Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Комплексна бакалаврська дипломна робота на тему:
«Засіб захищеної інтернет-телефонії. Частина 2. Модуль захищеного зв'язку»

08-20.БДР.015.00.000 ПЗ

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека
_____Олександр КОЗАК

Керівник: к. т. н., доцент, доцент каф. ЗІ
_____Юрій БАРИШЕВ
«16» червня 2023 р.

Рецензент: к. т. н., доцент, доцент каф. ПЗ
_____Наталя БАБЮК
«16» червня 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., професор
_____Володимир ЛУЖЕЦЬКИЙ
«16» червня 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., професор
_____Володимир ЛУЖЕЦЬКИЙ
«20» березня 2023 року

З А В Д А Н Н Я
НА КОМПЛЕКСНУ БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Козаку Олександру Михайловичу

1. Тема роботи: «Засіб захищеної інтернет-телефонії. Частина 2. Модуль захищеного зв'язку»
   керівник роботи: Баришев Юрій Володимирович, к. т. н., доцент, доцент кафедри ЗІ,
   затверджені наказом ректора ВНТУ від 20 березня 2023 року №67.
2. Строк подання студентом роботи 16 червня 2023 р.
3. Вихідні дані до роботи:
   - архітектура – клієнт-серверна;
   - спосіб шифрування – блоковий симетричний;
   - функціональні вимоги – підтримка аудіо та відео;
   - нефункціональні вимоги – платформонезалежний застосунок сумісний з Android, Windows.
4. Зміст текстової частини: Вступ. 1. Аналіз методів захисту зв'язку в інтернет-телефонії. 2. Архітектура модуля захищеного зв'язку. 3. Алгоритми роботи модуля захищеного зв'язку. 4. Тестування засобу захищеної інтернет-телефонії. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: порівняльний аналіз засобів інтернет-телефонії (плакат А4), аналіз методів шифрування (плакат А4), архітектура засобу захищеної інтернет-телефонії (плакат А4), структура модулю захищеного зв'язку (плакат А4), структура блоку шифрування трафіку (плакат А4),

узагальнений алгоритм роботи засобу (плакат А4), алгоритм блоку передавання даних (плакат А4), алгоритм роботи блоку криптографічного захисту (плакат А4), результати блокового тестування (плакат А4), результати інтеграційного тестування (плакат А4)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	11.04.23
2	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	08.05.23
3	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	15.05.23
4	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	10.06.23

7. Дата видачі завдання 20 березня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів комплексної бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	20.03.23 – 24.03.23	
2	Аналіз літературних джерел за напрямком комплексної бакалаврської дипломної роботи	27.03.23 – 07.04.23	
3	Розробка архітектури та алгоритму засобу	10.04.23 – 21.04.23	
4	Програмна реалізація та тестування	24.04.23 – 19.05.23	
5	Аналіз результатів тестування, висновки	22.05.23 – 24.05.23	
6	Оформлення пояснювальної записки	25.05.23 – 31.05.23	
7	Попередній захист та доопрацювання БДР	01.06.23 – 15.06.23	
8	Представлення БДР на захист	16.06.23 – 19.06.23	
9	Захист БДР	20.06.23 – 23.06.23	

Студент _____ Олександр КОЗАК

Керівник роботи _____ Юрій БАРИШЕВ

АНОТАЦІЯ

Комплексна бакалаврська дипломна робота складається з 82 сторінок формату А4, на яких є 29 рисунків, 3 таблиці, список використаних джерел містить 46 найменувань.

Комплексна бакалаврська дипломна робота присвячена розробці програмного засобу для реалізації захищеної інтернет-телефонії. В результаті аналізу існуючих аналогічних засобів, протоколів встановлення з'єднання для передачі медіа даних та методів шифрування трафіку обрано такий протокол для передачі медіа даних як WebRTC, метод симетричного блочного шифрування AES із довжиною ключа 256 біт та протокол генерації та розподілу ключів Diffie-Hellman. Розроблено архітектуру модуля захищеного зв'язку та наведено структуру її складових. Представлено алгоритми роботи модуля. Обґрунтовано вибір засобів розробки та описано основні семантичні одиниці програмного коду. Для доведення коректності прийнятих технічних рішень виконано тестування засобу та наведено його результати.

Ключові слова: захищена інтернет-телефонія, медіа дані, алгоритми шифрування, протокол передачі медіа даних, з'єднання.

ABSTRACT

The comprehensive bachelor thesis consists of 82 pages of A4 format, on which there are 29 figures, 3 tables, the list of used sources contains 46 items.

The comprehensive bachelor's thesis is devoted to the development of a software tool for the implementation of secure Internet telephony. As a result of the analysis of existing similar means, connection establishment protocols for media data transmission and traffic encryption methods, such media data transmission protocol as WebRTC, AES symmetric block encryption method with a key length of 256 bits, and Diffie-Hellman key generation and distribution protocol were selected. The architecture of the secure communication module is developed and the structure of its components is given. Algorithms of the module are presented. The choice of development tools is justified and the main semantic units of the software code are described. To prove the correctness of the adopted technical decisions, the tool was tested and its results are given.

Keywords: secure Internet telephony, media data, encryption algorithms, media data transfer protocol, connection.

ЗМІСТ

ВСТУП

За останні п’ять років потреба в віддаленій комунікації між людьми стрімко зросла у зв’язку виникнення епідемії COVID-19 та початком повномасштабної війни Росії проти України. Люди не мають змоги знаходитись в робочих офісах через загрозу повітряної тривоги або загрозу захворіти на вірус, через який виникла остання епідемія. Для зниження вірогідності виникнення цих загроз потрібно користуватись засобами інтернет-телефонії, адже саме вони можуть забезпечити спілкування між користувачами, які знаходяться у віддалених місцях де може ловити зв’язок, а саме, зникає потреба контактувати із зовнішнім середовищем та зникають описані вище загрози.

Однак, при використанні такої комунікації виникає проблема захисту даних, які передаються через засоби зв’язку та їх сервери, оскільки багато засобів зберігають персональну інформацію користувачів на власних серверах. Існує багато випадків викрадення інформації користувачів, яка проходить через сервери, або зберігається там. Цими серверами володіють компанії, які розробили відомі засоби, за допомогою яких реалізують зв’язок за допомогою інтернет-телефонії. Випадки несанкціонованого зберігання та виливу особистої та конфіденційної інформації користувачів з серверів є досить частою проблемою, оскільки доступ до інформації, яка зберігається на серверах, можуть отримати працівники компанії та сторонні шахраї. Тому, актуально розробляти засоби інтернет-телефонії, які дозволяють захистити персональні дані користувачів, шляхом використання методу пересилання медіа даних не через сервери, а саме від користувача до користувача.

Метою даної комплексної бакалаврської дипломної роботи є покращення захисту медіа даних користувачів під час користування засобами інтернет-телефонії, шляхом аналізу існуючих аналогів, протоколів та алгоритмів, які використовуються для встановлення з’єднання та захисту медіа даних, які передаються. Також, необхідно використати такі методи розповсюдження аудіо- та

відео-потоків, які не потребують третьої сторони для надсилання медіа даних між користувачами.

Для досягнення поставленої вище мети потрібно розв’язати такі визначені задачі:

- проаналізувати проблеми кібербезпеки відомих засобів інтернет-телефонії
- проаналізувати протоколи для передачі медіа даних
- проаналізувати алгоритми шифрування
- розробити узагальнену архітектуру засобу захищеної інтернет-телефонії
- розробити структуру основних модулів таких засобів
- розробити узагальнений алгоритм модуля захищеного зв’язку
- розробити алгоритми блоків шифрування даних та блоку встановлення захищеного з’єднання
- розробити програмний засіб для захищеної інтернет-телефонії
- провести тестування розробленого засобу.

Результати, отримані в комплексній бакалаврській дипломній роботі, доповідались на науково-практичних конференціях, за результатами яких було опубліковано одні тези, які були представлені на LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії [1].

Оригінальність результатів, отриманих в комплексній бакалаврській дипломній роботі, підтверджується двома отриманими свідоцтвами авторського права на такі твори (комп’ютерні програми) [2, 3].

1 АНАЛІЗ МЕТОДІВ ЗАХИСТУ ЗВ'ЯЗКУ В ІНТЕРНЕТ-ТЕЛЕФОНІЇ

1.1 Аналіз засобів інтернет-телефонії

Під час останніх п'яти років людського існування, відбулось досить багато різних подій, які призвели до диджиталізації методів зв'язку між багатьма людьми. Результатом даного явища, стала розробка нових та збільшення використання старих застосунків для комунікації способом інтернет-телефонії [4].

У сучасному світі існує чимало розроблених та реалізованих аналогів програмного засобу захищеного аудіо- та відеозв'язку. Відомі рішення можна розглянути залежно від методів захисту, технологій, що в них використовуються, та реалізації встановлення аудіо- та відеозв'язку. Для аналізу відомих реалізацій було розглянуто декілька популярних варіантів із таким набором функціональних можливостей: створення захищеного з'єднання між користувачами із можливістю передачі відео потоку та аудіопотоку даних.

Поміж цих засобів попри однакові функціональні можливості використовуються різні методи, а іноді, й підходи до захисту даних. Для початку було розглянуто, певно, найрозповсюдженіший програмний засіб захищеного аудіо та відеозв'язку WhatsApp від компанії Meta [5]. Цей програмний засіб є найстаршим поміж наведених нижче аналогів, що обумовило його високу популярність попри те, що він має такі функціональні можливості, які дуже розповсюджені у аналогів даного застосунку. Також варто зазначити, що через свій вік WhatsApp пройшов через велику кількість проблем з безпекою та захистом [6]. Цей засіб дозволяє таке: надсилання текстових, відео, аудіоповідомлень, встановлення аудіо- та відеозв'язку, можливість створення певних чатів та каналів для певних груп людей. Для здійснення гарного захисту від перехоплення повідомлень третіми особами під час передачі, WhatsApp реалізує наскрізне шифрування для всіх розмов [7]. WhatsApp забезпечує двоетапну перевірку для додаткової безпеки. З квітня 2016 з виходом оновлення версії 2.16.12 WhatsApp включив наскрізне шифрування (E2EE) для всіх користувачів на базі розробок протоколу Signal [8]. Цей протокол використовує комбінацію протоколу

узгодження ключів X3DH [9], алгоритму Double Ratchet [10] для безпечного керування ключами (рис. 1.1) та 256-бітного симетричного шифрування AES разом із обміном ключами за протоколом Діффі-Геллмана на основі еліптичних кривих (ECDH) для шифрування повідомлень [11].

