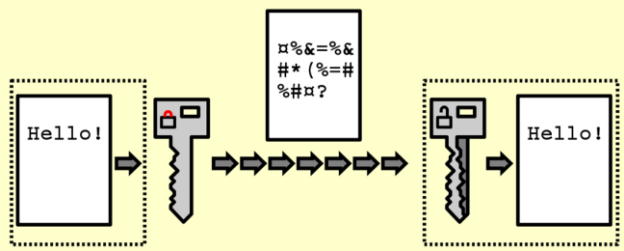

Рис. 2.1. Ілюстрація шифрування повідомлення

2.1.1 Аналіз алгоритмів шифрування

Advanced Encryption Standard (AES), також відомий, як *Rijndael* - симетричний алгоритм блокового шифрування. Нижче перелічені основні характеристи *AES:*

– Симетричний шифр *AES* використовує один ключ для як шифрування, так і розшифрування даних. Це означає, що той самий ключ використовується для зашифрування та розшифрування повідомлень.

– Розмір ключа *AES* підтримує ключі різної довжини: 128, 192 та 256 біт. Більша довжина ключа вважається більш безпечною, оскільки забезпечує більшу стійкість до криптоаналізу.

– Блочний шифр *AES* працює з блоками даних фіксованого розміру (128 біт), розбиваючи повідомлення на блоки перед шифруванням. Досліджуючи головні переваги алгоритму *AES,* можна виділити наступне:

– Надійність алгоритму *AES* базується на математичних особливостях, які здатністю стійко витримувати криптоаналіз. Зараз немає ефективних атак на *AES,* якщо використовується достатньо довгий ключ.

– Ефективність *AES* вважається ефективним та ресурсозаощадливим алгоритмом і може бути реалізований на різних пристроях, включаючи мобільні телефони, комп'ютери, мікроконтролери тощо.

– Підтримка *AES* є стандартом в багатьох сучасних криптографічних бібліотеках і програмних продуктах. Він широко підтримується операційними системами і програмними засобами, що дозволяє легко застосовувати його для шифрування та розшифрування даних. Також цей алгоритм піддається періодичним оновленням і переглядам для збереження високого рівня безпеки в майбутньому.

– Простота використання *AES* має простий інтерфейс, який дозволяє легко виконувати операції шифрування та розшифрування без спеціальних навичок або глибоких знань криптографії.

До головних недоліків алгоритму *AES* можна віднести симетричність ключа, тому що це вимагає постійного обміну ключами, що у свою чергу додає проблем у забезпечені безпечного його обміну. Також ключ треба якось зберігати, що ставить додаткові задачі для його безпечного збереження і в додачу, ключ треба час від часу змінювати.

Нижче наведений опис процесу шифрування *AES*:

Крок 1. Вибір ключа. Вибирається випадковий ключ, який буде використовуватися для шифрування та розшифрування даних. Розмір ключа може бути 128 біт, 192 біта або 256 бітів, в залежності від рівня безпеки, який ви бажаєте.

Крок 2. Підготовка даних. Розбиваються вхідні дані на блоки фіксованого розміру (наприклад, 128 біт).

Крок 3. Додавання ключа. Кожен блок даних комбінується з ключем за допомогою операції *XOR* (ексклюзивного “або”). Ця операція називається “*AddRoundKey*”.

Крок 4. Перестановка байтів. Байти в кожному блоку даних переставляються за певними правилами, що називається “*SubBytes*”.

Крок 5. Зсув байтів. Кожен рядок в кожному блоку даних зсувається вліво на певну кількість позицій, називається “*ShiftRows*”.

Крок 6. Змішування стовпців. Стовпці в кожному блоку даних змішуються між собою, називається “*MixColumns*”.

Крок 7. Додавання ключа (другий раз). Ключ додається до кожного блоку даних знову за допомогою операції *XOR*.

Крок 8. Повторення раундів. Кроки 4-7 повторюються для заданої кількості раундів (10 раундів для 128-бітного ключа, 12 раундів для 192-бітного ключа і 14 раундів для 256-бітного ключа).

Крок 9. Фінальне шифрування. Після виконання всіх раундів, дані додатково переставляються та додаються ключі, як у попередніх кроках.

Крок 10. Вивід шифртексту. Зашифровані блоки даних об'єднуються для утворення шифртексту.

Розшифрування виконується в зворотньому порядку, з використанням того самого ключа. *AES* є дуже надійним алгоритмом і залишається стандартом шифрування в багатьох додатках та системах для забезпечення конфіденційності даних [5].

Приведемо приклад шифрування слова "*HELLO*" за допомогою *AES-128* (128-бітний ключ):

1.Візьмемо 128-бітний ключ для шифрування, наприклад, *"SECRETKEY123456"*. Цей ключ повинен бути відомий як відправнику, так і одержувачу.

2. Перетворення слова у байтовий масив:

"H" кодується у байт 72.

"E" кодується у байт 69.

"L" кодується у байт 76.

"L" кодується у байт 76.

"O" кодується у байт 79.

Масив байтів для слова *"HELLO"* виглядає так: [72, 69, 76, 76, 79].

3. Підготовка ключа.

Ключ *"SECRETKEY123456"* також перетворюється в байтовий масив відповідно до *ASCII* кодування або іншого методу конвертації.

Виконання перших кроків алгоритму *AES*:

– Додавання ключа ("*AddRoundKey*") до початкового блоку даних.

– Зсув байтів в рядках (“*ShiftRows*”).

– Змішування стовпців (“*MixColumns*”).

– Ці операції виконуються над початковим блоком даних.

– Повторення раундів:

– Перші три операції (*AddRoundKey*, *ShiftRows*, *MixColumns*) повторюються для заданої кількості раундів, яка залежить від довжини ключа (10 раундів для *AES*-128). Кожен раунд включає в себе операції з ключем та над даними.

Завершення шифрування:

– Додавання ключа *(“AddRoundKey”)* в останньому раунді.

– Отримання шифрованого тексту.

Шифрований текст представляє собою байтовий масив, який ви можете передавати. Для розшифрування, одержувач повинен використовувати відповідний ключ *“SECRETKEY123456”,* що і відправник та розшифрувати шифрований текст за допомогою обернених операцій *AES*.

AES є дуже надійним алгоритмом і залишається стандартом шифрування в багатьох додатках та системах для забезпечення конфіденційності даних.

RSA (Rivest-Shamir-Adleman) - це асиметричний криптографічний алгоритм, який використовується для шифрування та цифрового підпису. Цей алгоритм базується на проблемі факторизації великих простих чисел, яка є важкою для обчислень у випадку великих чисел.

У свою чергу, алгоритм *RSA* відрізняється від *AES* асиметричним шифруванням і використанням асиметричного ключа. Алгоритм *RSA* має пару ключів - публічний та приватний. Публічний ключ використовується для шифрування даних, тоді як приватний ключ використовується для розшифрування даних. Ця двохкомпонентна система дозволяє надійно обмінюватися даними між сторонами, не обмінюючи приватний ключ. Повідомлення шифрується публічним ключем, а розшифрується приватним ключем. Цей алгоритм базується на проблемі факторизації великих простих чисел, яка є важкою для обчислень у випадку великих чисел. Це забезпечує конфіденційність даних, оскільки тільки власник приватного ключа може

розшифрувати дані, зашифровані його публічним ключем. Також головною відмінністю *RSA* використовується для створення цифрових підписів, що дозволяє перевіряти автентичність документів або повідомлень. Підпис створюється за допомогою приватного ключа і перевіряється за допомогою публічного ключа, забезпечуючи цілісність та автентичність даних.

До позитивних рис алгоритму *RSA* можна віднести його безпеку, яка базується на складності факторизації великих простих чисел. Зараз немає ефективних алгоритмів для швидкого розкладання великих чисел на їх прості множники. *RSA* дозволяє безпечний обмін ключами між сторонами без необхідності передачі самого ключа. Також його можливість створення цифрових підписів, що дозволяє перевіряти цілісність та автентичність документів та повідомлень. А для адаптивності під всі системи з різними характеристиками потужності *RSA* може використовувати ключі різних розмірів для різних потреб безпеки. Більші розміри ключів забезпечують більшу стійкість, але можуть вимагати більших обчислювальних ресурсів.

До негативних рис алгоритму *RSA* можна віднести його високу обчислювальну складність при роботі з великими числами, особливо при генерації ключів та шифруванні, розшифруванні великих об'ємів даних. Також для забезпечення надійної безпеки, необхідно використовувати довгі ключі, що може призвести до значного збільшення обчислювальних витрат при роботі з даними. Великий розмір ключів може спричинити уповільнення виконання операцій шифрування та розшифрування. Більші розміри ключів забезпечують більшу стійкість, але можуть вимагати більших обчислювальних ресурсів. Майбутні квантові комп'ютери можуть стати загрозою для безпеки *RSA*, оскільки вони можуть швидко розкладати великі числа на множники, що потенційно порушує складність алгоритму *RSA*.

Також відзначимо, що цей алгоритм використовується у веб-браузерах для шифрування трафіку між веб-серверами і клієнтами за допомогою *SSL (Secure Sockets Layer)* або *TLS (Transport Layer Security)*. Це забезпечує безпечний обмін інформацією між веб-серверами та користувачами в Інтернеті.

Процес шифрування даних за допомогою алгоритму *RSA* включає наступні кроки:

Крок 1. Генерація ключів. Ключі *RSA* складаються з двох частин: публічного ключа та приватного ключа. Публічний ключ містить e (публічна експонента) та n (модуль). Приватний ключ містить *d* (приватна експонента) та *n*.

Крок 2. Вибір простих чисел. Виберіть два великих простих числа *p* і *q*. Обчисліть *n = p * q* — це буде модуль *RSA*.

Крок 3. Обчислення функції Ейлера *(Euler's Totient)*. Обчисліть функцію Ейлера*: φ(n) = (p - 1) * (q - 1).*

Крок 4. Вибір публічної експоненти. Виберіть ціле число e, яке є взаємно простим зі значенням функції Ейлера *φ(n),* тобто *gcd(e, φ(n)) = 1*, де *gcd* - найбільший спільний дільник.

Крок 5. Обчислення приватної експоненти. Знайдіть число *d* таке, *що (d * e) mod φ(n) = 1.*

Крок 6. Створення публічного та приватного ключів. Публічний ключ - *{e, n}*. Приватний ключ - *{d, n}*.

Крок 7. Шифрування даних. Конвертуйте повідомлення *M* у числове представлення (зазвичай це число на основі його байтового вмісту). Зашифруйте числове представлення *M* за допомогою публічного ключа *{e, n}*. Використовуйте формулу: *C = M^e mod n*. Отримаємо *C* - зашифрований текст [6].

Приклад шифрування слово *“HELLO”* за допомогою *RSA*:

Для демонстрації використаємо малі числа для генерації публічного та приватного ключа.

1. Нехай *p = 61* та *q = 53*, обчислимо *n = p * q = 61 * 53 = 3233.*

2. Знайдемо функцію Ейлера*: φ(n) = (p - 1) * (q - 1) = 60 * 52 = 3120*.

3. Оберемо *e = 17* (для прикладу, як просте число, взаємно просте з *φ(n))*.

4. Знайдемо *d*, таке, що *(d * e) mod φ(n) = 1*. У цьому прикладі, можна встановити *d = 2753.* Таким чином, у нас є публічний ключ *{e = 17, n = 3233}* та приватний ключ *{d = 2753, n = 3233}*.

5. Шифруючи слово *“HELLO”*, перетворимо кожну літеру у *ASCII* код та об'єднаємо їх*: “H” = 72, “E” = 69, “L” = 76, “L” = 76, “O” = 79*. Об'єднані значення: 7269767679. Шифрування: *C = M^e mod n. M* (повідомлення) = 7269767679. *C* (зашифроване повідомлення) = *7269767679^17 mod 3233*.

Отже, шифроване повідомлення для слова *“HELLO”* буде результатом обчислення *C*. На жаль, розрахунок використовуючи великі числа не є практично можливим в цьому контексті, оскільки це потребує складних обчислень з великими числами. У реальності для *RSA* використовуються набагато більші значення для забезпечення високого рівня безпеки.

Алгоритм *DES* (*Data Encryption Standard*) може вважатися одним з популярних алгоритмів шифрування. Це симетричний блочний шифр, який був розроблений у 1970-х роках. Однак через зростання обчислювальної потужності комп'ютерів, яка зробила *DES* вразливим до атак *brute force*, його вважають застарілим з точки зору безпеки. *DES* використовував ключ довжиною 56 біт, що робило його вразливим до перебору ключів [7].

3DES (*3 Data Encryption Standard*) — це варіант *DES*, який використовується для підвищення безпеки, повторно застосовуючи *DES* з трьома ключами. Це дозволяє підвищити безпеку до певної міри, але *3DES* також вважається менш ефективним порівняно з більш сучасними алгоритмами шифрування через його обмежену швидкодію.

Основні характеристики *3DES*:

– Ключі 3*DES* використовує три ключі *DES* для шифрування даних. Кожен з цих ключів має довжину 56 бітів, що загалом дає 168 біт для шифрування та розшифрування.

– Блочний шифр так само як і оригінальний *DES, 3DES* шифрує дані по блоках фіксованого розміру. Зазвичай, розмір блоку у *3DES* - 64 біти.

– Режими роботи 3DES підтримує різні режими роботи, такі як *ECB* (*Electronic Codebook*), *CBC* (*Cipher Block Chaining*), *CFB* (*Cipher Feedback*), або *OFB* (*Output Feedback*), що впливає на те, як блоки даних обробляються та шифруються.