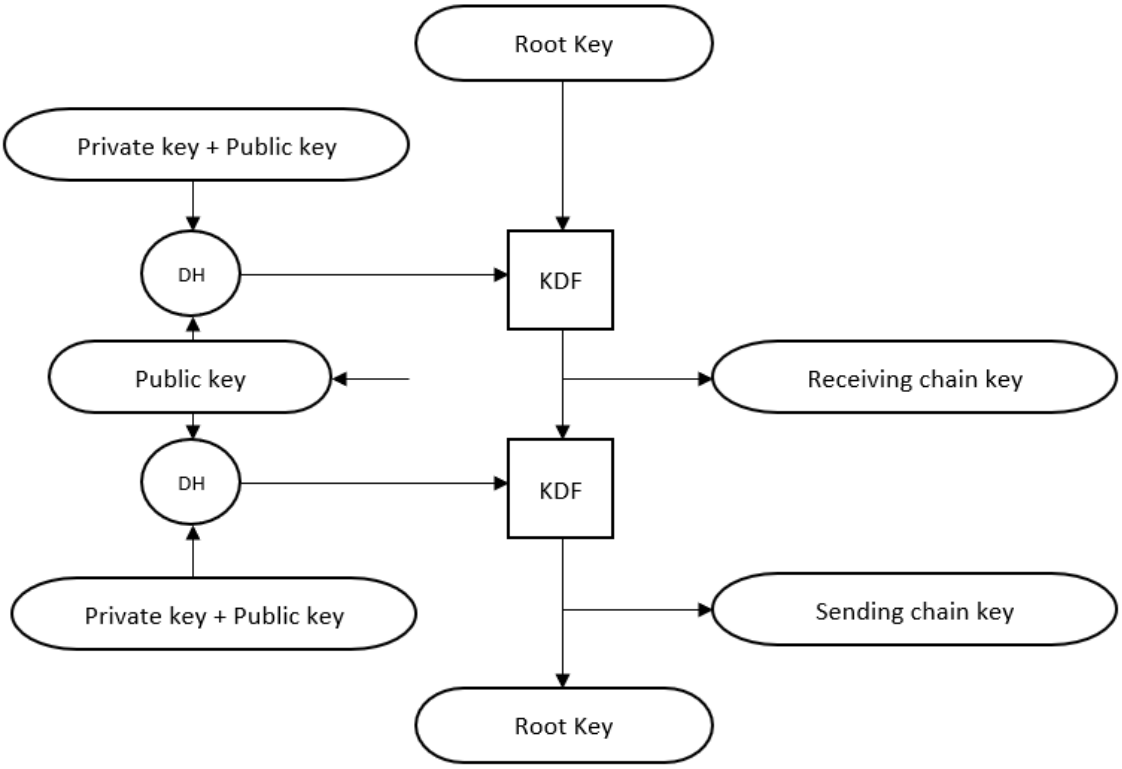


Рисунок 1.1 – Механізм роботи Double Ratchet алгоритму

Шифрування поширюється на всі типи повідомлень, а саме: аудіо- та відеоповідомлення, текстові та мультимедіа, яка включає відео та фото. Шифрування також доступне у групових чатах.

Наступним варіантом для аналізу був розглянутий програмний засіб Signal від розробника Signal Foundation [12]. Даний аналог має такі самі функціональні можливості, як і попередній, однак, саме Signal найбільше зосередився на конфіденційності даних користувача: застосунок не зберігає ніяких даних про дії користувача, лише час останнього візиту у форматі UNIX-часу та час створення аккаунту, також у форматі UNIX-часу (рис. 1.2).

Рисунок 1.2 – Облікові дані особистого облікового запису користувача Signal

На відміну від раніше переглянутого аналогу, Signal розроблений таким чином, щоб не зберігати ніякої особистої інформації користувача на серверах, це реалізовано за допомогою того, що використовується наскрізне шифрування не тільки для повідомлень і даних, що надсилаються, але і до всіх інших даних особистого кабінету, таких як контакти, соціальний граф, групові дані, інформація про стан груп, ім’я профілю, аватар профілю, відомості про геолокацію, історію пошуку та інше [13].

Ще одним варіантом засобу для інтернет-телефонії, є Skype [14]. На поточний момент, власниками даного програмного забезпечення є компанія Microsoft [15]. Розглянутий аналог, також використовує наскрізне шифрування для створення надійного захисту інформації, що передається під час комунікації між користувачами, але дане шифрування доступне лише для особистих чатів та голосових дзвінків. Під час комунікації іншими способами використовується блоковий алгоритм шифрування AES із довжиною ключа 256 біт. Для перевірки відкритих ключів користувачів використовується сертифікати ключів RSA довжиною 1536 та 2048 біт, перевірка здійснюється на сервері Skype [16]. Відмінністю даного засобу є можливість проводити конференції, з підтримкою відео- та аудіопотоків учасників, безкоштовно, в якій може брати участь до 100

осіб. Ще однією особливістю є перенесення коштів між обліковими записами за бажанням користувача [17], а також створення віртуальних номерів, які будуть закріплені за обліковим записом Skype [18]. За допомогою даного номеру можна приймати повідомлення та виклики з мобільного або стаціонарного телефону.

Наступним розглянутим засобом інтернет-телефонії є Zoom, який був розроблений та створений компанією Zoom Video Communications, Inc. [19]. Даний програмний продукт зорієнтований на інтеграцію із різними підприємствами для бізнесу. На відміну від попередньо розглянутого аналогу, конференції за допомогою Zoom є безкоштовними тільки 40 хвилин, обмеження стосовно учасників є таким самим у безкоштовній версії, але за допомогою платної версії, цей максимум розширюється до 1000 осіб. На відміну від попередньо розглянутих аналогів, Zoom використовує наскрізне шифрування для конференцій, але воно стає доступним лише тоді, коли всі учасники під'єднались до неї за допомогою застосунків Zoom Desktop Client, Mobile App або Zoom Rooms [20]. Ці застосунки доступні для таких операційних систем: Windows, MacOS, Linux, Android та iOS [21]. При застосуванні наскрізного шифрування ліміт учасників конференції зменшується до 200 осіб, навіть при наявності платної версії. Функцію наскрізного шифрування можна відключити, при цьому буде використовуватись блочний алгоритм AES із довжиною ключа 256 біт, а для транспортування відео та аудіо потоків (рис 1.3) учасників використовується алгоритм SRTP із тим самим блочним шифруванням [22].

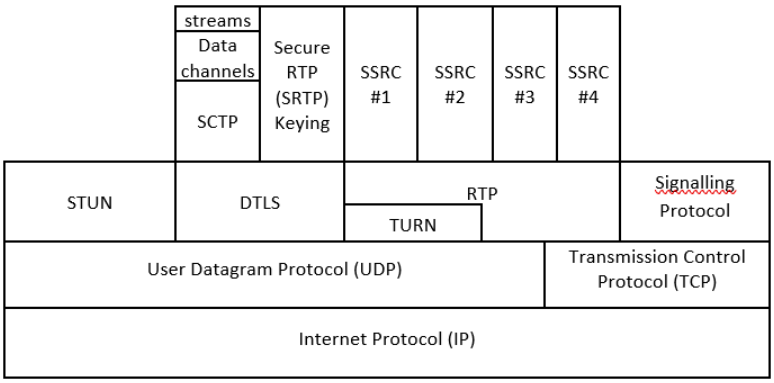


Рисунок 1.3 – Алгоритм передачі медіа-потоків із протоколом SRTP

Останнім проаналізованим засобом було проаналізовано Vonage [23], який спеціалізується на зв’язку за допомогою технології VoIP. Цей застосунок використовує шифрування, за допомогою протоколу TLS. Множину функціональних можливостей розширює можливість інтеграції із стільниковими телефонами. Vonage має досить велику інфраструктуру, що дозволяє підтримувати велику кількість користувачів та забезпечувати гарне відео та аудіо з’єднання між користувачами.

Отже, після проведеного аналізу відомих застосунків для інтернет-телефонії, був здійснений порівняльний аналіз (табл. 1.1) розглянутих вище програмних засобів за такими характеристиками: збереження анонімності користувачів (відсутність збереженої особистої інформації на серверах, які належать власнику розглянутого засобу), наявність відкритого коду, інтеграція засобу для бізнесу криптографічне шифрування, контроль доступу до конференцій, перевірка двоетапної автентифікації.

Таблиця 1.1 – Порівняльний аналіз проаналізованих засобів

Характеристика	WhatsApp	Signal	Skype	Zoom	Vonage
1	2	3	4	5	6
Збереження анонімності користувачів	-	+	-	-	+
Наявність відкритого коду	-	+	-	-	-
Перевірка двоетапної автентифікації	+	-	-	+	+
Інтеграція засобу для бізнесу	-	-	+	+	+
Контроль доступу до конференцій	-	-	-	+	+
Криптографічне шифрування	E2EE	E2EE	AES-256 або E2EE з обмеженнями	AES-256 або E2EE з обмеженнями	Шифрування TLS

Як видно з таблиці 1.1, усі розглянуті засоби мають свої переваги та недоліки. Signal та Vonage мають досить гарні характеристики щодо збереження конфіденційності користувачів. Signal не зберігає жодної особистої інформації облікових записів, але має недоліки стосовно методів автентифікації та контролем доступу до конференцій. Всі інші розглянуті засоби мають недоліки із збереження конфіденційності даних про користувача, але забезпечують гарне шифрування та функціональні можливості, які стосуються конференцій.

1.2 Аналіз методів передавання мультимедіа даних

Для передавання мультимедіа даних в рамках інтернет-телефонії було розроблено декілька методів, які були реалізовані у таких протоколах для передачі відео та аудіо потоків, а саме: HTTP, RTP, WebRTC та MPEG-DASH. Для більшого ознайомлення із цими протоколами був проведений аналіз кожного з них.

Для початку проаналізовано HTTP, даний варіант широко використовується для відображення веб-сторінок, але має багато розширення. Одне з них і дозволяє передавати мультимедіа дані користувача, а саме HLS, який був розроблений компанією Apple [24]. Архітектура цього протоколу складається з чотирьох частин (рис. 1.4). З самого початку потік збирається з пристроїв введення приладу, який використовується користувачем, далі зібрані дані надходять на серверну частину, де відбувається кодування, перетворення у визначений формат та сегментування на фрагменти та індексний файл. Після цього дані надходять на веб-сервер, який направляє у точку призначення. Отримувач надсилає запит на завантаження файлів та отримує результат у вигляді всіх файлів, збираючи їх в єдине ціле для створення безперервного потоку медіа-даних. Дане розширення підтримує можливість адаптивного стрімінгу, ця функціональна можливість дозволяє автоматично адаптувати якість переданих даних аналізуючи швидкість підключення до інтернету та можливості пристрою, яке використовує користувач. Недолік даного протоколу є те, що він не підтримує передавання мультимедіа потоку даних у режимі реального часу, оскільки створюється певна затримка при передаванні даних користувача через мережу. Основною перевагою даного варіанту є те, що він

підтримується багатьма сучасними браузерами, такими як: Chrome, Firefox, Safari та інші найпопулярніші варіанти схожим засобів. Даний протокол, в загальному, використовується на таких платформах, як YouTube, Netflix, Vimeo та інші відео-стрімінгові сервіси.

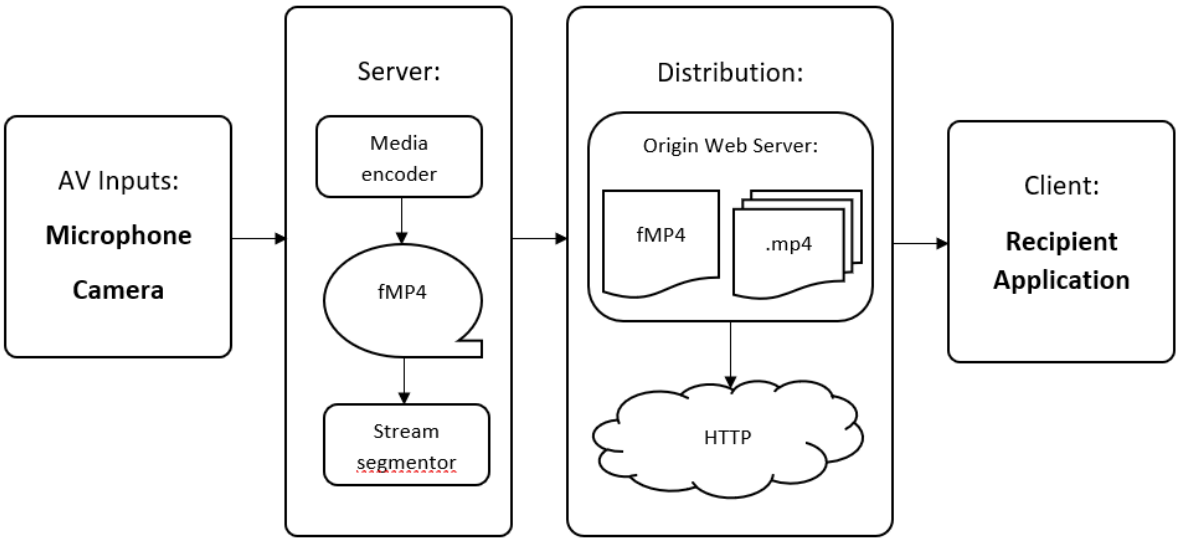


Рисунок 1.4 – Архітектура для передачі медіа потоків.

Далі проаналізовано протокол RTP, який розроблений Audio-Video Transport Working Group у відкритому міжнародному співтоваристві проектувальників та вчених (IETF) [25]. Особливістю даного протоколу є те, що він здійснює транслювання відео- та аудіоданих у режимі реального часу, отже забезпечує мінімальну затримку передавання медіа потоків, також, даний транспортний засіб передачі даних має бути доповнений протоколом керування, а саме RTCP, який додає певний набір функціональних можливостей, які вкрай важливі для передавання такого роду даних. Наприклад, вкрай важливі моніторинг якості обслуговування та обмін інформацією про учасників комунікації. RTP пакети мають забезпечення цілісності медіа потоку, оскільки мають у своєму вмісті порядковий номер та мітку часу, що дозволяє повторно збирати потік з пакетів, якщо вони надійшли в некоректній послідовності. Також, даний протокол підтримує передавання у груповому режимі, що дозволяє влаштовувати конференції із підтримкою аудіо- та відеопотоків від всіх учасників. Архітектуру взаємодії за даним протоколом проілюстровано на рисунку 1.5.

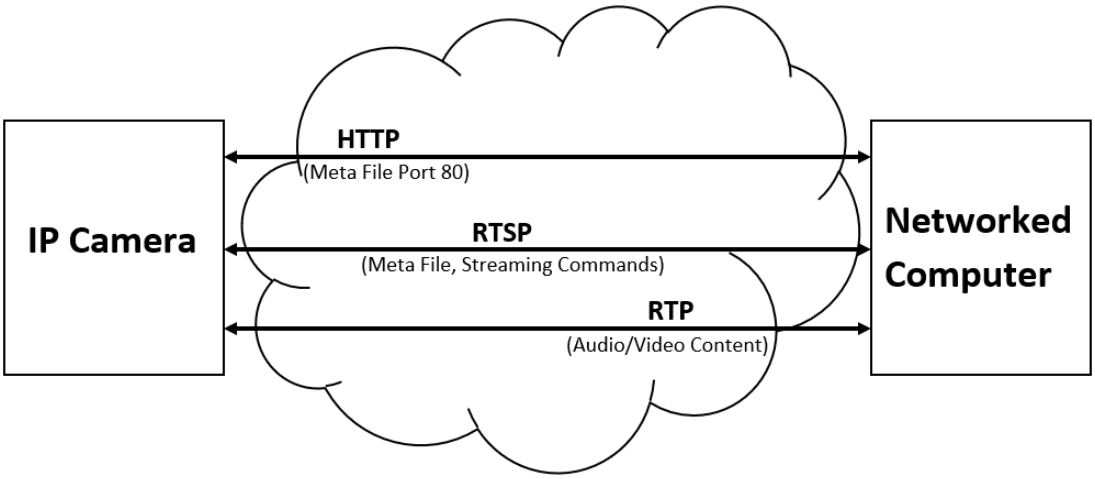


Рисунок 1.5 – Схема комунікації за протоколом RTP

Далі проаналізовано протокол WebRTC, розроблений компанією Google [26]. Даний протокол об’єднує можливості попередніх розглянутих аналогів, оскільки надає можливість передачі медіа-потоків у режимі реального часу та підтримує адаптивний стрімінг. WebRTC використовує два аудіокодеки та відеоформат VP8. Головна перевага даного протоколу міститься в тому, що для передачі аудіо- та відеопотоків між користувачами не особливо використовувати посередника, що збільшує таку властивість інформацію, як конфіденційність. Для коректної роботи WebRTC використовує декілька взаємопов’язаних програмних інтерфейсів (API) та різні протоколи, які працюють разом для налагоджування зв’язку та підтримання рівня певної якості методу комунікації за допомогою інтернет-телефонії. Серед таких протоколів є такі: ICE, STUN, NAT, TURN та SDP [27]. На їх основі, формуються програмні інтерфейси для встановлення зв’язку, пересилання даних, управління з’єднанням, отримання, надсилання та обробка певних подій, які пов’язані із діями учасників конференції. Оскільки стабільна версія вийшла лише 5 років тому, підтримка є не у багатьох браузерів, але цього достатньо для гарної якості зв’язку використовуючи цей протокол. Оскільки даний протокол не підтримує стійких протоколів із забезпеченням шифрування, його треба забезпечувати додатково, але переваги WebRTC є більш вагомі для створення засобів для встановлення гарного відео та аудіозв’язку, також для додаткової

безпеки є вбудований DTLS. Схему передавання медіа-потоків за допомогою цього методу проілюстровано на рисунку 1.6.

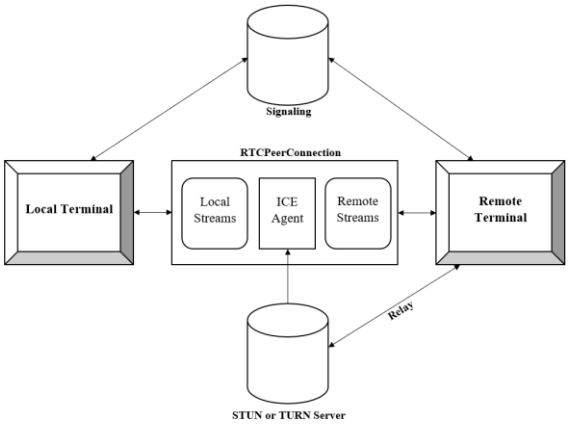


Рисунок 1.6 – Схема передачі медіа потоків WebRTC

Останнім проаналізовано протокол MPEG-DASH, який був розроблений Hewlett Packard Laboratories [28]. Цей аналог подібний до першого розглянутого протоколу, оскільки також використовує протокол HTTP та не дозволяє передавати медіа потоки користувачів у режимі реального часу, але має можливість реалізовувати адаптивний стрімінг відео- та аудіо-потоку даних користувача. Оскільки цей протокол є досить старим порівняно із попередніми розглянутими аналогами, він не підходить для реалізації теми та індивідуального завдання для цієї комплексної бакалаврської дипломної роботи.

Отже, після проведення аналізу відомих методів передачі мультимедіа даних проведено порівняльний аналіз (табл. 1.2) розглянутих вище протоколів за такими показниками їх якості: місця використання протоколу, можливість адаптивного стрімінгу, можливість передачі даних в режимі реального часу, підтримка браузерами, додаткові особливості використання.

Таблиця 1.2 – Порівняльний аналіз розглянутих методів

Характеристика	HTTP	RTP	WebRTC	MPEG-DASH
1	2	3	4	5
Використання	YouTube, Netflix	IP-трансляція медіа даних	Відеоконференції, онлайн-чати	Стрімінгові сервіси

Продовження таблиці 1.2

1	2	3	4	5
Адаптивний стрімінг	+	-	+	+
Підтримка реального часу	-	+	+	-
Підтримка браузерами	Всі популярні браузери	Залежить від пристрою користувача	Часткова підтримка популярних браузерів	Залежить від пристрою користувача
Особливості	Використання HLS	Використання RTSP	Безпосередня комунікація між кінцевими пристроями	Адаптивний стрімінг через HTTP
Адаптивний стрімінг	+	-	+	+

З аналізу табл. 1.2 випливає те, що використання WebRTC є досить влучним, оскільки це відносно новий протокол, який надає змогу реалізувати адаптивний стрімінг у режимі реального часу, також надає можливість встановити з’єднання безпосередньо між співрозмовниками. Інші протоколи мають певні недоліки, які проблематично проявлять себе під час практичної реалізації теми комплексної дипломної роботи.

1.3 Аналіз методів шифрування

Для здійснення стійкого захисту медіаданих є два основних методи, а саме: симетричне та асиметричне шифрування. Для забезпечення конфіденційності даних у даній сфері, зазвичай, використовують симетричне шифрування, але алгоритми асиметричного шифрування також необхідно проаналізувати. Для того, щоб збільшити стійкість шифрування, використовується протоколи ключової угоди. Досить розповсюджений алгоритм для розподілу ключів – Diffie-Hellman. Алгоритм шифрування та протокол ключової угоди будуть досить стійкою парою для збереження конфіденційності даних.

Найбільш розповсюджений алгоритм симетричного шифрування – це стандарт шифрування уряду США AES [29]. Даний алгоритм є доволі старим, але, все ж, широко використовується у сучасних протоколах та методах передачі даних та має досить гарну криптостійкість, навіть зважаючи на сучасні можливості зламу та можливі обчислювальні ресурси. AES має специфікації з довжиною ключа 128, 192 та 256 біт [30], для забезпечення максимального захисту, рекомендовано використовувати модифікацію саме з довжиною ключа 256 біт, оскільки це збільшує стійкість шифрування до зламу методом атаки перебором. Даний алгоритм заснований на мережі замін та перестановок (рис. 1.7).

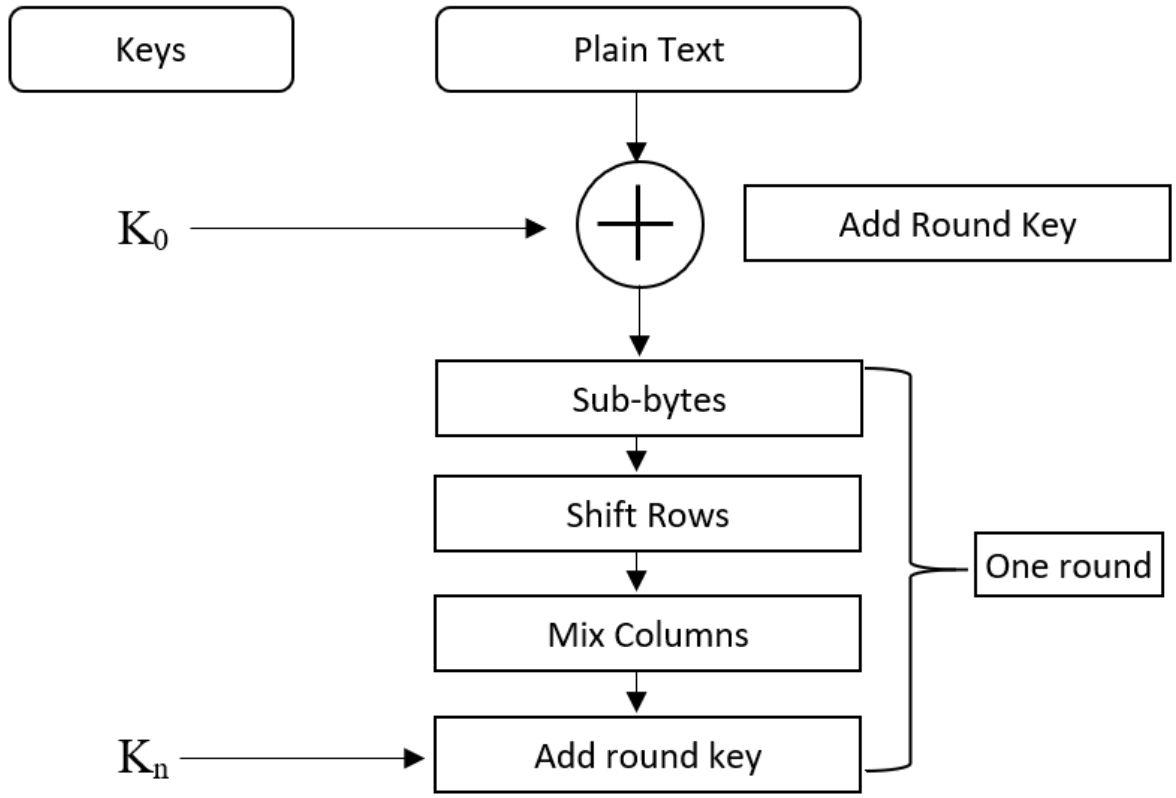


Рисунок 1.7 – Структура пристрою, що реалізує алгоритм роботи AES

Наступним алгоритмом для аналізу доцільно обрати шифр Калина, який є державним стандартом України [31]. Цей варіант має декілька режимів роботи, а саме: розмір блоку 128, 256 або 512 біт, розмір ключа 128, 256 або 512 біт, кількість раундів 10, 14 або 18. Розмір ключа визначається відповідно до розміру блока, а саме: для 128-бітного блоку підходить лише 128-бітні та 256-бітні ключі, для розміру блока 256 біт, підходить ключі розміром 256 та 512 біт, а для 512-бітного

блоку підходить лише 512-бітний ключ. Даний алгоритм спроектовано на основі SP-мережі. MDS-матриця виступає методом лінійного перетворення для цього шифру, для нелінійних перетворень, використовують чотири випадкові блоки для підстановки. Також, використовується попереднє та кінцеве забілювання за допомогою складання раундового ключа за модулем двійки в степені 64. Алгоритм роботи розглянутого шифру наведено на рисунку 1.8.