– Тричастинне шифрування: у *3DES* кожен блок даних проходить через три послідовні операції шифрування *DES* з трьома ключами.

Це може бути виконано за допомогою одного з наступних методів:

– *EDE (Encrypt, Decrypt, Encrypt)* - шифрування, розшифрування, шифрування.

– *EEE (Encrypt, Encrypt, Encrypt)* - потрійне шифрування без розшифрування.

Переваги алгоритму *3DEC*:

– Надійність алгоритму є високою завдяки використанню трьох ключів та потрійному застосуванню *DES, 3DES* збільшує безпеку в порівнянні з оригінальним *DES*. Це дозволяє залишати дані більш захищеними від атак.

– *3DES* може бути легко реалізований на основі існуючих систем та апаратних засобів, які вже підтримують *DES*, що робить його зручним для використання без потреби у великих змінах.

Недоліки алгоритму *3DEC*:

– Одним з основних недоліків *3DES* є його швидкодія. У порівнянні з більш сучасними алгоритмами шифрування, такими як *AES, 3DES* може бути помітно повільнішим через потребу виконання трьох раундів шифрування.

– *3DES* використовує 56-бітні ключі *DES*, що може бути вразливим до атак *brute force*. В умовах швидкого розвитку обчислювальної потужності, ця довжина ключа може бути недостатньою для надійного захисту даних.

– У зв'язку з потребою в трьох раундах шифрування, *3DES* вимагає більшу кількість обчислювальних ресурсів, що може бути неефективним у вимогливих до продуктивності системах.

– Одним з важливих моментів є те, що на сьогоднішній день рекомендується використовувати більш сучасні алгоритми шифрування з більшими довжинами ключів, такі як *AES*, замість *3DES*, через його обмежену безпеку та швидкодію [7].

Blowfish і *Twofish* - це два алгоритми блочного шифрування, розроблені для захисту конфіденційності даних. Обидва алгоритми були створені Брюсом

Шнайером. Вони відрізняються своєю структурою та характеристиками, і кожен з них має свої власні переваги та недоліки.

Blowfish був представлений в 1993 році і є алгоритмом симетричного блочного шифрування з фіксованим блоком даних розміром 64 біта. *Blowfish* призначений для використання з ключами довжиною від 32 до 448 біт. Алгоритм складається з трьох основних кроків: ініціалізації, шифрування та дешифрування. Він використовує *Feistel Network* - підхід, де блок даних розділяється на дві половини, які обробляються в змішаній послідовності. Це робить його дуже гнучким для різних застосувань. Характеризується високою швидкістю, що робить його ефективним для реалізації у різних системах.

Twofish був представлений в 1998 році як потенційна альтернатива *DES* та іншим алгоритмам. Це теж симетричний блочний шифр, але з більшим блоком даних розміром 128 біт. *Twofish* використовує ключі довжиною від 128 до 256 біт, що забезпечує достатню безпеку для багатьох застосувань. Ключі зазвичай розраховуються з вхідного паролю за допомогою функції хешування та процедури *Key Schedule*. Цей алгоритм базується на *Feistel Network* - це криптографічний підхід, де вхідний блок даних поділяється на половини, а потім проходить через серію раундів, де вони по черзі обробляються шифруючою функцією. *Twofish* використовує 16 раундів з використанням ключів раундів, які генеруються на основі вхідного ключа шифрування в процедурі *Key Schedule.*

Алгоритми *Blowfish* та *Twofish* володіють високими рівнями безпеки та легкою реалізацією, але через деякі патентні обмеження не так поширені, як, наприклад, *AES*.

Обидва алгоритми забезпечують надійний рівень безпеки, але вибір між ними може залежати від конкретних потреб і обмежень застосувань. У випадку нових застосувань рекомендується використовувати *Twofish* або інші сучасні алгоритми, оскільки *Blowfish* вважається менш безпечним через ряд відомих атак.

Реалізація алгоритму *Blowfish* починається з ініціалізації ключа. Генерується та ініціалізується подовжений ключ (*P-Array* та *S-Box*), який використовується для шифрування. Потім текст розбивається на блоки

фіксованого розміру (зазвичай 64 біта) і починаються етапи шифрування з першого проходу блоками через змішувальну мережу (*Feistel Network*) для створення підключів. Повторення серії раундів, кожен з яких включає в себе підключі та ітерації змішувальної мережі. Якщо потрібно розшифрувати дані, застосовується процес, обернений до шифрування. Остаточна обробка результату для отримання зашифрованого тексту.

З переваг алгоритму *Blowfish* можна виділити його швидкість, підтримку різних розмірів ключів, стійкість до криптоаналізу та простоту реалізації.

З недоліків алгоритму *Blowfish* можна виділити розмір блоку даних, особливо при обробці великих обсягів даних. Це може викликати певні проблеми з безпеці, оскільки шифрування однакових блоків даних може призвести до певних вразливостей. Також після певної кількості шифрувань з однаковим ключем є ймовірності появи вразливості у шифруванні.

Реалізація алгоритму *Twofish* починається з також з генерації підключів. Генерується розширений ключ шифрування за допомогою процедури *Key Schedule*, яка створює 40 ключів раундів по 32 біта кожен з основного ключа. Потім розбивається текст на блоки розміром у 128 біта. Кожен блок даних пройде через серію 16 раундів. Кожен раунд включає в себе декілька операцій, таких як змішування байтів, лінійні перетворення, зміщення та *XOR*-операції з ключами раундів. Потім використання мережі Фейстеля для додаткового розгортання ключа та дешифрування даних. Після завершення 16 раундів дані піддаються останньому лінійному перетворенню, яке завершує процес шифрування. Якщо потрібно розшифрувати дані, застосовується оборотний процес.

З позитивним рис алгоритму *Twofish* можна відмітити його високу безпечу. Цей алгоритм достатньо поширений і популярний, тому проблем з його вивченням і реалізацією не буде. Він чудово оптимізовний, тому може ефективно працювати на різних пристроях та платформах. Через симетричний шифр, операції шифрування та розшифрування проводить достатньо швидко.

З негативних рис алгоритму *Twofish* можна відмітити великі потреби у обсязі пам’яті для Для забезпечення ефективної роботи та зберігання проміжних

результатів, *Twofish* може потребувати більший обсяг пам'яті порівняно з деякими іншими шифрами. Потребує великої обчислювальної потужності, може потребувати більшої обчислювальної потужності порівняно з іншими алгоритмами, що може бути проблемою на менш потужних пристроях. Його реалізація може бути складнішою порівняно з деякими іншими шифрами через його складну структуру та багатоетапні операції. Великий розмір ключа, що може зробити його менш зручним для деяких використань, де важливо використовувати менші ключі.

Обидва алгоритми мають спільний підхід через мережу Фейстеля, але деталі реалізації, розмір блоків та інші характеристики відрізняються. У цілому, обидва алгоритми використовують схожий принцип роботи блочного шифрування зі змішувальною мережею.

Elliptic Curve Cryptography (ECC) - це сучасний підхід до криптографії, який базується на використанні властивостей еліптичних кривих для створення криптографічних систем. *ECC* використовує математичні властивості еліптичних кривих для забезпечення безпеки обміну даними та створення цифрових підписів.

Еліптичні криві - це геометричні об'єкти, що задаються рівняннями виду $y^2 = x^3 + ax + \mathrm{b}$, де x та y - координати точок на кривій, в свою чергу a та b – параметри кривої. Ці криві мають ряд унікальних математичних властивостей, що використовуються в криптографії.

Ключі алгоритму *ECC* представляють собою точки на еліптичній кривій, а не великі числа, як у *RSA*. Пара ключів складається з приватного ключа (точка на кривій) та відповідного публічного ключа (інша точка на кривій). Слід відмітити, що неправильно згенеровані параметри кривої можуть стати об'єктом криптоаналізу. Наприклад, якщо параметри кривої обрані не вірно, це може призвести до знайдення дискретного логарифму, що порушить безпеку системи.

Алгоритм *ECC* шифрування надає можливість створення підпису для створення криптографічних систем, таких як шифрування, підпис та обмін ключами, використовуючи математичні операції на еліптичних кривих.

Наприклад, *ECC* може бути використаний для створення цифрових підписів та безпечних протоколів обміну ключами.

Використання в мобільних пристроях та *IoT*: *ECC* особливо популярний в застосунках, де обмежені обчислювальні ресурси, таких як мобільні телефони, *IoT*-пристрої тощо, через його властивості ефективності і малих розмірів ключів.

Однією з головних переваг *ECC* є його висока стійкість до криптоаналізу при використанні менших ключів порівняно з іншими симетричними або асиметричними алгоритмами. Це дозволяє забезпечити однаковий рівень безпеки з меншими обчислювальними ресурсами, що у свою чергу впливає на швидкість. *ECC* особливо популярний в застосунках, де обмежені обчислювальні ресурси, таких як мобільні телефони, *IoT*-пристрої тощо, через його властивості ефективності і малих розмірів ключів. Використовуючи менші за обсягом ключі, алгоритм *ECC* надає кращий захист ніж *RSA*, з більшим ключем

Реалізація *ECC* складніша порівняно з деякими іншими алгоритмами криптографії, наприклад, *RSA*. Це може призвести до помилок в імплементації, які можуть збільшити ризик уразливості або знизити безпеку системи. Для безпечного використання *ECC* важливо обрати відповідну еліптичну криву та параметри. Некоректний вибір параметрів кривої може призвести до вразливості системи на атаки, що у свою чергу може призвести до негативних наслідків для користувачів цієї системи. Для безпеки *ECC* важливо мати надійний та випадковий генератор чисел. Якщо генератор чисел є слабким або піддатним до атак, це може знизити стійкість криптосистеми та зменшити її безпечність шляхом знаходження алгоритмів генерації випадкових чисел і це надає зловмисникам можливість вручну віднайти випадково згенеровані значення для ключа. Хоча *ECC* вважається стійким до багатьох сучасних криптоаналітичних атак, квантові комп'ютери мають потенціал зламати певні криптографічні схеми, включаючи ті, які базуються на *Elliptic Curve Cryptography*. Наразі важливо визначити та застосувати правильні стандарти та протоколи для використання *ECC*. Проблеми стандартизації можуть стати перешкодою для вирішення питань безпеки та впровадження цієї технології в різні системи.

Процес шифрування за допомогою алгоритму *Elliptic Curve Cryptography* наступний:

Крок 1. Вибір параметрів кривої. Для *ECC* важливо вибрати криву з відповідними параметрами (наприклад, *a, b* для рівняння еліптичної кривої, модуль *p*, параметр порядку точки, базова точка кривої та інші параметри). Для прикладу, використаємо стандартну криву *NIST P-256*.

Крок 2. Генерація ключів. Створення публічного та приватного ключів на основі обраної еліптичної кривої. Приватний ключ - випадкове число (наприклад, *k*). Публічний ключ - точка на еліптичній криві, що обчислюється як *k* множене на базову точку кривої.

Крок 3. Шифрування повідомлення. Процес шифрування в *ECC* може використовувати протоколи обміну ключами, підписи або інші методи забезпечення конфіденційності. У *ECC*, зазвичай, шифрування використовується для підпису або обміну ключами, а не для зашифрування текстового повідомлення як у *RSA*.

Крок 4. Публічний ключ, який був використаний для шифрування або підпису, використовується для перевірки підпису або розшифрування даних (залежно від того, який метод був використаний).

На жаль, процес шифрування *"HELLO"* за допомогою *ECC* прямо у формі зашифрованого тексту не є типовим застосуванням *ECC*. Щоб продемонструвати шифрування повідомлення, треба використовувати ключові обміни, підписи або інші протоколи, які базуються на принципах *ECC*.

Зауважте, *ECC* використовується для створення безпечних протоколів обміну ключами та підписів, оскільки вона забезпечує високий рівень безпеки при використанні коротших ключів порівняно з іншими криптографічними алгоритмами, такими як *RSA* [8].

Проаналізувавши п'ять алгоритмів шифрування, можна зробити висновок, що найкращим для реалізації у системі дистанційного спілкування на базі операційної системи "*Android*" буде алгоритм *AES* для захисту інформації у системі спілкування.

Таблиця. 2.1

Порівняння алгоритмів захисту інформації у системах спілкування

	AES	*RSA*	*ECC*	*3DEC*	*Twofish*
Актуальність	+	+	+	-	+
Простота у реалізації	+	+	-	+	+
Достатній рівень захисту	+	+	+	-	+
Швидкість виконання	+	-	+	-	-
Підтримка моб. платформ	+	+	+	+	-
Ресурсовитратність	+	-	+	-	-

Отже, проаналізувавши п’ять алгоритмів шифрування і склавши порівняльну таблицю, оберемо єдиний, котрий буде використовуватися у розробці захисту інформації у системі дистанційного спілкування. Порівнюватися алгоритми будуть за наступними критеріями: актуальність, простота у реалізації, достатній рівень захисту, швидкість, підтримка мобільних платформ, ресурсовитратність. Де “+” - відповідає запланованим цілям, а “–”- не відповідає запланованим цілям.

2.1.2. Цифровий підпис та електронний цифровий ключ

Цифровий підпис - це електронна або криптографічна техніка, що використовується для підтвердження автентичності, цілісності та незмінності електронного повідомлення, документа або даних (рис. 2.2).

Рис. 2.2. Ілюстрація цифрового підпису

Цифровий підпис створюється за допомогою приватного ключа підписувача і перевіряється за допомогою відповідного публічного ключа.

Процес створення цифрового підпису включає наступні кроки:

1) Створення хешу. Повідомлення або дані перетворюються на хеш-функцію, яка створює унікальний вихідний код фіксованої довжини.