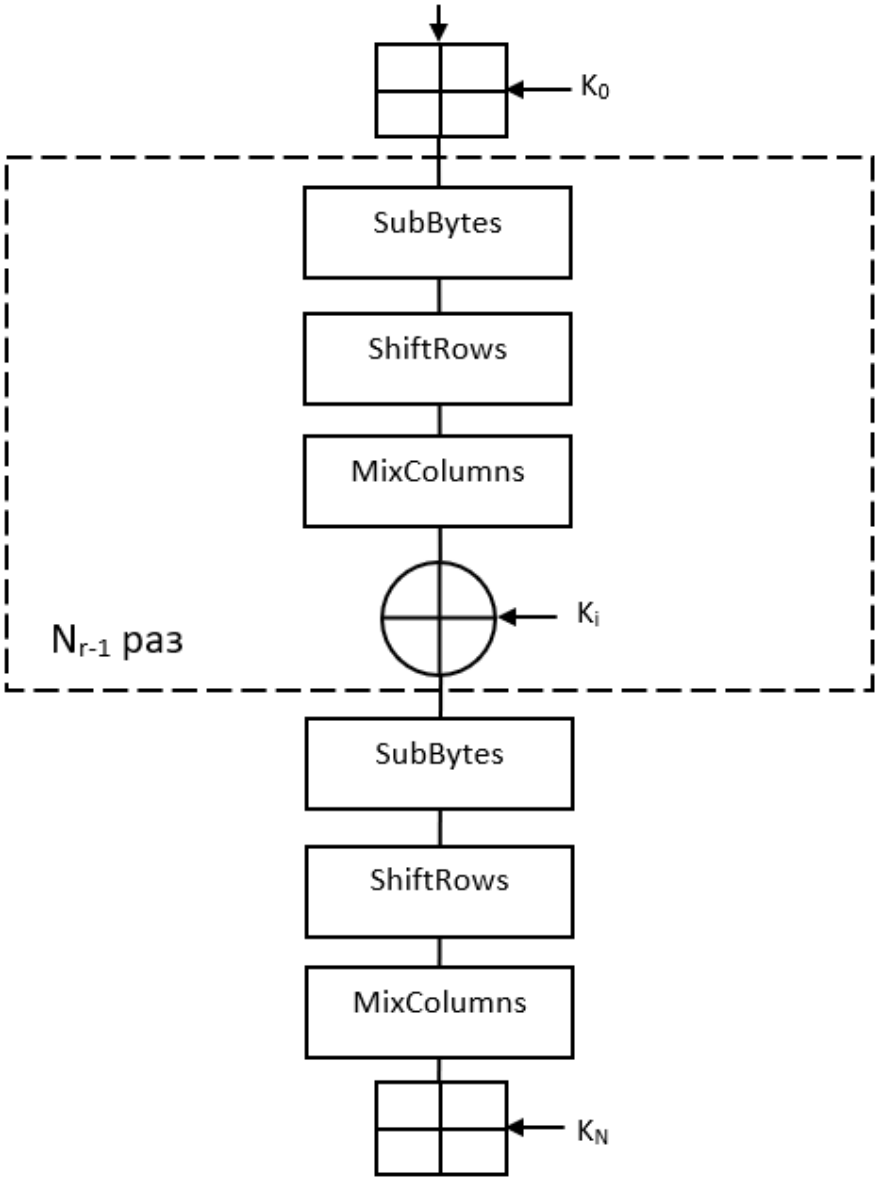


Рисунок 1.8 – Структура пристрою, що реалізує алгоритм Калина

Далі було проаналізовано алгоритм на основі мережі Фейстеля, а саме Camellia, який був розроблений Mitsubishi, NTT [32]. Даний шифр має такі режими роботи: розмір ключа 128, 192 або 256 біт, а розмір блоку 128 біт. Режими роботи

цього алгоритму схожі до раніше розглянутого шифру AES, а саме: електронна кодова книжна, кодовий зв'язок блоків, лічильник, шифрування з зворотним зв'язком по шифротексту та вихідний звортній зв'язок. Схема роботи Camellia зображено на рисунку 1.9.

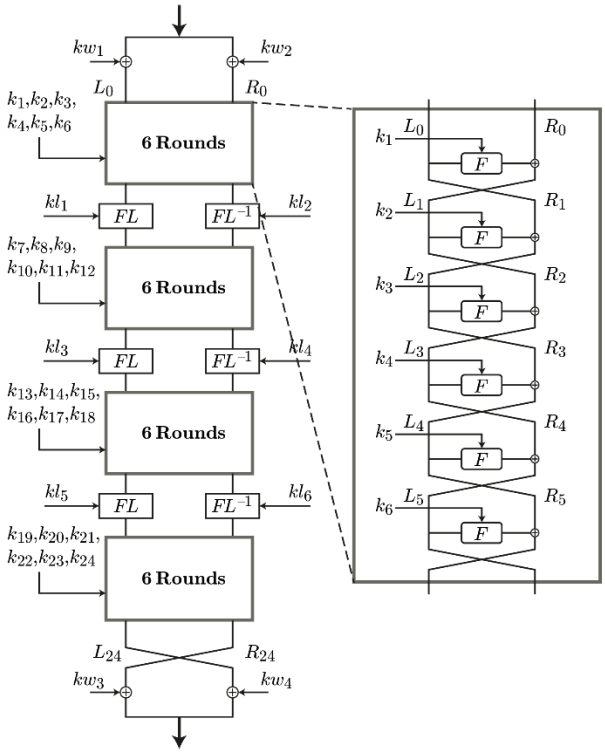


Рисунок 1.9 – Структура пристрою, що реалізує алгоритм Camellia

Ще один представник алгоритму на основі мережі Фейстеля, який було проаналізовано – це DES [33]. Цей шифр є досить застарілим, але у свій час був досить криптостійким та вніс істотний внесок у розвиток криптографії. DES має досить малий розмір блоку, порівняно із попередньо розглянутими, а саме: 64 біти, ключ має такий самий розмір, але при надходженні перетворюється у 54-бітовий, відкинуті 8 бітів використовуються для контролю парності, тому фактично використовується тільки 54 біт вхідного ключа.

При використані симетричного алгоритму шифрування, необхідно забезпечити захищене розподілення ключів між користувачами. Для цього завдання був проаналізований алгоритм розподілу ключів Diffie-Helman (рис 1.10) [34]. Оскільки криптостійкість засновується на обчисленні дискретного алгоритму, то атаки, які відомі та розповсюджені на сьогоднішній день, є досить складними та

вимагають значних обчислювальних ресурсів [35]. Для якісної роботи, необхідно коректно обирати параметри, які використовується даним алгоритмом, тому потрібно відповідально підходити до цієї задачі. При відповідальному виконанні усіх потрібних дій для забезпечення високих показників криптостійкості, пару алгоритмів AES-256 для шифрування та Diffie-Hellman для розподілу секретних ключів можна вважати надійним.

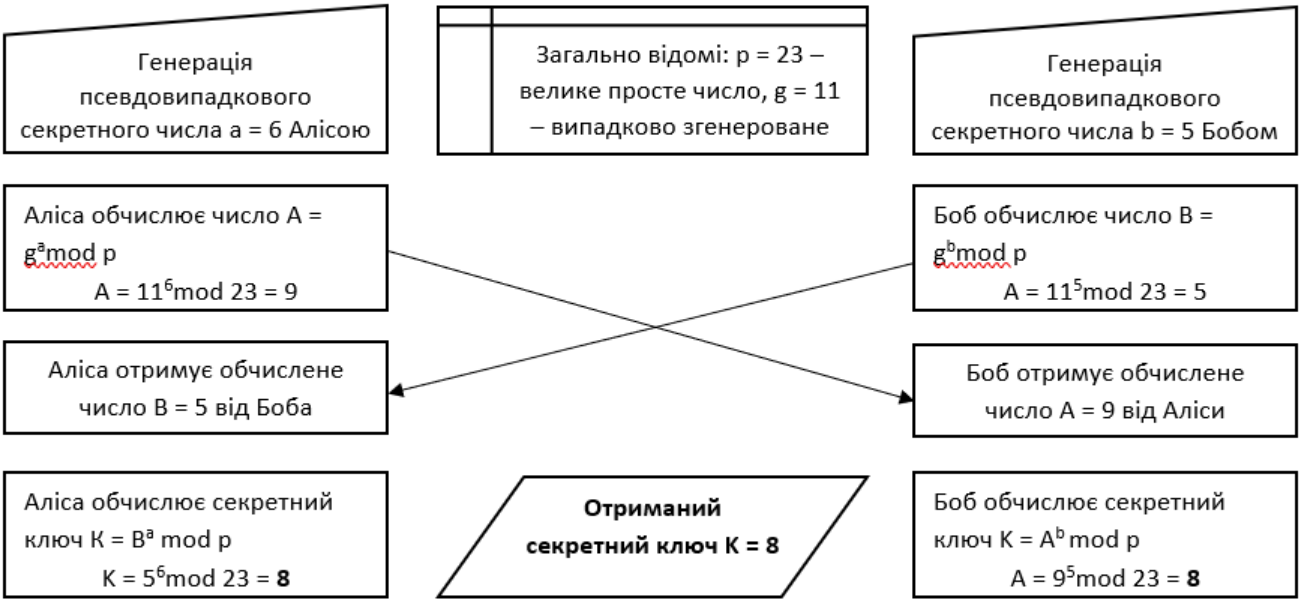


Рисунок 1.10 – Протокол Diffie-Hellman

Отже, після проведеного аналізу було зроблено порівняння (табл. 1.3) розглянутих методів шифрування за певними характеристиками, а саме: метод використання, тип алгоритму, розмір ключа, криптографічна операція, криптостійкість, обчислювальна складність, продуктивність та використання в протоколах.

Таблиця 1.3 – Порівняльний аналіз розглянутих методів шифрування

Характеристика	AES	Калина	Camellia	DES
1	2	3	4	5
Розмір блоку	128	128	128	64
Розмір ключа	128, 192, 256	128, 256, 512	128, 192, 256	56
Режими роботи	ECB, CBC, CTR, CFB, OFB	ECB, CTR, CFB	ECB, CBC, CTR, CFB, OFB	ECB, CBC, CFB

Продовження таблиці 1.3

1	2	3	4	5
Рік створення	2001	2015	2000	1975
Швидкість шифрування	Висока	Помірна	Висока	Повільна
Безпека шифрування	Висока (стандарт США)	Висока (стандарт України)	Висока	Низька

З таблиці 1.3 випливає, що DES однозначно застарілий та нерекомендований до застосування алгоритм. Найкращими варіантами серед розглянутих будуть алгоритми AES та Калина, але перший варіант є більш швидким, а цей показник є досить важливим при передачі медіа даних у вигляді інтернет-телефонії. У характеристиці безпеки AES поступається стандарту України, цей алгоритм має можливість використання більшого розміру ключа та більшу кількість раундів, що збільшує його криптостійкість.

1.4 Постановка задачі

Після проведеного аналізу існуючих засобів інтернет-телефонії випливає, що найбільш частою проблемою таких додатків є нехтування збереженням анонімності користувача, оскільки всі розглянуті засоби, окрім Signal, зберігають конфіденціальну інформацію на серверах, що породжує ризик, щодо виникнення інциденту інформаційної безпеки, а саме: вилив конфіденційної інформації користувачів у загальний доступ. Також, важливо досягти стійкого захисту інформації, яка передається під час конференції, або приватної розмови користувачів, з цього випливає задача, покращити існуючи методи шифрування, шляхом комбінації декількох розглянутих алгоритмів. Для реалізації стійкого захисту, було проведено аналіз алгоритмів блочного симетричного шифрування. З даного аналізу випливає те, що AES-256 для шифрування та Diffie-Hellman для обміну ключами створюють досить стійку пару алгоритмів для реалізації

криптостійкого захисту даних користувачів. Також, важливо зазначити, що запровадження гарного протоколу для передавання медіа даних під час комунікації також досить важлива задача. Для цього було проаналізовано методи для передачі медіа-даних за використанням різних протоколів, які мають основні функціональні можливості. На основі даного аналізу було зроблено порівняння, в результаті якого обрано основні можливості, які має забезпечувати метод передачі аудіо- та відео-потоків, а саме: адаптивний стрімінг в режимі реального часу.

Отже, мета цієї комплексної бакалаврської дипломної роботи в тому, щоб розробити модуль програмного застосунку для реалізації криптографічно-стійкого з'єднання між користувачами з підтримкою передачі аудіо- та відео-потоків з адаптивністю та в режимі реального часу.

1.5 Висновки з розділу

У першому розділі був проведений аналіз методів інтернет-телефонії, на основі якого був зроблений порівняльний аналіз, з якого випливають недоліки та переваги кожного з розглянутих додатків. Було проаналізовано методи передачі медіа даних, а саме: визначено існуючи протоколи для передачі відео- та аудіо-потоків із можливостями адаптивного стрімінгу у режимі реального часу, найбільш гарний варіант виявився WebRTC за всіма критеріями. На основі цього аналізу, було зроблено порівняння та виявлено позитивні та негативні сторони кожного з розглянутих протоколів. На останок було проведено аналіз методів шифрування медіа даних, на основі якого було визначено переваги та недоліки використання кожного з розглянутих методів, за допомогою порівняльного аналізу. Пара алгоритму шифрування AES та протоколу обміну ключів Diffie-Hellman визначено найбільш стійким та таким, що пасує для забезпечення достатньої продуктивності застосунку.

2 АРХІТЕКТУРА МОДУЛЯ ЗАХИЩЕНОГО ЗВ’ЯЗКУ

2.1 Узагальнена архітектура засобу захищеної інтернет-телефонії

Кожний програмний застосунок містить певну архітектуру, розгляд якої досить важливий для забезпечення інформаційної безпеки даних, які обробляються у застосунку. Для початку було розглянуто узагальнену архітектуру засобу інтернет-телефонії (рис. 2.1).

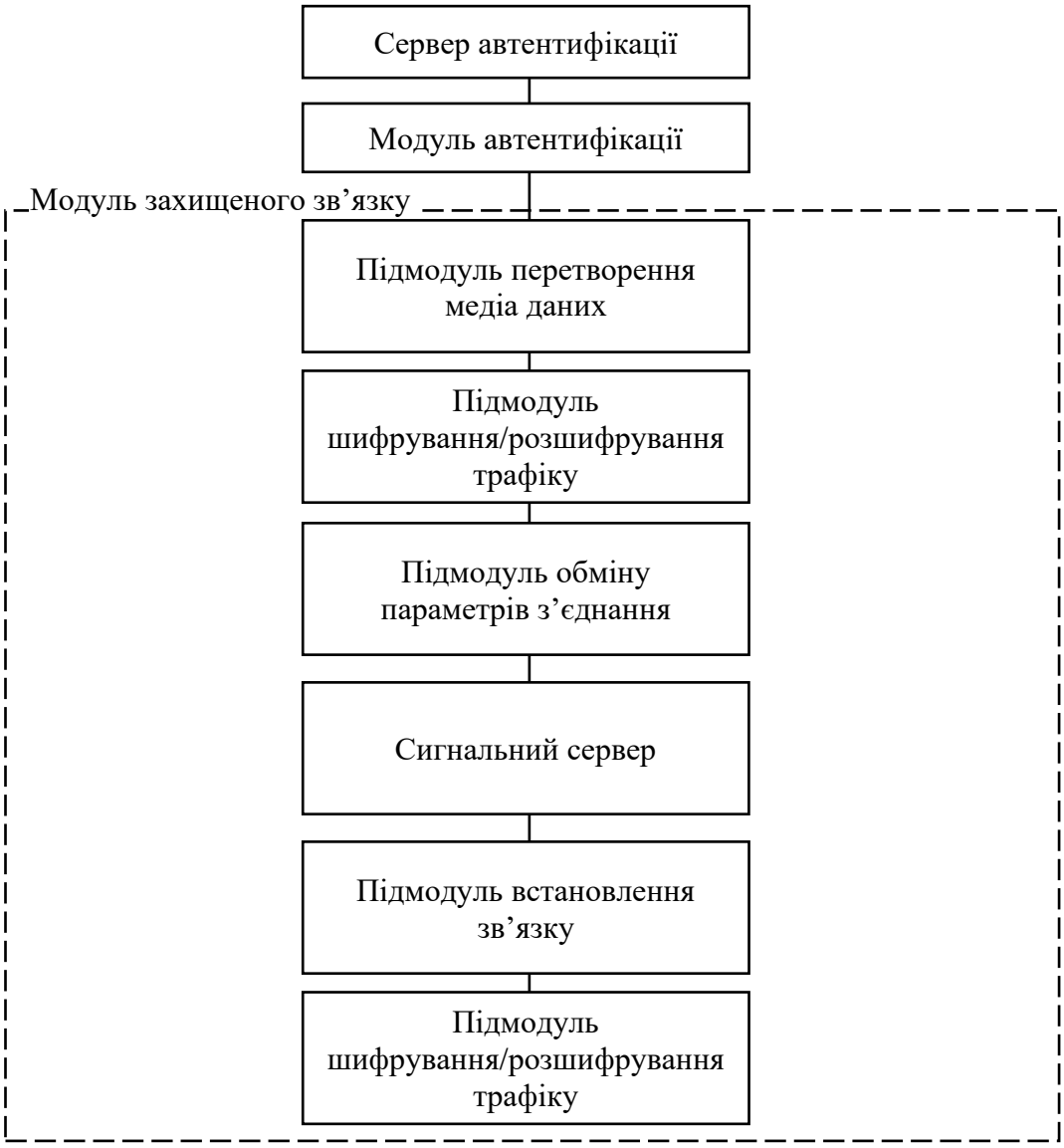


Рисунок 2.1 – Схема загальної архітектури засобу інтернет-телефонії

Цей програмний застосунок містить такі блоки: модуль автентифікації та модуль захищеного зв’язку, які працюють із серверами, які забезпечують API для

взаємодії цих частин та пересилання даних між ними. Модуль захищеного зв’язку поділяється на декілька підмодулів, які працюють послідовно з локальними та віддаленими медіа даними користувачів. Підмодуль перетворення медіа даних відповідає за зчитування та перетворення відео- та аудіо-потоку з пристроїв введення у потрібний формат для початку шифрування та пересилання отримувачу. Після цього, зчитані медіа дані надходять на вхід до підмодулю шифрування трафіку, який здійснює криптографічні перетворення для забезпечення конфіденційності під час передачі даних мережею. Паралельно з цим працює підмодуль обміну параметрів з’єднання, який формує та відправляє SDP-пакети на сигнальний сервер. Після цього починає працювати підмодуль встановлення зв’язку. Він відповідає за встановлення комунікації між користувачами та пересилання локальних та отримання віддалених медіа даних для розшифрування. Далі знов повторюється підмодуль шифрування трафіку, але він вже розшифровує аудіо- та відео-потоки даних віддалених користувачів. Отже, доцільно перейти до більш детального розгляду структури складових.

2.2 Структура модуля захищеного зв’язку

Модуль захищеного зв’язку реалізує збирання медіа даних користувачів, їх шифрування, створення комунікаційного з’єднання, відправку та отримання відео- та аудіо-потоків іншим користувачам. Ці функціональні можливості реалізуються за допомогою протоколу WebRTC [26] та симетричного блочного алгоритму шифрування AES-256 [29], секретні ключі для якого розповсюджуються між користувачами за допомогою алгоритму розподілу ключів Diffie-Hellman [34]. Дані, якими оперує даний модуль, передається до інших користувачів за допомогою протоколу SRTP, який є вбудованим в WebRTC. Для початку з’єднання, користувач формує SDP-пакет. Цей пакет містить всю необхідну інформацію про параметри з’єднання, а саме: які потоки даних будуть передаватись аудіо- або відео-потік, які кодеки при цьому будуть використовуватись, адресу користувача тощо. Для передачі цього пакету і використовується сигнальний сервер, який побудований на сокетах. Віддалений користувач отримує SDP-пакет та формує на

його основі власний, таким чином учасники конференції знають потрібну інформацію для передачі та отримання медіа потоків даних. Після цього формується RTCPeerConnection по якому іде передача аудіо- та відео-потоків між учасниками конференції. Детальна структура модуля захищеного зв'язку розбита на дві менші, які наведені на рисунках 2.2 та 2.3.

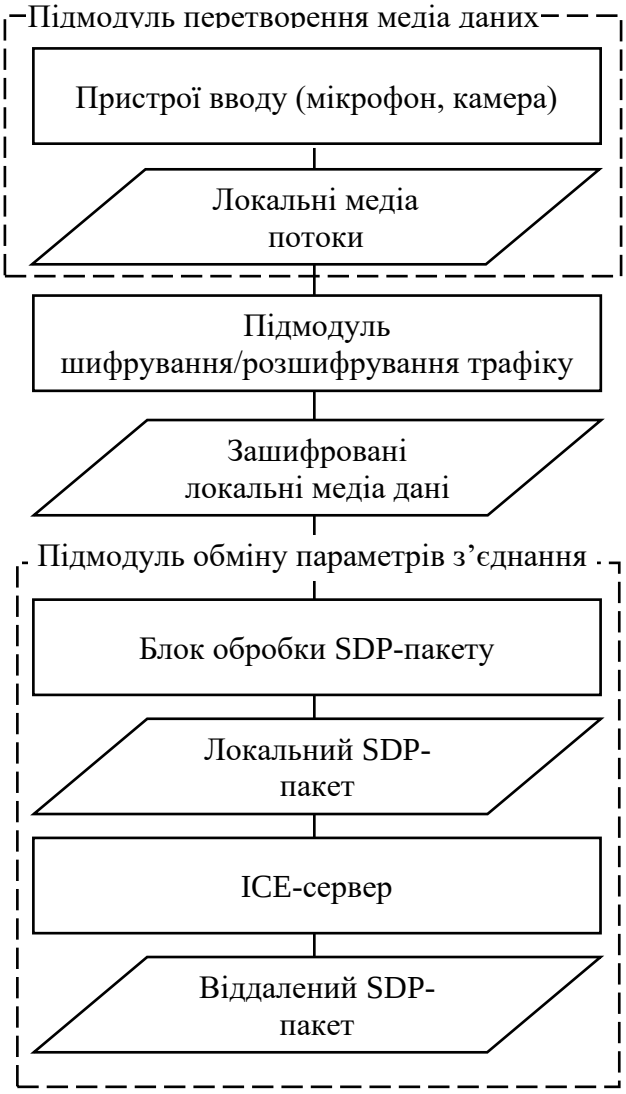


Рисунок 2.2 – Структура модуля збирання інформації для з'єднання

Рисунок 2.3 – Структура модуля встановлення зв'язку та обміну даних

Після того, як було розглянуто структуру модуля, розробка якого є темою комплексної дипломної роботи, необхідно розглянути структури певних важливих підмодулів, а також обґрунтувати вибір протоколів для встановлення з’єднання, алгоритмів шифрування та протоколу розподілу ключів.