2) Шифрування хешу. Приватний ключ підписувача використовується для шифрування хешу повідомлення.

3) Додавання підпису до повідомлення. Зашифрований хеш додається до повідомлення або документа, щоб утворити цифровий підпис.

При перевірці цифрового підпису одержувач розшифровує хеш за допомогою публічного ключа, пов'язаного з приватним ключем підписувача. Обчислює хеш повідомлення і порівнює його з розшифрованим хешем. Якщо хеші співпадають, це підтверджує автентичність та незмінність повідомлення.

Електронний цифровий ключ - це криптографічний ключ, який використовується для шифрування, розшифрування, цифрового підпису або інших операцій забезпечення безпеки в електронних системах. Він може бути використаний як приватний ключ або публічний ключ, залежно від конкретної схеми шифрування.

Приватний ключ – це секретний ключ, який використовується для шифрування, розшифрування або підпису повідомлень. Приватний ключ повинен бути добре захищений і відомий тільки власнику.

Публічний ключ це який розповсюджується та відкритий для загального доступу. Він використовується для розшифрування повідомлень, перевірки цифрових підписів або інших операцій, пов'язаних з безпекою. Публічний ключ походить від приватного ключа за допомогою криптографічних алгоритмів.

Електронний цифровий ключ використовується для забезпечення конфіденційності, цілісності інформації та автентичності даних в електронних комунікаціях. Він забезпечує захист від несанкціонованого доступу, змін або підробки даних, а також дозволяє ідентифікувати та перевіряти автентичність сторін, що взаємодіють між собою [9].

2.1.3. Захищені протоколи передачі даних

Захищені протоколи передачі даних - це протоколи, які забезпечують безпеку та конфіденційність під час обміну даними між комунікуючими сторонами. Вони використовують криптографію та інші механізми для захисту інформації від несанкціонованого доступу, змін або підробки. Ось декілька з таких протоколів:

– *SSL/TLS (Secure Sockets Layer/Transport Layer Security)* - це широко використовуваний протокол захищеної передачі даних через мережу. Він забезпечує шифрування, автентифікацію та цілісність даних між клієнтом та сервером. *SSL/TLS* використовує сертифікати для перевірки автентичності сервера та публічного ключа шифрування. Протокол забезпечує захист від атак типу “прослуховування” та “підробки” даних (рис. 2.3).

Рис. 2.3. Ілюстрація протоколу *SSL*

– *IPsec (Internet Protocol Security)* це набір протоколів для захищеної передачі даних на рівні мережевого протоколу. Він забезпечує шифрування даних, їх цілісність та автентифікацію пакетів даних, що передаються між мережевими вузлами. *IPsec* може використовуватися для встановлення віртуальних приватних мереж *(VPN)* та забезпечення безпеки комунікації між різними мережами, динамічно змінюючи *IP* користувача.

– *SSH (Secure Shell)* це протокол для захищеного термінального доступу та передачі файлів між вузлами мережі. Він забезпечує шифрування, автентифікацію та цілісність даних, переданих через мережу.

SSH застосовується для безпечного входу на віддалені сервери, виконання команд і передачі файлів забезпеченому способом (рис. 2.4).

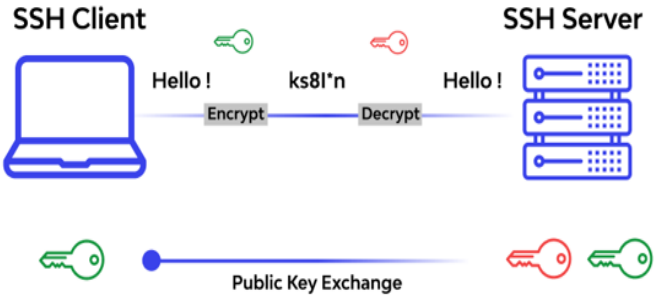

Рис. 2.4. Ілюстрація *SSH* протоколу

– *HTTPS (Hypertext Transfer Protocol Secure)* це захищений варіант протоколу *HTTP*, який використовує *SSL/TLS* для шифрування комунікації між веб-клієнтом та веб-сервером. Використання *HTTPS* забезпечує конфіденційність, цілісність та автентичність даних, переданих через мережу, що дозволяє захистити конфіденційну інформацію, таку як паролі та платежі (рис. 2.5).

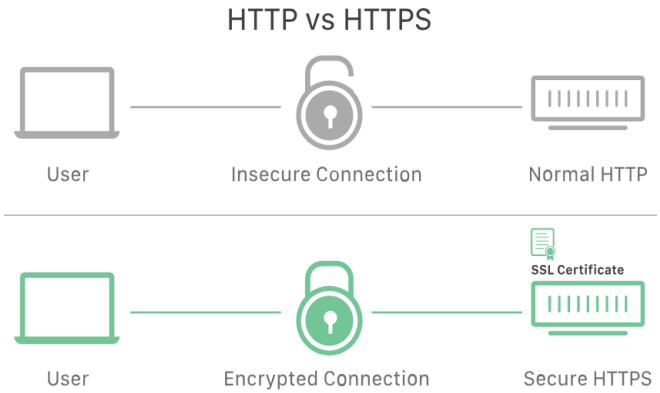

Рис. 2.5. Ілюстрація *HTTPS* протоколу

Ці протоколи використовуються для створення безпечного каналу передачі даних, що забезпечує захист від атак та забезпечує конфіденційність і цілісність інформації. Вибір конкретного протоколу залежить від ваших потреб та вимог проекту або застосування.

Вибір конкретного протоколу залежить від ваших потреб та вимог проекту або застосування.

Звичайному користувачеві слід обирати такі ресурси для обміну інформацією, котрі використовують один з вище перелічених протоколів передачі даних, не захищені – слід ігнорувати та оминати. Щоб визначити чи вебсайти використовують безпечні протоколи треба звернути увагу який протокол використовується у посиланні, якщо *http* - ненадійний, *https* - надійний. Також,

якщо ліворуч від адреси сайту ви побачите спеціальну позначку у вигляді замку вказує на те, що сайт використовує протокол SSL, якщо там знак оклику – з’єднання незахищене [10].

2.2. Захист мережевого сполучення та комунікації

2.2.1. Використання віртуальних приватних мереж *(VPN)*

Віртуальна приватна мережа *(VPN)* - це технологія, яка створює захищене з'єднання між віддаленими мережами або пристроями через незахищену мережу, таку як Інтернет.

Головні фактори для використання *VPN*:

1. *VPN* забезпечує конфіденційність даних. *VPN* використовує різні криптографічні протоколи для шифрування даних, що передаються між вузлами мережі. Це унеможливлює прослуховування або зловживання даними в транзиті, забезпечуючи конфіденційність інформації.

2. *VPN* може допомогти обійти обмеження та блокування які накладаються на доступ до певних веб-ресурсів або послуг.

3. *VPN* маскує вашу справжню *IP*-адресу і замість неї надає вам віртуальну *IP*-адресу, що приховує вашу ідентичність. Це допомагає зберегти вашу приватність та анонімність під час перегляду веб-сторінок і використання онлайн-сервісів.

4. *VPN* дозволяє вам безпечно підключатися до внутрішньої мережі організації або дому з будь-якого місця, що має доступ до Інтернету. Це особливо корисно для віддалених працівників, які потребують доступу до ресурсів інформаційної системи компанії.

5. Захист від загроз безпеки: використання *VPN* забезпечує захист від різних загроз безпеки, таких як вторгнення або атаки з боку зловмисників. За допомогою шифрування і автентифікації, *VPN* забезпечує безпеку вашого з'єднання та захищає вашу мережу.

Узагальнюючи, використання віртуальних приватних мереж *(VPN)* (рис. 2.6) дозволяє створити безпечне та приватне з'єднання між вузлами мережі через незахищену мережу, забезпечуючи конфіденційність, цілісність та анонімність даних. Це корисний інструмент для захисту вашої приватності та забезпечення безпеки під час передачі даних через Інтернет [11].

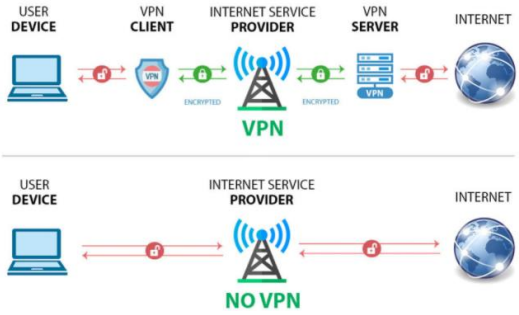

Рис. 2.6. Ілюстрація роботи *VPN*

2.2.2. Фірмові фаєрволи та інтрузійні системи виявлення

Фірмові фаєрволи та інтрузійні системи виявлення *(Intrusion Detection Systems, IDS)* є ключовими компонентами в системах безпеки мережі. Вони допомагають виявляти та захищати мережу від небажаних вторгнень, шкідливих програм та зловмисного поведінки. Інтрузійні системи виявлення допомагають виявляти вторгнення, шкідливі програми та інші загрози безпеці мережі. Ось докладніші пояснення про кожен з цих елементів:

– Фірмові файєрволи це пристрої або програми, які контролюють та фільтрують трафік, що проходить через мережевий вузол або точку доступу до мережі. Вони базуються на наборі правил, які визначають, які пакети дозволяються або блокуються на основі деяких характеристик, таких як *IP*-адреса, порти або протоколи. Якщо виявляється підозріла активність, IDS може сповістити адміністратора про потенційну загрозу або навіть прийняти заходи для блокування зловмисної діяльності.Фірмові фаєрволи допомагають захищати мережу від несанкціонованого доступу та зловмисних атак, таких як *DDoS* або перехоплення пакетів. Ось перелік популярних фірмових фаєрволів: *Cisco ASA(Adaptive Security Appliance), Palo Alto Networks Next-Generation Firewall, Fortinet FortiGate: FortiGate.*

– Інтрузійні системи виявлення це програми або пристрої, які моніторять мережевий трафік та виявляють небажані або зловмисні активності. Вони аналізують пакети даних, порівнюючи їх зі знаннями про відомі атаки або підозрілі шаблони поведінки. Якщо виявляється підозріла активність, IDS може сповістити адміністратора про потенційну загрозу або навіть прийняти заходи для блокування зловмисної діяльності. Інтрузійні системи виявлення допомагають виявляти вторгнення, шкідливі програми та інші загрози безпеці мережі.

На ринку існує багато популярних фірмових фаєрволів з різними функціями та можливостями. Як приклади популярних фірмових фаєрволів можна навести наступні: *Cisco ASA, Palo Alto Networks Next-Generation Firewall, Fortinet FortiGate, Check Point Firewall.*

Cisco ASA (*Adaptive Security Appliance*) є одним з найпопулярніших фірмових фаєрволів. Він надає широкий спектр функцій, включаючи фільтрацію пакетів, *VPN*-з'єднання, інспекцію *SSL*, запобігання вторгненням та багато іншого.

Palo Alto Networks Next-Generation Firewall - цей фаєрвол пропонує передові функції захисту, включаючи виявлення загроз, контроль застосунків, захист від витоку даних та інтеграцію з іншими системами безпеки мережі. Може стати невід’ємним компонентом системи безпеки мережі.

Fortinet FortiGate - *FortiGate* є інтегрованою платформою з багатьма функціями, яка включає фаєрвол, *VPN*, антивірусний сканер, інтрузійну систему виявлення та запобігання загрозам, захист від витоку даних та багато іншого. Що робить її актуальною у наш час.

Check Point Firewall - ця платформа фаєрволів надає рішення для захисту мережі від загроз, включаючи атаки зловмисників, витоки даних, віруси та шкідливі програми. Вона має розширені можливості керування та моніторингу трафіку, що при аналізу та певних навичках, надає можливість віднайти підозрілу шахрайську активність.

Ці дві компоненти - фаєрволи і інтрузійні системи виявлення - часто використовуються разом для створення повноцінної системи безпеки мережі. Фірмові фаєрволи забезпечують першу лінію оборони, контролюючи трафік і

блокуючи небажані з'єднання, а інтрузійні системи виявлення доповнюють це, надаючи аналіз і виявлення підозрілої або зловмисної активності в мережі. Разом вони допомагають забезпечити безпеку та захист мережі від широкого спектру загроз.

2.3. Захист автентифікації та авторизації

Захист автентифікації та авторизації є критичними аспектами в системах інформаційної безпеки. Ці процеси дозволяють перевіряти ідентичність користувачів та контролювати їх доступ до ресурсів системи. Задля забезпечення надійної автентифікація треба використовувати більш надійних методів, ніж прості паролі, таких як двоетапна перевірка, біометричні дані (відбитки пальців, розпізнавання обличчя) або використання апаратних ключів. Реалізовувати “принцип найменшого доступу” (least privilege), коли користувачам надаються тільки необхідні права доступу для виконання їхніх обов'язків. Це допомагає уникнути надання зайвих прав, які можуть бути зловживанні.

Корисною практикою також буде застосування шифрування для захисту важливих даних під час передачі через мережі, що дозволяє уникнути несанкціонованого доступу. Задля виявлення аномалій або спроб несанкціонованого доступу, бажано постійно вести журнал подій, аналіз цих журналів може допомогти виявити вразливі місця або несправності у захисту системи. Як додатковими методами можна користуватися обмеженням часу доступу. Встановивши обмеження на час доступу користувачів до системи або ресурсів. Це може запобігти можливості несанкціонованого доступу через довгий період не активності (рис. 2.7).

Рис. 2.7. Ілюстрація методів захисту автентифікації та авторизації

Використання сучасних систем, які дозволяють визначати, чи є дані відповідними і чи має особа/система потрібні права в реальному часі.

Забезпечення захисту автентифікації та авторизації - це надзвичайно важливе завдання у сфері кібербезпеки, оскільки це становить перший і ключовий рівень захисту доступу до конфіденційної інформації та ресурсів в будь-якій інформаційній системі.

2.4. Захист даних та конфіденційності

Зашифрування даних на рівні файлової системи використовується для забезпечення конфіденційності інформації, збереженої на зовнішньому носії, такому як жорсткий диск або флеш-накопичувач. Цей підхід дозволяє зашифрувати дані на рівні файлів або папок, що забезпечує їх захищений доступ.

Зашифрована файлова системи включає наступне:

– Дозволяє зберігати дані в зашифрованому вигляді. Це означає, що навіть якщо зловмисник отримає фізичний доступ до носія інформації, він не зможе прочитати або розібрати дані без правильного ключа шифрування.

– Забезпечує захист від випадкової втрати даних, наприклад, внаслідок втрати або крадіжки носія інформації. Без правильного ключа шифрування неможливо отримати доступ до зашифрованих даних, що робить їх некорисними для незаконних користувачів.

– Зашифрована файлова система дозволяє контролювати доступ до даних шляхом вимагання правильного ключа шифрування. Тільки користувачі, які мають відповідний ключ, зможуть розшифрувати та отримати доступ до даних.

Деякі популярні зашифровані файлові системи включають:

– *BitLocker* розроблена компанією *Microsoft, BitLocker* є вбудованим інструментом шифрування для операційних систем Windows. Вона надає можливість шифрувати дані на рівні файлової системи та підтримує різні режими шифрування, включаючи шифрування всього диска або віртуальних дисків.

– *FileVault* це зашифрована файлова система, розроблена компанією *Apple* для операційних систем *macOS*. *FileVault* надає можливість шифрувати весь внутрішній жорсткий диск або специфічні папки та файли.

– *VeraCrypt* це безкоштовний інструмент для шифрування даних, який підтримує різні операційні системи, включаючи Windows, *macOS* і *Linux*. *VeraCrypt* дозволяє створювати зашифровані контейнери або шифрувати цілі диски або розділи [12].

Захищене сховище даних, відоме також як “сховище для секретів” або “сховище паролів”, є механізмом, який дозволяє зберігати та управляти конфіденційною інформацією, такою як паролі, ключі шифрування, сертифікати та інші секрети. Його ціль полягає в тому, щоб забезпечити безпеку цих секретів шляхом їх захищеного зберігання та контролю доступу до них. У захищеному сховище використовуються такі методи як:

– Захищене сховище даних забезпечує механізми контролю доступу, які дозволяють обмежувати, хто має право переглядати або змінювати збережені секрети. Це може включати автентифікацію, авторизацію, рівні доступу та інші механізми безпеки.

– Шифрує данні, збережені в захищеному сховищі, зазвичай шифруються для забезпечення конфіденційності. Це означає, що навіть якщо зловмисник отримає фізичний доступ до сховища, він не зможе прочитати збережені дані без правильного ключа шифрування.

– Сховища даних зазвичай надають можливість резервного копіювання та відновлення даних. Це дозволяє уникнути втрати секретів у разі випадкової втрати або пошкодження сховища.

– Функції аудиту та журналювання реєструють всі дії, пов'язані з доступом до конфіденційної інформації. Це дозволяє виявляти незвичайну або підозрілу активність та вживати відповідних заходів безпеки. Це буде корисним у випадку, коли до вашого пристрою підключиться зловмисник і віддалено буде ним керувати. Навіть якщо він вирішить видалити всі сліди своєї активності, то воно все одно збережеться у журналюванні.

Прикладом популярних захищених сховищ даних включають можуть бути *KeePass* та *LastPass* .

KeePass - це безкоштовною і відкритою програмою для керування паролями. Вона призначена для збереження та організації паролів у безпечному місці. KeePass дозволяє користувачам створювати безпечні паролі, зберігати їх у зашифрованій базі даних та доступатися до них за допомогою одного основного пароля або ключового файлу. Він працює на різних платформах і має функції шифрування та захищеного доступу.

LastPass - це онлайн-сервіс зберігання паролів, який пропонує захищене сховище для паролів та автоматичне заповнення форм. Він також надає можливість синхронізувати дані між різними пристроями та має додаткові функції безпеки.

1Password - це ще один популярний сервіс зберігання паролів та секретів, який пропонує шифрування та захищений доступ до даних. Він підтримує різні платформи та має додаткові функції, такі як автоматичне заповнення форм, автоматичний вхід, генератор паролів та обмін паролями з іншими користувачами. Захист паролів створений завдяки сильному шифруванню, також користувачі можуть використовувати майстер – пароль та відбиток пальця.

2.5. Планування та управління інцидентами безпеки

Планування та управління інцидентами безпеки є важливою частиною стратегії безпеки і допомагає користувачам та організаціям ефективно реагувати на інциденти та забезпечувати відновлення і захист систем та даних.

Етапи планування та управління інцидентами безпеки включають наступні кроки:

1. Підготовка. Імплементування та налагодження необхідних методів та алгоритмів для захисту потрібної інформації і оцінювання ризиків негативного впливу.

2. Виявлення. Отримання звіту, або отримання інформації шляхом спостереження, негативної або аномальної поведінки системи, зміни даних або їх підстановка. Розробка механізмів та інструментів і / або проведення аудиту та спостереження за змінами у поведінці системи та стану конфіденційної інформації та паролів.

3. Оцінка. Аналізування ймовірного негативного впливу, або можливих негативних дій у майбутньому, отриманої шкоди, її наслідків та впливу інциденту на поведінку системи, її безпеку та на конфіденційну інформацію та паролі.

4. Реагування. Розробка та втілення планів протидії інциденту, включаючи процедури відновлення контролю, втрачених або викрадених даних.

5. Аналіз та вдосконалення. Після успішного усунення наслідків інциденту треба проаналізувати та виявити слабкі місця і вжити всіх необхідних заходів задля упередження подібних інцидентів у майбутньому, шляхом модернізації та оновлення.

Це загальний підхід до планування та управління інцидентами безпеки (рис. 2.8). Кожна організація може розробити власний план, враховуючи свої особливості, ризики та ресурси [13].

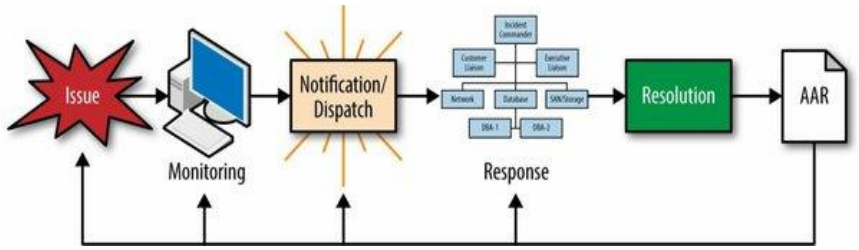

Рис. 2.8. Ілюстрація основних етапів планування та управліннями інцидентами

2.6. Використання на практиці методу автентифікації зі сховищ даних

Задля забезпечення ідентичності користувачів, обмеження доступу одних користувачів до приватних даних інших та контролю їх доступу до ресурсів системи найкращим, еталонним та обов’язковим методом є проходження користувачем автентифікації та авторизації.

Якщо потрібна система або ресурс надає можливість активізувати двохфакторну автентифікацію за допомогою мобільного номеру, електронної пошти чи коду доступу з побічного додатку, наприклад з “*Google Authenticator”* , то її обов’язково треба активувати.

Двохфакторна автентифікація дозволяє користувачеві поліпшити свою безпеку та даних за рахунок генерації тимчасового випадкового паролю, котрий буде приходити на мобільний номер, електронну пошту, або генеруватися у сторонньому додатку.

Також, ще однією практичною рекомендацією є використання складних та великих паролів під час автентифікації, а ще й у кожній новій системи використовувати новий пароль. З плином часу, у користувача може назбиратись велика кількість різних паролів. Не для кожного буде зручно виписувати всі паролі на листочок, а зберігати їх в одному файлі текстового формату на комп’ютері або телефоні не є безпечним і надійним, адже можна втратити файл з паролями, або зловмисник отримає доступ до вашого списку паролів.

Отже, задля вирішення цієї проблеми і покращення методів автентифікації та авторизації буде використано захищене сховище даних, де кожен користувач зможе зберегти всі свої паролі, котрі також будуть додатково зашифровані та обмежені у доступі за допомогою паролю. Це все можна реалізувати за допомогою сховища “*KeePass”*.

При роботі зі сховищем “*KeePass”*, треба створити базу даних з майбутніми паролями, задля цього треба для бази даних створити пароль (рис. 2.9).

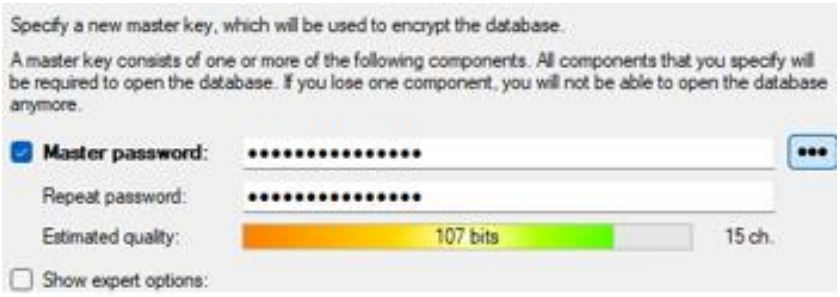

Рис. 2.9. Створення бази даних з майбутніми паролями у захищеному сховищі даних “*KeePass”*

Після пройти етап налаштування бази даних (рис. 2.10).

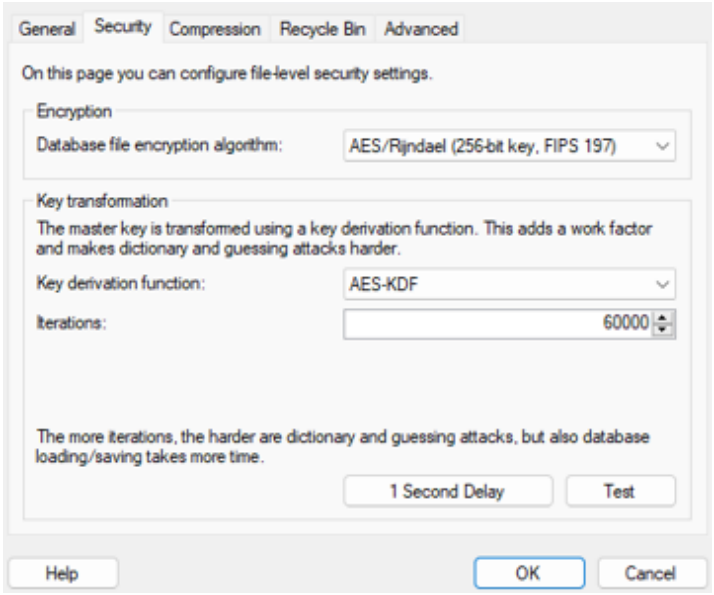

Рис. 2.10. Налаштування бази даних з паролями

Кожен користувач зможе за власним бажанням налаштувати сховище паролів та методи їх шифрування.

Після створення бази даних, користувач може розпочати її заповнення, зручно формуючи зміст та керувати тонкими налаштуваннями сховища даних. Створити нові данні для входу в будь-яку систему зі згенерованим паролем, налаштувати його шифрування та використати інші тонкі функції менеджменту пароля та бази даних.

Віконце створення нового паролю (рис. 2.11).

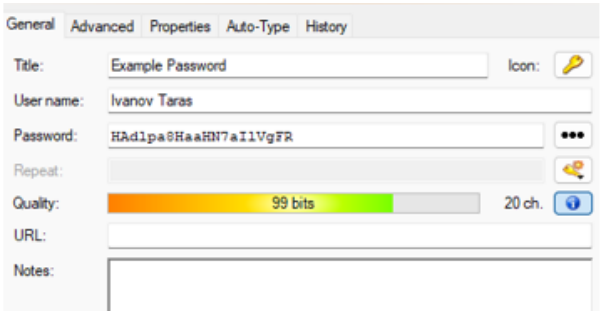

Рис. 2.11. Створення нового паролю

Отже, запропонований варіант, дозволяє поліпшити метод автентифікації користувачів у потрібних їм системах, забираючи на себе рутинну частину обов’язків та оптимізуючи процеси від створення до захисту паролю від зловмисників. Таким чином, всі паролі, створені у захищеному сховищі будуть

мати набагато більше шарів безпеки від зловмисників та гарантуватимуть надійний захист персональної та конфіденційної інформації користувачів.

Нижче наведено практичні рекомендації щодо захисту інформації в системі дистанційного спілкування:

– Використовуйте надійні паролі. Переконайтеся, що ваші паролі складаються з комбінації великих і малих літер, цифр та спеціальних символів. Використовуйте унікальні паролі для кожної онлайн-платформи або сервісу і регулярно змінюйте їх.

– Активуйте двоетапну перевірку. Використовуйте можливості двоетапної перевірки, такі як одноразові паролі або підтвердження через мобільний пристрій, для забезпечення додаткового рівня безпеки під час входу до системи дистанційного спілкування.

– Оновлюйте програмне забезпечення. Переконайтеся, що всі програми, включаючи програмне забезпечення для дистанційного спілкування, оновлені до останньої версії. Оновлення часто містять виправлення вразливостей та покращення безпеки.