2.3 Обґрунтування вибору криптографічних алгоритмів

Для використання протоколу WebRTC для встановлення комунікації між користувачами, необхідно обґрунтувати такий вибір. Даний протокол має досить багато переваг, серед яких [36]: підтримка адаптивного стрімінгу, тобто автоматичне налаштування якості передачі медіа потоків в залежності від можливостей пристроїв, налаштування з’єднання та швидкості інтернет-з’єднання, можливість стрімінгу у режимі реального часу, а саме забезпечення низької затримки для досягнення максимально можливої швидкості передачі аудіо- та відео-потоків. Протокол забезпечує стійке шифрування даних, під час передачі між сторонами, оскільки використовує вбудовані протоколи для передачі даних, а саме SRTP та DTLS. Також, WebRTC використовує тільки протокол HTTPS на прикладному рівні, при користуванні веб-версії застосунку. Забезпечує можливістю демонструвати захоплення контенту користувачем, наприклад робочого столу комп’ютера, що досить важливо у сучасних конференціях. Досить гарною перевагою є те, що це кросплатформений протокол, тому засіб, який розроблений для операційної системи Android зможе під’єднатись до конференції, яку створив користувач засобу для Windows. Також, це проект з відкритим кодом, тому його можна використовувати без усіляких бібліотек або обгорток, для забезпечення безпеки під час розробки та вибору необхідних залежностей. Також, даний протокол використовує досить сучасні кодеки для ущільнення аудіо- та відео-даних, які забезпечують швидке кодування, високий рівень ущільнення, підтримку змінного бітрейту та забезпечення перевірки щодо втрати кадрів. Недоліком даного протоколу є те, що WebRTC визначає реальні IP-адреси користувачів для встановлення якісного з’єднання, але від цього страждає анонімність користувачів, проте, використання TURN-серверу [37] забезпечить

приховування реальної IP-адреси. Також, підсилення конфіденційності даних користувача стане використання симетричного блочного алгоритму AES-256 [29] у парі із протоколом розподілу ключів Diffie-Hellman [34]. Даний протокол шифрування є національним стандартом США та забезпечує стійкість алгоритму при використанні довжини ключа 256 біт. Даний алгоритм відповідає вимогам шифрування, що використовуються в урядових та військових організаціях. Процес шифрування та розшифрування відбувається досить швидко та співвідношення потрібних обчислювальних ресурсів для цього є досить гарним. AES-256 є досить розповсюдженим алгоритмом, що робить його підтримку та додавання до архітектури програмного засобу досить легким завданням, оскільки вже існує досить багато реалізацій цього шифрування для багатьох сучасних мов програмування. Недоліком є те, що це симетричний алгоритм, отже вимагає використання одного і того ж секретного ключа для шифрування та розшифрування, виникає проблема розповсюдження між сторонами даного ключа. Дана проблема вирішується з використанням протоколу для створення спільного секрету Diffie-Hellman, який має такі переваги: за допомогою даного алгоритму можна створити захищені ключі та розділити між користувачами без їх пересилання, що підсилює безпеку секретного ключа, що і потрібно для вирішення проблеми, яка виникла. Також, є можливість розширення даного протоколу для групових комунікацій, наприклад конференцій, в даному випадку багато сторін можуть обмінюватись секретними ключами з максимально можливим збереженням конфіденційності створеного ключа. При всіх перевагах даного протоколу, великих обчислювальних ресурсів він не потребує, оскільки криптографічні операції є не досить складними та відбуваються досить швидко. Недоліки протоколу Diffie-Hellman такі: має бути довіра до початкових даних, які формуються на кожній початковій стороні. Ці дані використовуються для формування та передачі секретного ключа, тому, якщо вони будуть зкомпроментовані, то і весь механізм розповсюдження секрету зазнає невдачі при збереженні конфіденційності ключів. Але, даний недолік вирішується модулем

автентифікації, який підтверджує кожну сторону комунікації, що породжує довіру до кожної із них.

2.4 Структура підмодуля шифрування

Цей підмодуль досить важливий для підвищення криптостійкості захисту медіа даних користувачів. Вхідними даними для цього підмодуля є аудіо- та відео-потоки та параметри зв’язку, а саме бітрейт з яким необхідно транслювати медіа потоки, які саме треба передавати медіа дані, інформація про мережеве з’єднання кожного з користувачів та інформація про можливості пристроїв користувачів, які використовуються під час зв’язку. Ці дані є вихідними з попереднього підмодуля, а саме перетворення медіа даних. Для шифрування використовується блочний симетричний алгоритм AES з довжиною ключа 256 біт. Для генерації спільного секрету та розповсюдження його між всіма користувачами використовується протокол Diffie-Hellman, структура описаного вище підмодуля шифрування наведено на рисунку 2.4.

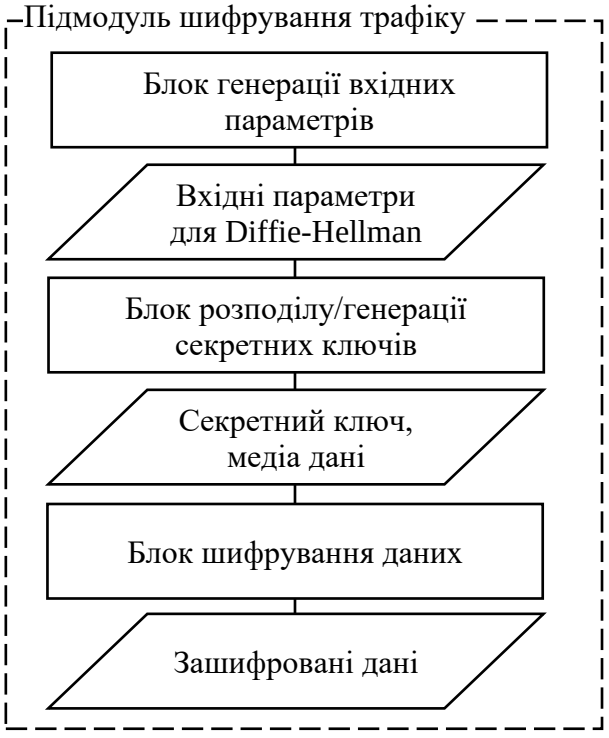


Рисунок 2.4 – Структура підмодуля шифрування

Блок генерації вхідних параметрів відповідає за забезпечення стійких параметрів для протоколу розподілу секретних ключів, такі як: два відкритих простих великих числа, одне з яких виступає модулем при обчислені секретного ключа, а інше виступає числом, яке підноситься до степені, секретний ключ, який генерується на кожній стороні та виступає у вигляді степені до якої підноситься раніше згенероване велике просте число. Після генерації усіх необхідних параметрів, вони надходять у блок розподілу/генерації секретних ключів за протоколом Diffie-Hellman. Цей блок відповідає за криптографічні перетворення вхідних даних та надсилання їх іншим користувачам. Вихідними даними і є секретний ключ, який було використано для шифрування медіа потоків. Далі йде сам блок шифрування, вхідні дані якого є медіа дані користувача та секретний ключ. Він відповідає за проведення криптографічних операцій для підвищення конфіденційності інформації, яка передається. Вихідними даними цього блоку є зашифровані аудіо- та відео-потоки та сформовані SDP-пакети користувачів, які використовуються для обміну параметрами з'єднання для встановлення якісного з'єднання для передавання медіа даних.

2.5 Структура підмодуля встановлення зв'язку

Цей підмодуль відповідає за встановлення DTLS каналу, через який передаються зашифровані медіа дані, які є вхідними даними разом із SDP-пакетом, який містить всі необхідні параметри для встановлення зв'язку між користувачами. З'єднання встановлюється за допомогою адреси віддаленого користувача, який отримується з SDP-пакета. Цей підмодуль реалізує встановлення зв'язку, відправлення локальних та отримання віддалених аудіо- та відео-потоків. Далі Сигнальний сервер забезпечує виявлення та налаштування мережевих шляхів між клієнтами. Це допомагає уникнути проблем з мережевою конфігурацією і забезпечити найкращий шлях для передачі медіа даних між учасниками зв'язку. Структура описаного підмодуля наведено на рисунку 2.5.

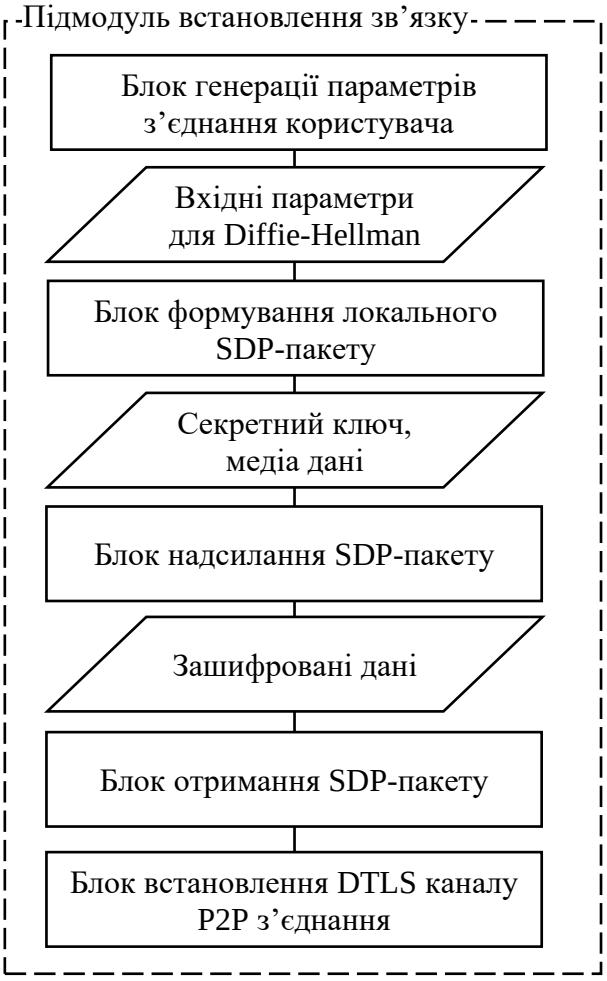


Рисунок 2.5 – Структура підмодуля встановлення зв’язку

Після встановлення безпечного DTLS каналу, створюються RTCP канали для передачі відео- та аудіо-потоків. Основною перевагою протоколу WebRTC це використання саме таких каналів, які забезпечують передачу медіа даних в режимі реального часу із можливістю адаптації якості передачі в залежності від мережевих можливостей. Тобто, ці протоколи реалізують дуже малу затримку при передачі медіа даних, що дозволяє імітувати передачу аудіо- та відео-потоків у режимі реального часу, а адаптивність дозволяє змінювати бітрейт та параметри передачі медіа даних користувачів під час розмови, що надає велику перевагу при використані цих протоколів передавання інформації.

2.6 Висновки з розділу

У другому розділі було розроблено та проаналізовано узагальнену архітектуру засобу захищеної інтернет-телефонії, що дозволило здійснити керування складністю процесу проектування шляхом декомпозиції задачі. В наслідок цієї декомпозиції було виділено модулі, які співпрацюють для забезпечення стійкого захисту. Проведено детальний аналіз модуля захищеного зв’язку під час якого виділено підмодулі, які обробляють медіа дані користувачів та забезпечують шифрування. Для визначення найкращого рішення щодо реалізації криптографічного захисту були проаналізовані криптографічні алгоритми для шифрування та протоколи, які забезпечують з’єднання між користувачами. Наведено обґрунтування вибору саме цих криптографічних алгоритмів, продемонструвавши переваги та недоліки кожного з них. Детально розглянуто структуру основних підмодулів, а саме: підмодуль шифрування трафіку та підмодуль встановлення зв’язку. В результаті було розроблено всю необхідну архітектуру та структури модулів та підмодулів для розробки алгоритмів роботи та подальшу розробку практичної реалізації засобу захищеної інтернет-телефонії. На основі досліджень, які були зроблені у цьому розділі, необхідно розробити алгоритми роботи усіх основних модулів та підмодулів засобу захищеної інтернет-телефонії та узагальнений алгоритм роботи програмного застосунку.

3 АЛГОРИТМИ РОБОТИ МОДУЛЯ ЗАХИЩЕНОГО ЗВ’ЯЗКУ

3.1 Узагальнений алгоритм роботи засобу захищеної інтернет-телефонії

Для розробки узагальненого алгоритму роботи засобу захищеної інтернет-телефонії було використано напрацювання з попереднього розділу. Початок роботи засобу починається із автентифікації користувачів сервером, який за це відповідає. Також, перед встановленням захищеного з’єднання, користувачам необхідно пройти двосторонню автентифікацію для підтвердження користувачів один перед одним. Якщо результат попередньої операції позитивний, тоді починається робота модуля встановлення захищеного зв’язку. Схема узагальненого алгоритму роботи засобу захищеної інтернет-телефонії було проілюстровано за допомогою двох схем, наведених на рисунку 3.1 та 3.2.

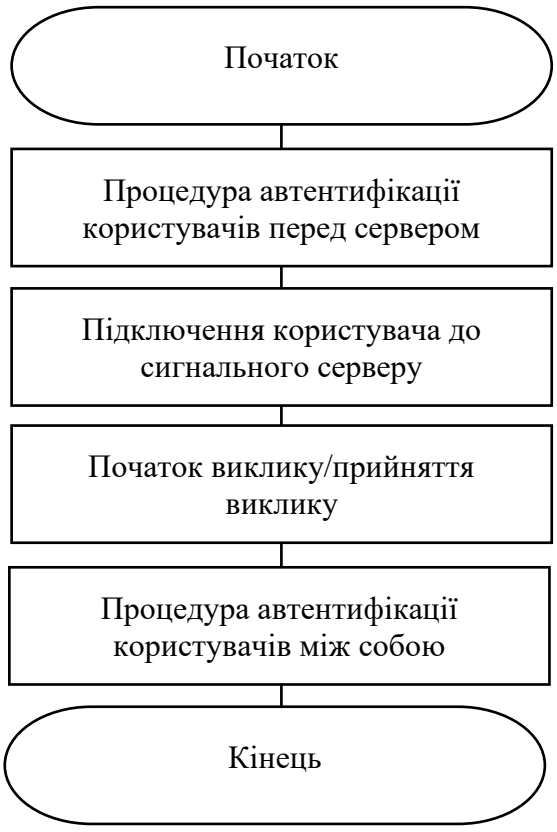


Рисунок 3.1 – Схема узагальненого алгоритму роботи модуля автентифікації

Рисунок 3.2 – Схема узагальненого алгоритму роботи модуля захищеного зв’язку

Спочатку користувач, який виступає ініціатором зв’язку, формує свій SDP-пакет та надсилає його із подією створення кімнати для конференції на сигнальний

сервер. Після цього формуються аудіо- та відео-потоки для надсилання віддаленим користувачам. Після цього віддалений користувач має під'єднатись до створеної кімнати за допомогою спеціального ідентифікатора, він також надсилає сформований SDP-пакет та формує свої аудіо- та відео-потоки. Після підключення користувачів до однієї кімнати та обміну SDP-пакетів формується P2P тип з'єднання. Далі виконується генерація та розподіл секретних ключів між користувачами за допомогою протоколу Diffie-Hellman. Після отримання секретного ключа, користувачі шифрують медіа дані та надсилають їх іншим користувачам. При отримані аудіо- та відео-потоків відбувається розшифрування та виведення користувачу віддалених медіа даних.

Для використання протоколу WebRTC було використано бібліотеку flutter_webrtc [38], яка забезпечує зручні класи для налаштування та ініціалізації RTCPeer з'єднання. Для з'єднання із сигнальним сервером використовується бібліотека socket.io-client [39]. Цей модуль забезпечує такі функціональні можливості, як підключення до потрібного сокету, який розміщений на сервері, надсилання та опрацювання певних подій для встановлення зв'язку.

3.2 Алгоритм роботи блоку передавання даних

Розробка алгоритму роботи блоку передавання даних є досить важливою частиною. Даний блок починає свою роботу після підмодуля шифрування даних. Для отримання аудіо- та відео-потоків, необхідно створити дві змінні, які належать класу, який відповідає за зберігання, конвертацію та всі інші маніпуляції над аудіо- та відео-потоками, для локальних та віддалених медіа даних. Спочатку відбувається ініціалізація рендерів для відео- та аудіо-потоків. Після цього відбувається отримання локальних стрімів за допомогою асинхронної функції, яка ідентифікує пристрої введення та виведення пристрою, який використовує користувач. Детальний опис було наведено після ілюстрації схеми алгоритму. Схема алгоритму роботи блоку надсилання медіа даних представлена на рисунку 3.3.

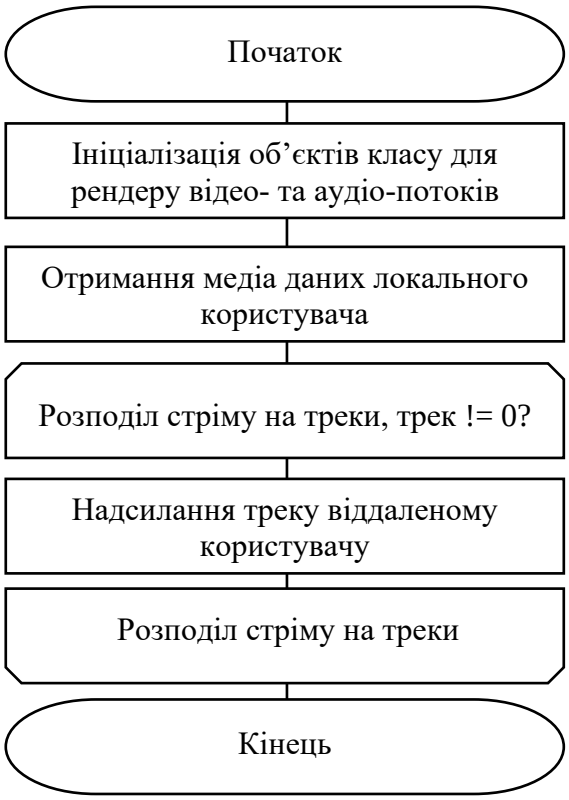


Рисунок 3.3 – Схема алгоритму роботи блоку надсилання медіа даних

Вхідним параметром передається об’єкт із параметрами зчитування стрімів, в ньому зазначаються потоки, які потрібно зчитувати та, якщо потрібно зчитувати відео, то яку камеру використати фронтальну чи основну. Зчитаний стрім записується у змінну, яка належить класу для зберігання медіа стрімів. За допомогою стандартного методу такого класу можна отримати масив треків із стріма, такі як відео- та аудіо-потік. Після цього стандартним методом перебору масиву відбувається прохід по всім елементам а саме трекам. Кожний трек відправляється разом із певною подією за допомогою методу класу, який відповідає за встановлення RTCP з’єднання, ця функція приймає два параметра, а саме трек, який надсилається та стрім, з якого був отриманий трек. Після того, як було встановлене з’єднання та створено об’єкт цього класу, потрібно встановити обробник подій на додавання треків за допомогою певного поля класу, який відповідає за з’єднання, в яке записується функція, яка спрацьовує при події додавання треку. Ця функція реалізує отримання віддалених треків у раніше створений об’єкт класу для рендеру, а саме у поле даного об’єкту, яке має містить

посилання. Схема алгоритму роботи блоку надсилання даних проілюстровано на рисунку 3.4.

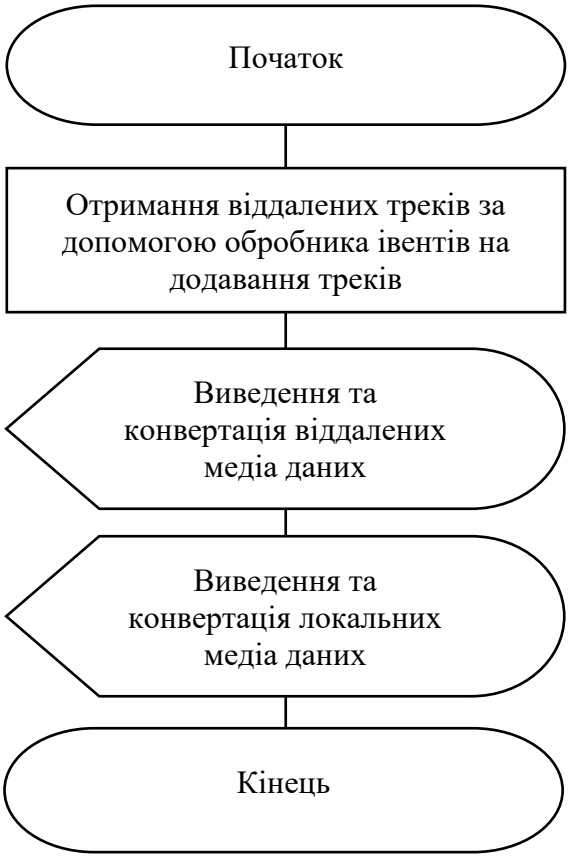


Рисунок 3.4 – Схема алгоритму роботи блоку отримання медіа даних

Після усіх вище описаних процедур та функцій, локальний та віддалений стріми було записано у об’єкти класу, який створений для рендеру медіа даних. Ці об’єкти передаються у вигляді параметра у віджет, який забезпечує бібліотека flutter_webrtc. Саме це дає можливість вивести медіа дані користувачів на екран. А для захисту цих даних використовується криптографічний захист.

3.3 Алгоритм роботи підмодуля криптографічного захисту

Алгоритм роботи підмодуля криптографічного захисту досить стандартний. Для реалізації криптостійкого захисту використано протокол для розподілу спільного секрету між користувачами та алгоритм блокового симетричного

шифрування AES-256, схема алгоритму роботи протоколу Diffie-Hellman проілюстровано на рисунку 3.5.

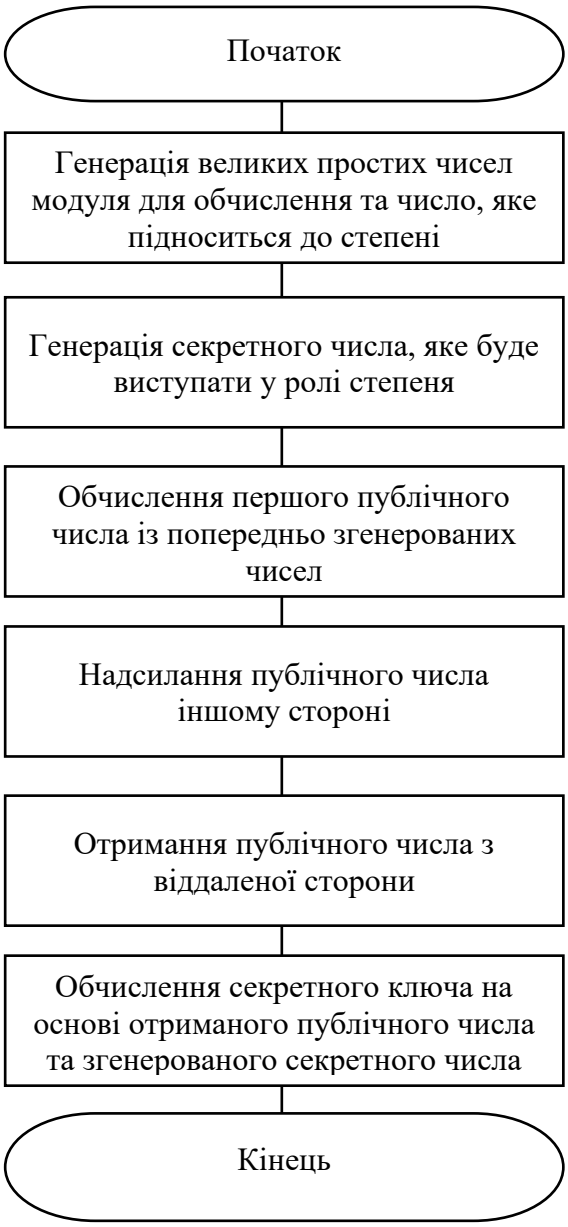


Рисунок 3.5 – Схема алгоритму роботи протоколу Diffie-Hellman

Для використання цього алгоритму було використано бібліотеку PointyCastle [40]. Це зручний набір класів та методів для здійснення різних криптографічних перетворень. Для генерації простих великих чисел для протоколу розподілу ключів було розроблено функцію, яка приймає кількість бітів як параметр та повертає просте число із вказаною кількістю бітів. Для перевірки згенерованого числа на простоту використано тест Міллера-Рабіна [41]. Даний алгоритм забезпечує досить низьку імовірність помилки при належній кількості циклів перевірки, але вимагає

досить багато часу для перевірки. Схема алгоритму роботи блоку шифрування проілюстрована на рисунку 3.6.