– Захищений доступ до засобів спілкування. Використовуйте захищені канали зв'язку, такі як шифровані підключення *HTTPS*, для передачі інформації. Уникайте відкритих або незахищених *Wi-Fi* мереж, особливо при обміні конфіденційною інформацією.

– Керуйте правами доступу. Надайте доступ до системи дистанційного спілкування лише необхідним користувачам і встановіть права доступу відповідно до ролей і обов'язків користувачів. Регулярно переглядайте та відкликайте права доступу користувачів, які більше не потребують доступу.

– Уважність при посиланнях та вкладеннях. Уникайте небезпечних посилань та небезпечних вкладень, особливо від невідомих або ненадійних джерел. Будьте обережні при відкритті файлів, які надійшли вам електронною поштою або через повідомлення.

– Зберігайте резервні копії даних. Регулярно робіть резервні копії важливих даних, включаючи файли спілкування, і зберігайте їх на захищених носіях або в хмарних сховищах. Це допоможе уникнути втрати даних у разі інциденту.

– Освіта користувачів. Навчайтесь основам кібербезпеки та небезпекам, пов'язаним з дистанційним спілкуванням.

– Моніторинг та виявлення. Встановіть механізми моніторингу та виявлення вторгнень для системи дистанційного спілкування, які дозволять вам вчасно виявити незвичайну або підозрілу активність.

– Аудит безпеки. Регулярно проводьте аудит безпеки системи дистанційного спілкування, щоб ідентифікувати можливі вразливості та вжити заходів для їх вирішення [14].

Ці рекомендації допоможуть забезпечити безпеку і захист інформації в системі дистанційного спілкування. Проте, важливо пам'ятати, що безпека - постійний процес, і потрібно підтримувати свої заходи безпеки оновленими та адаптованими до нових загроз.

Висновки за розділом

Основні методи захисту інформації у системах дистанційного спілкування є необхідними для забезпечення безпеки та конфіденційності під час використання таких систем. Були розглянуті різні аспекти захисту інформації, починаючи з криптографічних методів шифрування, що дозволяють захистити дані від несанкціонованого доступу та перехоплення.

Принципи криптографії відіграє ключову роль у захисті конфіденційності, цілісності, автентифікації та незаперечності даних. Вона використовує ключі для шифрування та розшифрування даних та забезпечує відмовостійкість криптографічних алгоритмів.

Процес шифрування особистих даних: Перший крок - вибір шифрувального алгоритму. Існує багато алгоритмів, таких як *AES, RSA, DES*. Вибір залежить від безпеки, швидкодії та вимог системи. Потім генерується ключ для шифрування:

симетричний або асиметричний. Генерація ключа потребує безпечних методів, включаючи криптографічно стійкі генератори випадкових чисел. Для початку шифрування даних використовують обраний алгоритм та ключ, який перетворює особисті дані у шифротекст. Шифрування забезпечує конфіденційність даних під час зберігання чи передачі.

Процес розшифрування: Для отримання зрозумілих даних з шифротексту використовується відповідний ключ та шифрувальний алгоритм. Правильний ключ перетворює шифротекст у вихідні особисті дані.

Важливо враховувати, що криптографія вимагає високого рівня уваги до безпеки і обережності у виборі алгоритмів, генерації та зберігання ключів. Рекомендується використовувати найкращі практики безпеки та надійні інструменти для захисту особистих даних.

Із загальновідомих та популярних алгоритмів шифрування можна виділити наступні: *AES, RSA, ECC* та *3DES.*

Основна інформація по алгоритму *AES* - перевірений і надійний симетричний алгоритм шифрування з ключами 128, 192 та 256 бітів, який широко використовується для захисту конфіденційності даних. Переваги *AES*: швидкий, ефективний та стійкий до атак. доступні різні розміри ключів, висока швидкодія, простота реалізації та стійкість. Недоліки *AES* - можливість атак методами перебору на дуже довгому терміні, особливо для менших розмірів ключів.

Основна інформація по алгоритму *RSA* - це асиметричний алгоритм, який використовується для шифрування та цифрового підпису. Відомий своєю можливістю шифрування з використанням публічного та приватного ключа. Забезпечує захист інформації та ефективність в цифрових підписах, але може бути вимогливим щодо обчислювальних ресурсів.

Переваги *RSA* - підтримка різних розмірів ключів, створення цифрових підписів, безпечний обмін ключами.

Недоліки *RSA* - великі обчислювальні витрати для деяких операцій, особливо з великими ключами.

Основна інформація по алгоритму *ECC:* ще один асиметричний алгоритм, що базується на математичних властивостях еліптичних кривих. Забезпечує високий рівень безпеки з меншими обчислювальними витратами.

Переваги *ECC* - використовує менше ресурсів, ніж *RSA*, для того ж рівня безпеки. Висока безпека при менших обчислювальних витратах. .

Недоліки *ECC* - залежність від безпеки еліптичних кривих, яка може бути під загрозою при недостатньо вибраному параметризації.

Основна інформація по алгоритму *3DES:* старіший симетричний алгоритм, який використовується для забезпечення додаткової захисту шляхом потрійного шифрування з використанням ключів різної довжини Має добру стійкість, але у порівнянні з AES вважається менш безпечним через вік деяких його аспектів.

Переваги *3DES* - широко використовується в старіших системах, все ще залишається деякою мірою безпечним.

Недоліки *3DES* - повільніший через потребу у виконанні трьох раундів шифрування, ресурсозатратний та через обмежену довжину ключа може бути вразливим до атак *brute force.*

Також були розглянуті практичні аспекти захисту, такі як використання віртуальних приватних мереж *(VPN)* для створення захищеного з'єднання та шифрування даних під час їх передачі. Додаткові заходи безпеки, такі як використання фірмових фаєрволів, інтрузійних систем виявлення, а також забезпечення автентифікації та авторизації, вносять вагомий внесок у стійкість системи дистанційного спілкування.

Правильне планування та управління інцидентами безпеки, а також врахування вимог регуляторів, є важливими аспектами забезпечення безпеки і виконання вимог щодо конфіденційності персональних даних. Крім того, застосування захищених протоколів передачі даних та зашифрування на рівні файлової системи додатково підвищують рівень безпеки системи дистанційного спілкування.

Необхідно постійно оновлювати свої знання про останні загрози та розвиток технологій безпеки, щоб залишатись впевненими в безпеці своєї інформації та

забезпечувати надійну захист в системах дистанційного спілкування. Із зростанням важливості цифрової безпеки, необхідно постійно адаптувати та вдосконалювати методи захисту для запобігання новим загрозам і вразливостям.

Узагальнюючи, застосування основних методів захисту інформації в системах дистанційного спілкування є невід'ємною складовою безпечного та конфіденційного обміну даними. Забезпечення безпеки вимагає комплексного підходу, включаючи технічні, організаційні та людські аспекти, і постійного вдосконалення для адаптації до нових загроз та ризиків.

РОЗДІЛ 3
ЗАБЕЗПЕЧЕННЯ НАДІЙНОГО ЗАХИСТУ У СИСТЕМАХ ДИСТАНЦІЙНОГО СПІЛКУВАННЯ

3.1. Середовище розробки – *“Android Studio”*

“Android Studio”, заснована на програмному забезпеченні *IntelliJ IDEA* від *JetBrains*, офіційному інструменті для розробки додатків *“Android”*. Ця розробка середовища доступна для *Windows, OS X* і *Linux*.

Середовище розробки адаптовано для виконання типових завдань, які вирішуються в процесі розробки програм для платформи *“Android”.* У тому числі інструменти для спрощення тестування програм на сумісність з різними версіями платформи та інструменти розробки додатків, що працюють на пристроях з різною роздільною здатністю екранів (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в *IntelliJ IDEA*, в *“Android Studio*” реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема для складання, тестування та розгортання додатків на основі інструментів складання *Gradle* і підтримує використання засобів безперервної інтеграції.

Для прискорення розробки додатків представлена колекція стандартних елементів інтерфейсу та візуальний редактор для їх розміщення, що забезпечує зручний попередній перегляд різних станів інтерфейсу додатка. Для створення нестандартних інтерфейсів присутній майстер створення елементів дизайну, що підтримує використання шаблонів. Середовище має вбудовані функції для завантаження типових прикладів коду з *GitHub*.

До складу також входять налаштовані платформи *“Android”,* розширені інструменти рефакторингу, перевірки сумісності з минулими проблемами, виявлення проблем продуктивності, моніторинг споживання пам'яті та оцінка зручності використання. У редакторі додано швидкий режим внесення змін.

Підсвічування, статичний аналіз і виявлення помилок системи розширена підтримка *API "Android"*. Інтегрована підтримка оптимізатора коду *ProGuard*. Вбудовані засоби генерації цифрового підпису. Надано інтерфейс для керування перекладами на інші мови.

Структура програмного забезпечення складається з файлів *Grandle Scripts*, котрі використовується для різних налаштувань, керування додатком, тощо. Головними складовими додатку є три каталоги:

1. Маніфест, який містить один файл *Manifest.xml*. Цей файл є обов'язковим у кожному проекті *Android* і містить декларації компонентів додатка, таких як активності *(Activities),* сервіси, отримувачі *(Broadcast Receivers)* та інші елементи, які складають ваш додаток.

2. *Java*, який містить дві вкладені папки, одна з яких містить файли класів, а інша створена для тестів.

3. *Res*, який містить файли ресурсів, які розповсюджуються окремо підпапками, такими як: *drawable*, котрий містить графічні ресурси, призначені для різних розмірів екрана; *layout* містить *xml*-файли, які описують зовнішній вигляд форм та їх різноманітних елементів; при створенні проекту вже є файл *activity_main.xml*, який відповідає за перегляд основної програми *Activity*); меню - ця папка містить ресурси для меню; вже створений файл *menu_main.xml*, який відповідає за меню *main Activity* програма; *mipmap* зберігає різні піктограми програми та її структура збігається з *drawable*, у якій раніше все це містилося, а тепер використовується для інших графічних ресурсів і каталог змінних він містить будь-які рядкові ресурси та ресурси теми, кольори, стилі тощо.

3.2. Технології, котрі використовуються для покращення безпеки інформації

Задля поліпшення надійності захисту інформації було проаналізовані різні рішення і можливості поліпшення захисту інформації та на скільки було можливо, вивчено захист у найбільших та популярних у системах віддаленого спілкування.

На основі отриманої інформації було обрано наступні рішення та технології:

– Алгоритм шифрування *AES (Advanced Encryption Standard).*

– Генератор складних паролів.

– Подвійна обфускація коду.

– Перевизначив правила використання бази даних користувачами.

3.2.1. Алгоритм *AES*

Алгоритм шифрування *AES (Advanced Encryption Standard)* є одним із найбільш поширених та надійних симетричних алгоритмів шифрування. Він був прийнятий Національним Інститутом Стандартів і Технологій (*NIST*) у 2001 році як стандарт шифрування для захисту чутливої інформації. *AES* використовує один ключ для шифрування та розшифрування даних, тому він є симетричним алгоритмом.

Мною було обрано використання 128-бітного ключа, ключ складається з 16 випадково згенерованих значень. Кожен символ *ASCII* кодується 8-бітним байтом, отже, 16 символів дорівнюють 16 * 8 = 128 бітам. Розмір цього ключа є достатнім для того, щоб його підбір був достатньо складним, а його швидкодія була на достатньому рівні, а ресурсозатратність не шкодила працездатності системи.

3.2.2. Генератор складних паролів

Користувачі не завжди мають бажання або час щоб придумати складний та унікальний пароль і нехтують безпекою своєї інформації, тому було додано генератор складних паролів. Генератор використовує символи з кода *ASCII*. Мінімальний символ (33) - це символ '!' та максимальний символ (126) - '~' (у коді *ASCII*). Отже, символи, які можуть бути використані у згенерованих паролях, включають у себе всі символи, які знаходяться між '!' і '~' в таблиці *ASCII*, включаючи літери верхнього та нижнього регістрів, цифри, спеціальні символи і знаки пунктуації.

Цей діапазон включає в себе багато різних символів, і паролі, створені з використанням цього коду, будуть містити символи, що можуть бути важкими для перебору або підбору, забезпечуючи деякий рівень безпеки.

Також генерація довжини пароля опирається на рандомізацію, тому довжина паролів у користувачів буде різною, що додатково ускладнює підбір паролів. Задля зручності їх подальшого зберігання, згенерований пароль одразу зберігається у буфер обміну, задля подальшого реалізації у інших полях та зберігання у захищених місцях.

3.2.3. Обфускація коду додатку

Обфускація коду - це техніка захисту програмного коду, яка полягає в перетворенні читабельного вихідного коду програми в менш зрозумілий і складніший код, що називається “обфускованим кодом”. Основна мета обфускації полягає в тому, щоб ускладнити реверс-інженерінг, зменшити ймовірність незаконного копіювання програми та забезпечити додатковий рівень безпеки для програмного продукту. Додатковими функціями, котрі не впливають на захист додатку та системи в цілому є прибирання зайвого коду та коментарів, методів та функцій.

Основні методи обфускації включають наступне:

– Перейменування змінних і функцій. Імена змінних, функцій та методів можуть бути замінені на незрозумілі та короткі ідентифікатори, що робить код менш зрозумілим.

– Зміна порядку інструкцій. Порядок виконання інструкцій може бути змінений, щоб заплутати читача і зробити аналіз коду складнішим.

– Видалення коментарів: коментарі та зайвий вихідний текст можуть бути видалені, щоб зменшити зрозумілість коду.

– Видалення непотрібних інструкцій. Непотрібні або дубльовані інструкції можуть бути видалені, щоб зменшити обсяг коду.

– Використання бінарних операцій. Замість звичайних арифметичних операцій, таких як додавання і віднімання, можуть використовуватися бінарні операції, що робить код менш зрозумілим.

У середовищі розробки вже запропоноване стандартне рішення для обфускації коду і використовується *ProGuard* або *R8* (в залежності від налаштувань). Але так як воно стандартне, то кожен може вивчити його

алгоритми та логіку роботи, що ставить під сумнів надійність та доцільність використання. Тому було вирішено, що задля усунення недоліків обфускатора *ProGuard* або *R8*, використати ще один обфускатор, тобто відбувається подвійна обфускація коду, за різними алгоритмами [12].

Додатковим обфускатором є *Allatori* - це один із багатьох інструментів, які використовуються для обфускації *Java*-коду. *Allatori* надає такі функції обфускації:

– Перейменування змінних і методів. *Allatori* може змінити імена змінних і методів на більш складні та беззмістовні назви. Це робить код важчим для зрозуміння.

– Видалення непотрібного коду. *Allatori* може видаляти непотрібні методи, змінні та класи з коду, що дозволяє зменшити його розмір.

– Перетворення рядків. він може перетворити літерали рядків на менш зрозумілі значення, що робить важчим їх розкриття.

– Захист від деобфускації. *Allatori* намагається запобігти деобфускації свого коду, роблячи його важчим для аналізу.

– Зміна структури коду. Обфускатор може перегруповувати код, щоб ускладнити його аналіз.

- Захист від злому. *Allatori* також надає захист від декомпіляції та злому коду [13].

Помітно, що обфускація коду може покращити безпеку вашого додатка. Обфускація рекомендується для додатків, де програмний код продукту є важливим і його пряме зчитування може призвести до проблем з безпекою, особливо в тих випадках, коли ви тримаєте здебільшого додаток на клієнтському боці.

3.2.4. Правила безпеки у базі даних

Щоб покращити безпеку *Firebase Realtime Database* і дозволити тільки авторизованим користувачам взаємодіяти з базою даних, треба використовувати правила, що забезпечать потрібну логіку.

Правила, що реалізують потрібні користувацькі обмеження:

```
{
  "rules": {
    ".read": "auth != null", // Тільки авторизовані користувачі можуть читати дані.
    ".write": "auth != null", // Тільки авторизовані користувачі можуть записувати дані.
  }
}
```

Ці правила встановлюють, що доступ до читання та запису в базі даних доступний тільки для авторизованих користувачів. Якщо користувач не авторизувався, він не матиме доступу до даних в базі.

Крім цього, ви можете застосовувати більш специфічні правила для окремих частин вашої бази даних, якщо потрібно обмежити доступ окремих користувачів або груп до конкретних даних. Наприклад:

```
{
  "rules": {
    "publicData": {
      ".read": "auth != null", // Для загальних даних доступ до читання є для всіх авторизованих користувачів.
      ".write": "auth != null", // Для загальних даних доступ до запису є для всіх авторизованих користувачів.
    },
    "privateData": {
      ".read": "auth != null && root.child('users').child(auth.uid).child('role').val() == 'admin'",
      ".write": "auth != null && root.child('users').child(auth.uid).child('role').val() == 'admin'",
    }
  }}
```

В цьому прикладі дані у *“publicData”* доступні для читання і запису всім авторизованим користувачам. У *“privateData”* доступ до читання і запису мають лише користувачі з роллю *“admin”*, і їхні ролі зберігаються в іншій частині бази даних.

3.3. Принцип роботи системи дистанційного спілкування

Система заснована на користувачах, які використовують для спілкування мобільний додаток, який повинен підключатися через Інтернет до сервісу *Google Firebase*. Служба автентифікації *Google Firebase* дозволяє користувачеві зареєструватися та увійти в мобільну програму для доступу до бази даних і функцій програми, одночасно захищаючи конфіденційність електронною поштою та паролем (рис. 3.1).

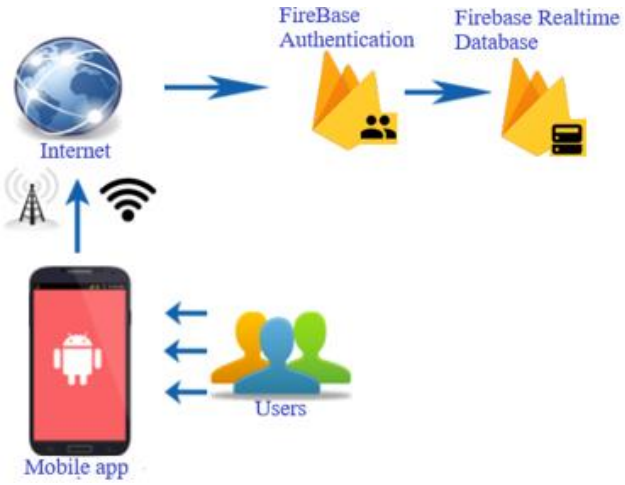

Рис. 3.1. Зніком принипів реєстрації та авторизації користувачів

Принцип надсилання та отримання повідомлень користувачами:

1. Користувач вибирає одержувача та натискає на його профіль.

2. Відправник надсилає повідомлення одержувачу. Це повідомлення з ідентифікатором відправника та одержувача надходить до бази даних.

3. База даних надсилає повідомлення одержувачу за ідентифікатором одержувача.

4. Одержувач може надіслати відповідь так само, як описано на другому кроці (рис. 3.2).

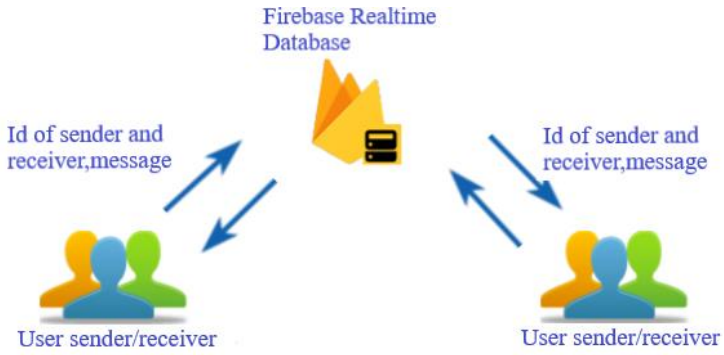

Рис. 3.2. Зніком принципу передачі повідомлень

3.4. Показники системи дистанційного спілкування

Провівши аналіз важливих показників системи дистанційного спілкування можна виділити наступні:

1. Корисність у використанні. У сучасному світі важко уявити собі день без онлайн комунікації. Тому функції цього додатку досить корисні, адже додаток дозволяє людям спілкуватися один з одним, не витрачаючи зайвого часу. Наприклад, цей додаток можна використовувати для спілкування між студентами та викладачами.

2. Функціональність. Функціонал даного додатку на задовільному рівні, все, що було заплановано, додаток виконує. Цей додаток легко оновлювати та підтримувати його життєвий цикл.

3. Доступність для користувачів. Цей додаток базується на операційній версії *android* 5.0 під назвою *Lollipop*. За статистикою, цим додатком зможуть скористатися 98% користувачів мобільних пристроїв на базі *“Android”*. Тому цей додаток має високу оцінку доступності.

4. Можливість провести тестування. Додаток був розроблений за принципів модулів з використанням архітектури, котра передбачає чітко розділені файли, кожен з яких виконує лише те, що необхідно. Якщо логіка файлу виходить за межі

його функціональності, для цього коду створюється новий файл. Завдяки такій реалізації систему легко підтримувати, тестувати, знаходити та виправляти помилки.

5. Підтримуваність. Оскільки використовується шаблонна архітектура додатку, популярні та загальноприйняті рішення побудови додатку, його буде легко оновлювати та впроваджувати нові функції.

6. Безпека. Система дистанційного спілкування має багатошаровий захист: реєстрація та авторизація за допомогою сервісу *Firebase Authenticator*, яка дає можливість доступу до бази даних лише за допомогою електронної пошти та пароля; покращені правила доступу бази даних; використовується протокол передачі даних *HTTPS*, який підтримує шифрування за допомогою криптографічних протоколів *SSL* і *TLS*; надано можливість користувачам генерувати складний пароль; додано подвійну обфускація коду додатку, що запобігає читанню викраденого коду додатку; шифрування та розшифрування повідомлень, котрими обмінюються користувачі за допомогою алгоритму *AES*.

3.5. Практична реалізація технологій захисту інформації

3.5.1. Реалізація алгоритмів *AES* для шифрування повідомлень

За допомогою надійного генератора випадкових чисел створимо симетричний ключ для шифрування та розшифрування даних. Ініціалізуємо об’єкти, необхідні для роботи функцій шифрування та розшифрування. Розмір ключа буде складати 16 символів, що дорівнює 128 бітам (рис. 3.3).

```
private final byte[] encryptionKey =
        {9, 115, 51, 86, 105, 4, -31, -23, -68, 88, 17, 20, 3, -105, 119, -53};
3 usages
private Cipher cipher, decipher;
3 usages
private SecretKeySpec secretKeySpec;
```

Рис. 3.3. Знімок ключа для шифрування

Після створення ключа та об’явлення об’єктів, створюються об’єкти *“cipher”* і *“decipher”*.

Котрі використовуються для ініціалізації шифрування і розшифрування за допомогою алгоритму AES (рис. 3.4).

```
try {
    cipher = Cipher.getInstance( transformation: "AES");
    decipher = Cipher.getInstance( transformation: "AES");
} catch (NoSuchAlgorithmException | NoSuchPaddingException e) {
    e.printStackTrace();
}
```

Рис. 3.4. Знімок створення об’єктів *“cipher”* і *“decipher”*

Наступним кроком є створення об’єкту *“secretKeySpec”,* який використовує ключ шифрування для ініціалізації *“Cipher”* перед шифруванням і розшифруванням (рис. 3.5).

```
secretKeySpec = new SecretKeySpec(encryptionKey,  algorithm: "AES");
```

Рис. 3.5. Знімок створення об’єкта *“secretKeySpec”*

Створюємо методи для шифрування та розшифрування:

AESEncryptionMethod(String string) метод шифрує вхідний текстовий рядок, перетворюючи його в байтовий масив, а потім застосовує шифрування *AES* з використанням вказаного ключа (рис. 3.6).

```
private String AESEncryptionMethod(String string){

    byte[] stringByte = string.getBytes();
    byte[] encryptedByte = new byte[stringByte.length];

    try {
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec);
        encryptedByte = cipher.doFinal(stringByte);
    } catch (InvalidKeyException | IllegalBlockSizeException
             | BadPaddingException e) {
        e.printStackTrace();
    }

    String returnString;

    returnString = new String(encryptedByte,
            StandardCharsets.ISO_8859_1);
    return returnString;
}
```

Рис. 3.6. Знімок створення методу для шифрування даних

Після шифрування він перетворює байтовий масив у рядок і повертає його. Коли зашифрований байтовий масив готов, він передається на сервер, де зберігається і при першій нагоді передається отримувача повідомлення .

AESDecryptionMethod(String string) метод приймає зашифрований текстовий рядок, перетворює його в байтовий масив, а потім застосовує розшифрування *AES* з використанням того самого ключа. Після розшифрування він повертає розшифрований рядок (рис. 3.7).

```
private String AESDecryptionMethod(String string)
        throws UnsupportedEncodingException {
    byte[] EncryptedByte = string.getBytes(StandardCharsets.ISO_8859_1);
    String decryptedString = string;

    byte[] decryption;

    try {
        decipher.init(Cipher.DECRYPT_MODE, secretKeySpec);
        decryption = decipher.doFinal(EncryptedByte);
        decryptedString = new String(decryption);
    } catch (InvalidKeyException | BadPaddingException
             | IllegalBlockSizeException e) {
        e.printStackTrace();
    }
    return decryptedString;
}
```

Рис. 3.7. Знімок створення методу для розшифрування даних

Обмін зашифривонами повідомленнями відбувається наступним чином:

1. Користувач друкує повідомлення та натискає кнопку “відправити”.
2. Повідомлення шифрується за допомогою обраного алгоритму.
3. Зашифроване повідомлення відправляється до бази даних де і зберігається.
4. З бази даних зашифроване повідомлення надходить до отримувача.
5. Зашифроване повідомлення проходить етап розшифрування.
6. Після успішного розшифрування повідомлення відображається отримувачу.

Даний процес ідентичний для обох учасників спілкування.

3.5.2. Реалізація генерації складних паролів

За створення складних паролів для користувача відповідає одна функція, котра використовує таблицю *ASCII*, починаючи з тридцять третього елементу таблиці і закінчуючи сто двадцять шостим елементом (рис. 3.8). Функція після створення паролю повертає його і вставляє у потрібне поле для вводу паролю. Пароль буде приховано спеціальними знаками.

```
private static final int MIN_CODE = 33, MAX_CODE = 126;

1 usage
public static String process(int lenght){
    StringBuilder builder = new StringBuilder();

    for (int i = 0; i< lenght; i++){
        builder.append((char) ThreadLocalRandom.current()
                .nextInt(MIN_CODE, bound: MAX_CODE + 1));
    }
    return builder.toString();
}
```

Рис. 3.8. Знімок створення методу для генерації випадкового паролю

Довжина паролю є випадковою так як і символи, з якого він складається. Після створення, пароль зберігається в буфер обміну та відображається у полі для паролю.

3.5.3 Реалізація та налаштування обфускаторів

Реалізація стандартного обфускатора *ProGuard* виконується лише зміною одного значення з *“false”* на *“true”* (рис. 3.9).

```
buildTypes {
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
                'proguard-rules.pro'
    }
```

Рис. 3.9. Знімок ввімкнення обфускатора *ProGuard*

На відміну від стандартного обфускатора, імплементація *Allatori* займає більше часу і має більше коду і додаткових файлів.

Для початку у проект треба додати файл, котрий містить в собі різні тонкі налаштування його методів і функцій та створити випадкове значення, для унікальності результату його роботи (рис. 3.10).

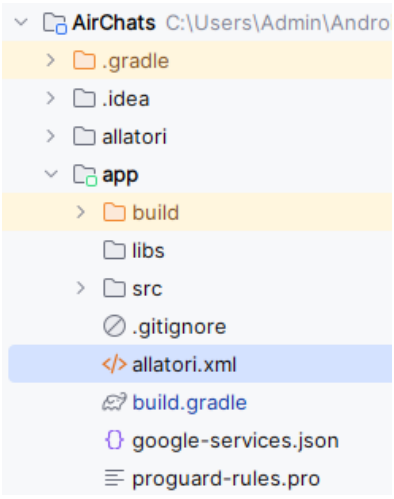

Рис. 3.10. Знімок зображує доданий файл у структуру проекту

Наступним кроком буде додавання у файл з налаштуваннями проекту коду, що буде запускати обфускатор кожен раз при компіляції проекту (рис. 3.11).

```
applicationVariants.configureEach { ApplicationVariant variant ->
    variant.javaCompileProvider.get().doLast {
        runAllatori(variant)
    }
```

Рис. 3.11. Знімок зображує код запуску обфускатоа *Allatori* при кожній компіляції проекта

3.6. Тестування технологій захисту інформації

3.6.1. Тестування роботи алгоритму *AES*

Для початку запустимо додаток та відправимо два тестових повідомлення, українською та англійською (рис. 3.12).

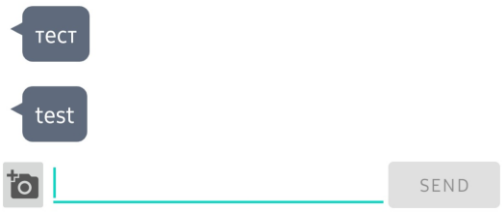

Рис. 3.12. Знімок відправлених повідомлень

Повідомлення були зашифровані і відправлені у базу даних. На (рис.3.13) наведено інформації про два останніх тестових повідомлення. Інформацію про відправника, його ім’я та айді, айді отримувача та відправленні повідомлення, котрі надійшли та зберігаються у зашифрованому вигляді у базі даних.

Отримувачу надійде зашифроване повідомлення, але перед його відображенням, воно розшифрується.

Отже, у результаті стає зрозуміло, що будь-які повідомлення успішно шифруються та розшифровуються. Результат роботи алгоритму відповідає очікуванням і вдовольняє потреби шифрування.

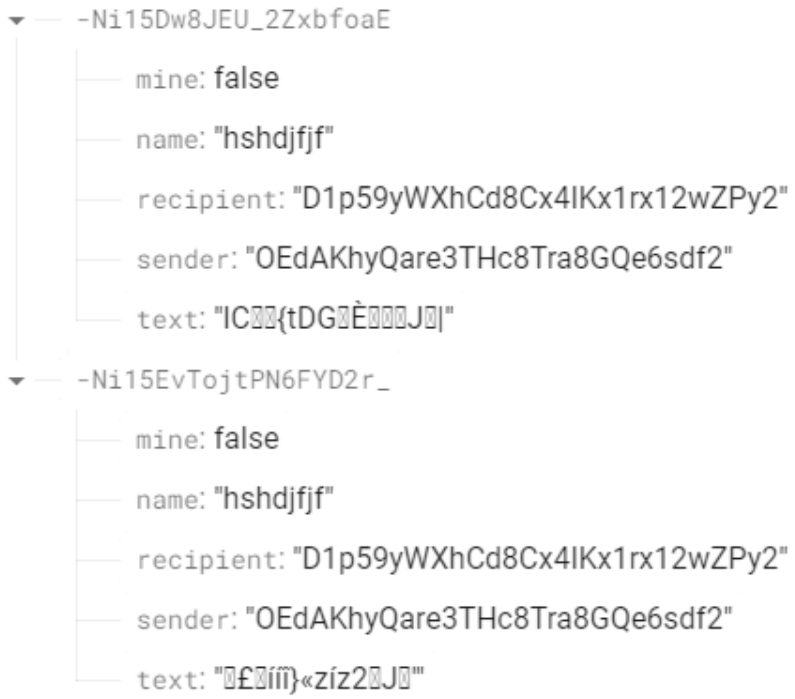

Рис. 3.13. Знімок останніх двох тестових повідомлень

3.6.2. Перевірка генерації складних паролів

Під час реєстрація нового акаунту, користувачеві доступна можливість створити складний згенерований пароль, задля цього створено відповідну кнопку. Коли користувач натисне на кнопку *“Generate password”* спрацює потрібний метод і користувач отримає складний, згенерований пароль, котрий автоматично з’явиться у полі *“Enter password”* та додасться у буфер обміну, щоб користувач міг його повторно використати та зберегти у надійному місці. Це полегшить для нього процес реєстрації та покращить захист його конфіденційної інформації.

Задля наглядності, згенерований пароль вставимо у поле *“Enter your name”* (рис. 3.14).

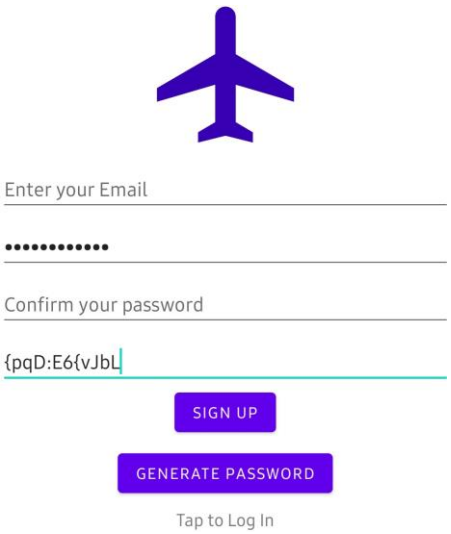

Рис. 3.14 Знімок результату створення паролю

Отже, маємо достатньо складний пароль, дуже відмінний від тих, котрі ми звикли бачити. Такі паролі достатньо складно підібрати, що у свою чергу допомагає користувачеві захистити свої персональні дані. А зручність у використанні, спонукатиме користувачів використовувати переважно авто-генерацію складних паролів. Результат роботи методу створення паролів повністю відповідає очікуванням.

Висновки за розділом

У третьому розділі детально розглядається середовище розробки, імплементація алгоритмів і технологій для покращення захисту конфіденційної інформації у системах дистанційного спілкування, також, протестовано їх роботу.

Середовище розробки *“Android Studio”* обраний за певними критеріями: інтеграція необхідних функцій для нових версій *“Android”*, вбудований *SDK*, швидкість роботи, підтримка *Gradle*, зручний інтерфейс, мова *Java* була обрана через кількість доступних бібліотек, оновлень і підтримки *“Android Studio”*.

“Android Studio”, заснований на програмному забезпеченні IntelliJ IDEA від JetBrains, офіційному інструменті для розробки додатків «Android». Ця розробка середовища доступна для Windows, OS X і Linux.

Алгоритмом шифрування повідомлень обрано алгоритм *AES(Advanced Encryption Standard. AES* – симетричний алгоритм шифрування, основним елементом, яким оперує *AES*, є байт. *AES* базується на архітектурі *SQUARE* , для якої характерно:

1) Представлення блоку у вигляді масиву байтів.

2) Шифрування за один раунд всього блоку даних.

3) Виконання криптографічних перетворень.

Алгоритм *AES* має кілька переваг в порівнянні з іншими популярними алгоритмами шифрування, які роблять його привабливим для використання в різних застосунках:

– *AES* вважається надзвичайно безпечним алгоритмом. Він пройшов ретельний аналіз і велику кількість криптоаналітичних атак, і дотепер не має публічних атак, які б дозволили дешифрувати дані, зашифровані за допомогою правильно вибраного ключа.

– Висока швидкість, гнучкість ключа.

– *AES* є стандартом у багатьох сферах, включаючи інформаційну безпеку, фінансові системи, телекомунікації, хмарні служби, та багато інших галузей.

– *AES* може бути реалізований на різних платформах та вбудований в апаратне забезпечення, що дозволяє ефективно використовувати обмежені ресурси, такі як мобільні пристрої.

– *AES* є публічним стандартом, який підтримується і рекомендується урядовими органами, банками та різними міжнародними організаціями, що сприяє його поширенню та використанню.

– *AES* може використовуватися в різних режимах шифрування, що дозволяє вибрати найбільш підходящий для конкретних потреб, такі як *ECB, CBC, CFB, OFB*, і деякі інші.

Хоча *AES* має багато переваг, важливо пам'ятати, що ефективність шифрування також залежить від правильного вибору ключів і відповідної реалізації в конкретному додатку або системі.

Успішно розроблено метод для генерації складних користувацьких паролів. Цей метод допомагає користувачам створити багатозначний пароль, натиском на одну кнопку. Створений пароль використовує символи, які знаходяться між '!' і '~' в таблиці *ASCII*, включаючи літери верхнього та нижнього регістрів, цифри, спеціальні символи і знаки пунктуації.

Задля забезпечнння захисту програмного коду та уникнення реверс-інженерінгу додатку, зменшити ймовірність незаконного копіювання програми та забезпечити додатковий рівень безпеки для програмного продукту використовується обфускатор *Allatori. Allatori* - це один із багатьох інструментів, які використовуються для обфускації *Java*-коду. Його обрання обґрунтовується підтримкою програмної мови Java та потрібним набором можливостей.

Використання обфускатора *Allatori* надає наступні переваги:

– Перейменування змінних і методів: *Allatori* може змінити імена змінних і методів на більш складні та беззмістовні назви. Це робить код важчим для зрозуміння.

– Видалення непотрібного коду. *Allatori* може видаляти непотрібні методи, змінні та класи з коду, що дозволяє зменшити його розмір.

– Перетворення рядків. він може перетворити літерали рядків на менш зрозумілі значення, що робить важчим їх розкриття.

– Захист від деобфускації. *Allatori* намагається запобігти деобфускації свого коду, роблячи його важчим для аналізу.

– Зміна структури коду. обфускатор може перегруповувати код, щоб ускладнити його аналіз.

– Захист від злому. *Allatori* також надає захист від декомпіляції та злому коду.

Використовуючи всі вище перелічені методи та алгоритми захисту інформації, суцільний захист системи стає багатошаровим, що ставить перед

зловмисниками комплексну задачу. Навіть якщо зловмисник подолає один рівень захисту, на нього чекатимуть наступні.

Отже, реалізація усіх обраних алгоритмів та методів захисту інформації було успішно виконане. Комплексне тестування дало результат – всі компоненти системи працюють згідно із очікуванням, алгоритми та методи захисту інформації успішно виконують свої функції. Загальний рівень безпеки системи значно підвищився. Результати – задовільні.

ВИСНОВКИ

Зважаючи на природу людини, ми є соціальними істотами і потребує спілкування з іншими людьми. Із зростанням кількості населення, зростають і розповсюджуються зв’язки між ними. Через пандемію *COVID-19*, люди по-всьому світі були змушені ізолюватись вдома і їх життя тоді безповоротно змінилося. Людству довелося адаптовуватися до нових реалій життя. Неочікувана поява вірусу вимагала від населення світу прийняти нові заходи та змінити підходи до спілкування, роботи, навчання та навіть здійснення покупок. У цей час системи дистанційного спілкування стали ключовим інструментом для забезпечення безпеки та неперервності соціальних і професійних взаємодій.

Системи дистанційного спілкування перетворилися на головний засіб збереження зв'язків між людьми у віддалених регіонах та навіть в межах одного місця. Вони стали не тільки засобом забезпечення продуктивної роботи та навчання в умовах самоізоляції, але й інструментом для підтримки психологічного здоров'я та соціальної взаємодії в умовах обмежень.

Дистанційне спілкування стало парадигмою нової реальності, де віддаленість вже не є перешкодою для обміну інформацією, навчання, роботи чи навіть проведення особистих подій. Важливість цих систем полягає в їхній здатності не лише забезпечувати спілкування, але й викликати почуття злагодженості та співпраці у віртуальному просторі.

COVID-19 ретельно продемонстрував, що системи дистанційного спілкування не є просто доповненням до звичного способу життя, вони стали важливою складовою для збереження безпеки, ефективності та підтримки в умовах кризи. Тим самим вони відкрили нові можливості та перетворили спосіб, яким ми сприймаємо та взаємодіємо з світом. До пандемії, коли ці технології розглядалися як вторинні та менш важливі, смартфони та інші пристрої для дистанційного спілкування тепер займають лідируючі позиції у своїй доступності та універсальності. Недоліки традиційного, “живого” спілкування, такі як

обмеження в часі та місці, легко нівелювались , оскільки віддалене спілкування дозволяє зв'язувати людей з будь-якої точки планети, що веде до більш широкого та глибшого обміну інформацією та ідеями.

Порівнюючи смартфони з іншими пристроями для віддаленого спілкування, слід відзначити їхню мобільність, яка робить їх надзвичайно зручними для використання у будь-який час та місце. Ця мобільність дозволяє людям бути постійно на зв’язку, незалежно від того, чи йдеться про професійні, особисті або навіть невеликі щоденні комунікації. Тому смартфони є найактуальнішим пристроєм для дистанційного спілкування.

Аналіз предметної області показав, що на сьогоднішній день найпопулярніші форми дистанційного спілкування це: відеоконференції, чати (месенджери) та соціальні мережі.

Зі зростанням тренду віддаленої роботи та розвитком технологій спілкування, паралельно зростає і необхідність покращення всіх можливих методів та алгоритмів захисту приватної інформації. Адже зараз світ проходить наймасовіший етап цифровізації, з кожним днем робляться кроки та прикладаються до цього зусилля. Все більше приватної інформації підіймається з архівів та оцифровується, все більше користувачів довіряють свої дані компаніям, котрі зберігають їх дані на серверах та особисто діляться ними різними соціальними мережами. Тому захист інформації у системах дистанційного спілкування є надзвичайно критичною проблемою за багатьма аспектами і є гостра необхідність у постійному розвитку захисту цих систем.

Наразі основними загрозами приватної інформації у системах дистанційного спілкування є:

– Вразливості програмного забезпечення.

– Віддалений або фізичний доступ до пристроїв або носіїв інформації.

– Фішингові атаки, кібератаки, шахрайство, шпигунське ПЗ .

– Соціальний інжиніринг, віруси.

– *Packet Sniffing* (прослуховування пакетів), *Wi-Fi* Інтерцептування.

– *Man-in-the-Middle (MITM)* атаки.

Проаналізувавши різні рішення стосовно забезпечення захисту інформації такі як: криптографічні методи, цифрові підписи та електронні цифрові ключі, протоколи передачі даних, *VPN*, захист автентифікації та авторизації, захищене сховище даних, було обрано наступне для практичної реалізації:

– Криптографічні методи.

– Захист авторизації та автентифікації.

– Використання захищеного сховища даних.

– Додаткові методи такі як: обфускація коду додатку та генерація складних паролів для користувачів.

Розглядалися чотири алгоритми шифрування: *AES, RSA, 3DEC, ECC*.

AES (Advanced Encryption Standard) - це симетричний алгоритм шифрування, який використовується для захисту конфіденційності даних. Він є одним з найпопулярніших і найбільш надійних алгоритмів шифрування, який використовується у багатьох сучасних системах та протоколах для захисту інформації.

RSA (Rivest-Shamir-Adleman) - це асиметричний криптографічний алгоритм, який використовується для шифрування та цифрового підпису. Цей алгоритм базується на проблемі факторизації великих простих чисел, яка є важкою для обчислень у випадку великих чисел.

Elliptic Curve Cryptography (ECC) - це сучасний підхід до криптографії, який базується на використанні властивостей еліптичних кривих для створення криптографічних систем. *ECC* використовує математичні властивості еліптичних кривих для забезпечення безпеки обміну даними та створення цифрових підписів.

DES (Data Encryption Standard) може вважатися одним з популярних алгоритмів шифрування. Це симетричний блочний шифр, який був розроблений у 1970-х роках. Однак через зростання обчислювальної потужності комп'ютерів, яка зробила *DES* вразливим до атак *brute force*, його вважають застарілим з точки зору безпеки. *DES* використовував ключ довжиною 56 біт, що робило його вразливим до перебору ключів.

3DES — це варіант *DES*, який використовується для підвищення безпеки, повторно застосовуючи *DES* з трьома ключами. Це дозволяє підвищити безпеку до певної міри, але *3DES* також вважається менш ефективним порівняно з більш сучасними алгоритмами шифрування через його обмежену швидкодію.

Провівши порівняльний аналіз чотирьох алгоритмів шифрування і склавши їх порівняльну таблицю можна зробити висновок, що найбільш оптимальним і актуальним для практичної реалізації буде алгоритм *AES*.

Також реалізований додатковий захист вихідного коду додатку за допомогою двох обфускаторів *ProGuard* та *Allatori*. Обфускація коду - це техніка захисту програмного коду, яка полягає в перетворенні читабельного вихідного коду програми в менш зрозумілий і складніший код, що називається “обфускованим кодом”. Основна мета обфускації полягає в тому, щоб ускладнити реверс-інженерінг, зменшити ймовірність незаконного копіювання програми та забезпечити додатковий рівень безпеки для програмного продукту. Реалізація подвійної обфускації пояснюється тим, що обфускатори *ProGuard* та *Allatori* мають різні можливості і поєднуючи їх, можна отримати унікальне рішення та чудовий захист. Адже вони не працюють окремо один від одного, а разом обфускують код, роблячи результат непередбачуваним та занадто складним у маніпуляції для зловмисника.

Для додаткового захисту системи, було реалізовано захист авторизації та автентифікації за допомогою електронної пошти та паролю, щоб користувацькі дані залишалися приватними і їх конфіденційність не порушувалась.

Для нових користувач розроблено рішення для полегшення створення складних паролів, бо часто користувачі цим нехтують через певні особисті причини. Користувачам надається можливість згенерувати складні паролі з використанням символів *ASCII*. Мінімальний символ (33) - це символ '!' та максимальний символ (126) - '~' (у коді *ASCII*), тобто можуть використовуватися всі символи у діапазоні від 33 – 126 символу.

Стосовно компонентів системи дистанційного спілкування, то заздалегідь було обрано безпечні рішення для обміну інформацією між користувачами, цим

рішенням є база даних *Firebase Realtime Database*. Ця база даних має свій внутрішній захист, використовує захищені протоколи передачі інформації та безпечні правила її користуванням.

Всі ці рішення створюють надійну багатошарову оболонку захисту користувацької інформації, котра захищає користувача на всіх етапах користування системою дистанційного спілкування.

Під час практичної реалізації і тестування стало зрозуміло, що системи без захисту зовсім ненадійні і реалізація навіть мінімального захисту суворо необхідна. Під час тестування було отримано очікуваний та успішний результат. Стало зрозуміло, що реалізація навіть мінімального об’єму захисту не є складною задачею і обов’язкова до виконання, навіть якщо системою дистанційного спілкування користується декілька людей.

Отже, кожна система дистанційного спілкування має використовувати методи та алгоритми захисту інформації. Кожен крок у захисту системи, навіть найменший, має бути реалізований. Адже нехтування безпекою може призвести до різного ступеню небажаних наслідків.

Кожен розробник багатокористувацьких систем має бути освіченим у сфері захисту інформації. Кожен користувач має бути освіченим у необхідності, та базовим методам захисту персональної інформації. Адже у сфері технологій існує постійна еволюція, де деякі інноваційні рішення продовжують розвиватися та вдосконалюватися, тимчасом як інші технології поступово втрачають свою актуальність і стають менш вживаними чи ефективними, також не варто забувати, що щодня зловмисники навчаються. Тому універсального та ідеального рішення у захисті інформації на сьогодні не існує. Єдине рішення можна поділити на дві складові:

– Для розробників це побудова захисту, використовуючи сучасні рішення, у майбутньому їх підтримка та оптимізація, за необхідності заміна і додавання нових методів та алгоритмів.

– Для користувачів це ознайомлення з правила захисту персональної інформації і їх сумлінне дотримання.

Коли щось одне з вище переліченого не виконується, то ніяка система не зможе надати досконалий та надійний захист.

Захищена система дистанційного спілкування була успішно розроблена і протестована. Було проведено поетапне тестування всіх окремих модулів на двох віртуальних пристроях з різними параметрами, також успішно протестовано на двох фізичних пристроях. Всі помилки виправлено, система дистанційного спілкування працює успішно, імплементований захист успішно виконує поставлені задачі, всі завдання виконані.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *The Complete Guide to Remote Communication* [Електронний ресурс]. — Режим доступу до ресурсу: https://www.techsmith.com/blog/remote-communication-visuals/ (дата звернення 10.10.23р). – Назва з екрана.

2. *Data Protection in the Age of the Remote Working Environment* [Електронний ресурс]. — Режим доступу до ресурсу: https://www.linkedin.com/pulse/data-protectionage-remote-working-environment-lalit-popli/ (дата звернення 14.10.23р). – Назва з екрана.

3. *Remote work cybersecurity: 12 risks and how to prevent them* [Електронний ресурс]. — Режим доступу до ресурсу: https://www.techtarget.com/searchsecurity/tip/Remote-work-cybersecurity-12-risks-and-how-to-prevent-them (дата звернення 16.10.23р). – Назва з екрана.

4. Криптографія [Електронний ресурс]. — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F (дата звернення 03.11.23р). – Назва з екрана.

5. Практична криптологія. [Електронний ресурс]. — Режим доступу до ресурсу: https://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%209.pdf (дата звернення 03.11.23р). – Назва з екрана.

6. Алгоритм шифрування *RSA*, види атак на нього. Реалізація мовою *Python* [Електронний ресурс]. — Режим доступу до ресурсу: https://dou.ua/forums/topic/43026/ (дата звернення 03.11.23р). – Назва з екрана.

7. Шифр *DES* [Електронний ресурс]. — Режим доступу до ресурсу: https://stud.com.ua/179769/informatika/shifr (дата звернення 03.11.23р). – Назва з екрана.

8. Алгоритм шифрування *3DES* [Електронний ресурс]. — Режим доступу до ресурсу: https://studfile.net/preview/9129743/page:13/ (дата звернення 03.11.23р). – Назва з екрана.

9. Ковтун В. Ю. Методи та алгоритми арифметичних перетворень із зменшеною обчислювальною складністю на кривих алгебри для криптографічних Додатків: Дисертація кандидата технічних наук: Системи захисту інформації. - Харківський університет Повітряних Сил. – Україна: Харків, 2005. – 249 с.

10. Богуш В.М. Теоретичні основи захищених інформаційних технологій. Навч. Посібник / В.М. Богуш, О.А. Довидьков, В. Г. Кривуца – К.: ДУІКТ, 2010. – 454 с.

11. Швець О.Ю., Лазаренко В.В. Аналіз методів і засобів захисту інформації та сучасних вимог до них / О.Ю. Швець, В.В. Лазаренко [Електронний ресурс]. — Режим доступу до ресурсу: *http://www.rusnauka.com/25_DN_2008/Informatica/28842.doc.htm* (дата звернення 08.11.23р). – Назва з екрана.

12. Гайворонський М.В. Безпека інформаційно-комунікаційних систем / М.В. Гайворонський, О.М. Новіков. – К.: Видавнича група BHV, 2009. – 608 с.

13. Дронь М.М. Основи теорії захисту інформації: Навч. Посібник / М.М.

14. Кузьменко Б.В. Захист інформації. Навчальний посібник. Ч. 1. (Організаційно-правові засоби забезпечення інформаційної безпеки) / Б.В. Кузьменко, О.А. Чайковська – К.: Техносвіт, 2009. – 83 с.

15. *What Is AES Encryption and How Does It Work?* [Електронний ресурс]. — Режим доступу до ресурсу: *https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption* (дата звернення 27.11.23р). – Назва з екрана.

16. *Shrink, obfuscate, and optimize your app* [Електронний ресурс]. — Режим доступу до ресурсу: *https://developer.android.com/build/shrink-code* (дата звернення 28.11.23р). – Назва з екрана.

17. *Allatori Java Obfuscator* [Електронний ресурс]. — Режим доступу до ресурсу: *http://english.cogitosoft.com/html/product/item.aspx?id=3040* (дата звернення 28.11.23р). – Назва з екрана.

Функція запуска обфускатора *Allatori*

```
apply plugin: 'com.google.gms.google-services'
@runAllatori
def runAllatori(variant) {
    copy {
        from "$projectDir/allatori.xml"
        into "$buildDir/intermediates/classes/"
        // old Gradle versions:
        // expand(classesRoot: variant.javaCompile.destinationDir,
        //        kotlinRoot: "${buildDir}/tmp/kotlin-classes/${variant.name}",
        //        androidJar:
"${android.sdkDirectory}/platforms/${android.compileSdkVersion}/android.jar",
        //        classpathJars: variant.javaCompile.classpath.getAsPath(),
        //        logFile: "allatori-log-${variant.name}.xml")
        // new Gradle versions:
        expand(classesRoot: variant.javaCompileProvider.get().destinationDir,
               kotlinRoot: "${buildDir}/tmp/kotlin-classes/${variant.name}",
               androidJar:
"${android.sdkDirectory}/platforms/${android.compileSdkVersion}/android.jar",
               classpathJars: variant.javaCompileProvider.get().classpath.getAsPath(),
               logFile: "allatori-log-${variant.name}.xml")
        rename('allatori.xml', "allatori-${variant.name}.xml")
    }

    // old Gradle versions:
    // new File("${variant.javaCompile.destinationDir}-obfuscated").deleteDir()
    // new Gradle versions:
    new File("${variant.javaCompileProvider.get().destinationDir}-obfuscated").deleteDir()

    // Kotlin support
    // new File("${buildDir}/tmp/kotlin-classes/${variant.name}-obfuscated").deleteDir()

    javaexec {
        main = 'com.allatori.Obfuscate'
        classpath = files("$rootDir/allatori/allatori.jar")
        args "$buildDir/intermediates/classes/allatori-${variant.name}.xml"
    }

    // old Gradle versions:
    // new File("${variant.javaCompile.destinationDir}").deleteDir()
    // new File("${variant.javaCompile.destinationDir}-obfuscated").renameTo(new
```

```
File("${variant.javaCompile.destinationDir}"))
 // new Gradle versions:
    new File("${variant.javaCompileProvider.get().destinationDir}").deleteDir()
    new File("${variant.javaCompileProvider.get().destinationDir}-
obfuscated").renameTo(new File("${variant.javaCompileProvider.get().destinationDir}"))

    // Kotlin support
    // new File("${buildDir}/tmp/kotlin-classes/${variant.name}").deleteDir()
    // new File("${buildDir}/tmp/kotlin-classes/${variant.name}-
obfuscated").renameTo(new File("${buildDir}/tmp/kotlin-classes/${variant.name}"))
}
```