Рисунок 3.6 – Схема алгоритму роботи шифрування AES-256

Для шифрування за допомогою алгоритму AES-256 використовується згенерований секретний ключ, який зберігається у змінній secretKey. Для початку відбувається генерація вектору ініціалізації розміром 16 байтів та запис його у зміну із параметром final. Потім створюються параметри, які складаються з секретного ключа та вектору, ці параметри записуються у змінну params. Метод для створення параметрів реалізований у бібліотеці, яка використовується для шифрування. Наступним кроком відбувається ініціалізація режиму роботи класу cipher, а саме виставляється режим шифрування та передаються параметри у вигляді змінної params. Після попередньо проведених операцій, можна

використовувати функцію для шифрування, яка є методом класу cipher process. Цей метод отримує на вхід лише один параметр, дані, які необхідно зашифрувати.

В результаті виконання блоку шифрування було отримано зашифрований текст, який необхідно закодувати у форматі utf8, оскільки отримані дані після шифрування необхідно передавати у рядковому типі даних.

3.4 Алгоритм роботи підмодуля обміну параметрів з'єднання

Даний підмодуль вкрай важливий для встановлення якісного з'єднання між користувачами засобу, оскільки він відповідає за розповсюдження параметрів якості мережевого з'єднання клієнтів, медіа дані, які будуть передаватись, кодеки, які будуть використані для кодування аудіо- та відео-потоків та інші параметри комунікації. Для реалізації всіх цих задач, необхідно сформувати SDP-пакет, надіслати його через RTCP з'єднання та отримати такий самий об'єкт від віддаленого користувача, схема алгоритму роботи цього підмодуля була розбита на дві схеми, схема алгоритму роботи блоку надсилання SDP-пакету користувачам проілюстровано на рисунку 3.7.

Рисунок 3.7 – Схема алгоритму роботи блоку надсилання SDP пакету

Алгоритм роботи блоку надсилання SDP-пакету починається з ініціалізації об’єкту для опису RTCP з’єднання, яке записується у певну змінну. Даний об’єкт містить метод для формування локального об’єкту із параметрами для з’єднання, а саме кількість та типи стрімів та треків, оскільки це асинхронний метод, необхідно викликати його у конструкції async/await із ключовим словом await, що означає перетворення асинхронного коду у синхронний. Після формування локального SDP-пакету, він записується у спеціальну змінну. Наступним кроком є присвоєння сформованого об’єкту у змінну, яку відповідає за збереження параметрів з’єднання між користувачами. В кінці, для надсилання цього об’єкту віддаленому користувачу, використовується сигнальний сервер і певна подія (рис 3.8).

Рисунок 3.8 – Схема алгоритму роботи блоку отримання SDP-пакету

Алгоритм роботи блоку отримання SDP-пакету схожий до надсилання, однак спочатку було отримано віддалений об’єкт, оброблюючи подію вхідного виклику,

яка надходить від сигнального серверу. Після цього, ініціалізовано RTCP з'єднання між користувачами, об'єкт якого записується у певну змінну для наступних маніпуляцій. Наступним кроком було призначення віддаленого SDP-пакету за допомогою методу класу, який відповідає за встановлення RTCP з'єднання, після цього формується відповідний SDP-пакет із локальними параметрами з'єднання у відповідь на основі віддаленого об'єкту за допомогою методу того самого класу та записано у певну зміну для подальшого виведення стрімів користувачам. Останнім кроком є призначення локального дескриптора за допомогою методу того самого класу керування з'єднанням між користувачами.

В результаті роботи даного підмодуля формується RTCP з'єднання між користувачами на основі віддаленого та локального SDP-пакетів. Після формування об'єкту, який описує RTCP з'єднання відбувається подальша комунікація у вигляді надсилання та отримання аудіо- та відео-потоків.

3.5 Висновки з розділу

У даному розділі було розроблено узагальнений алгоритм роботи засобу захищеної інтернет телефонії та проілюстровано схему, яка описує розроблений алгоритм, що дозволило ефективно оцінити оптимально необхідні обчислювальні та мережеві ресурси для виконання даного алгоритму. Основну схему було поділено на два менших алгоритма, що дозволило досягнути більшого розуміння принципу роботи засобу та архітектури коду та спростило написання та розробку структури коду. Після цього було розроблено алгоритм роботи важливих блоків та підмодулів, таких як блок передавання даних, підмодулі криптографічного захисту та обміну параметрами з'єднання. Розробка детальних алгоритмів мінімальних складових програмного засобу дозволило передбачити різні сценарії під час виконання коду та підготувати обробку різних виключних ситуацій. Для всіх розроблених алгоритмів було наведено схеми, які були розділені на більш мілкі для детального пояснення функціональних можливостей підмодулів і блоків засобу захищеної інтернет-телефонії. Після розробки детального алгоритму роботи програми, необхідно переходити до реалізації та тестування програмного засобу.

4 ТЕСТУВАННЯ ЗАСОБУ ЗАХИЩЕНОЇ ІНТЕРНЕТ-ТЕЛЕФОНІЇ

4.1 Обґрунтування засобів розробки

Для розробки програмного засобу, необхідно обрати мову програмування для реалізації, середовище для розробки та необхідні фреймворки та бібліотеки, які потрібно використати для досягнення мети. Даний вибір є досить важливий, оскільки у кожного програмного застосунку є свої особливості розробки та свої вимоги, яких можна досягти, коректно обравши програмні засоби для розробки. Враховуючи основні вимоги до засобу, а саме кросплатформеність, можливість встановлення зв’язку між користувачами за допомогою протоколу WebRTC, можливість реалізації шифрування медіа даних та протоколу розподілу ключів, також, має бути досить висока швидкодія та зручний метод написання інтерфейсу користувача. Для початку необхідно проаналізувати та порівняти найбільш відомі мови програмування які відповідають висунутим вимогам. Серед найбільш відомих мов програмування можна виділити саме такі, як Java [42], JavaScript [43] та Dart [44]. Оскільки для даного виду застосунку досить важливий такий параметр як швидкодія, було проаналізовано цю характеристику обраних мов програмування. JavaScript має найнижчі показники серед обраних, оскільки не компілює код, який був написаний, у native машинний код, а виконуються за допомогою віртуальних машин JavaScript. Мова програмування Java також використовує віртуальну машину JVM, але, спочатку, компілюється у байт-код, а потім інтерпретується у native машинний код, тому є більш швидкою за JavaScript. Мова Dart підтримує динамічну компіляцію перед виконанням та компілюється у більш ефективний машинний код, що дозволяє збільшити швидкодію. При використанні цієї мови із фреймворком Flutter [45], можна досягнути майже такої самої швидкодії, як і у нативних мов програмування.

Наступним важливим показником є кросплатформеність, кожна із обраних мов програмування із використанням певних фреймворків дозволяє розробляти програмний застосунок одразу для кількох платформ, наприклад Android, iOS та Windows. Мова JavaScript із фреймворком React-Native та мова Dart із

фреймворком Flutter, надають легші методи побудови інтерфейсу користувача. Також, програмні застосунку, які розроблені за допомогою Flutter має більшу швидкодію та займає менше об’єму дискового простору. Також, великою перевагою мови програмування Dart є строга типізація, що забезпечує надійне, швидке та безпечне виконання програмного застосунку. Також, для мови програмування Dart доступна велика кількість пакетів, які дають змогу реалізувати додаткові функціональні можливості у програмних засобах, такі як шифрування, з’єднання користувачів за допомогою протоколу WebRTC та багато іншого.

Отже, із вище розглянутих мов програмування, найбільш підходить мова Dart, тому надалі усі засоби для розробки будуть аналізуватись для мови програмування Dart.

Далі необхідно проаналізувати середовище для розробки мовою програмування Dart та засіб для використання віртуальних мобільних пристроїв для тестування програмного засобу під час розробки. Основними середовищами для аналізу було обрано Visual Studio Code, IntelliJ IDEA, та WebStorm. Звісно, усі середовища розробки підтримують Dart та Flutter SDK, але найбільш різноманітний функціонал для збільшення зручності розробки надає Visual Studio Code, оскільки має досить багато плагінів, які покращують розробку як користувацького інтерфейсу, так і розробку серверу та основного функціоналу. Основним недоліком WebStorm є те, що він спрямований саме на розробку веб-застосунків, тому не є рекомендованим для розробки застосунків на інші платформи. Основною перевагою Visual Studio Code є вбудований Flutter SDK, який встановлюється за необхідністю, має дуже багато бібліотек для розширення можливостей розробки та має готові візуальні елементи для облегшення побудови інтерфейсу. У налаштуванні інтерфейсу, Visual Studio Code є досить легким та гнучким засобом. Також, має потужну інформаційну підтримку щодо користування. Для зручного зневадження програмного застосунку є підтримка віртуальних машин із Android Studio, що дозволяє спостерігати зміни на пристрої у режимі реального часу, тобто після збереження змін у коді, зміни одразу відображаються на віртуальній машині.

Для реалізації основних функціональних можливостей було обрано такі бібліотеки, як flutter_webrtc, socket.io_client та pointycastle. Бібліотека, яка забезпечує керування з'єднанням між користувачами за допомогою протоколу WebRTC та формування у коректному вигляді SDP-пакети є flutter_webrtc. Вона створена саме розробниками протоколу WebRTC для створення додатків для інтернет-телефонії за допомогою Flutter. Бібліотека socket.io_client забезпечує можливість з'єднання між користувачем та сокетом на стороні серверу, відслідковування та передачу певних івентів та даних разом. Саме ця бібліотека обрана, оскільки вона взаємодіє із архітектурою серверу, який використовує сокети, за допомогою бібліотеки socket.io. Остання обрана бібліотека реалізує багато алгоритмів шифрування даних у зручному представленні та має досить зрозумілу та велику документацію з прикладами для повного розуміння та коректного використання. Також, було розглянута бібліотека cryptography для Flutter, яка реалізує алгоритми шифрування. Фаворитом було обрано бібліотеку pointycastle, оскільки вона має більше користувачів та оновлюється по цей час, що збільшує довіру до того, що вона буде мати менше вад у своєму коді.

Отже, основним стаком засобів для розробки програмного засобу було обрано мову програмування Dart, фреймворк Flutter, середовище розробки Visual Studio Code та основні бібліотеки для реалізації основного функціоналу flutter_webrtc, socket.io_client, pointycastle.

4.2 Основні семантичні одиниці програмного коду

Для реалізації програмного застосунку було створено основні класи, які відповідають за інтерфейс користувача та містять певний стан в якому зберігаються основні змінні для таких даних, як аудіо- та відео-потоків користувачів, ідентифікатора користувача, спільний секрет для шифрування даних, об'єкту налаштування Peer-to-peer зв'язку між користувачами, об'єкти для рендеру локальних та віддалених медіа даних користувачів. Для використання сигнального серверу, було розроблено клас, який реалізує підключення до сокету на сервері та основні обробники подій для перевірки існування з'єднання між сервером та

користувачем. Єдине поле цього класу є змінна socket типу Socket, яка зберігає дескриптор створеного з’єднання до сокету на сервері. Два методи, які містить цей клас, це init({required String webSocketURL, required String selfCallerrID}) та dsconect(). Перший метод реалізує підключення до серверу за переданому посиланню та встановлює основні обробники подій для контролю за зв’язком між сервером та користувачем. Другий метод відповідає лише за відключення від серверу та сокету за необхідністю. Оскільки, генерація великих випадкових простих чисел для протоколу розповсюдження спільного секрету Diffie-Hellman потребує перевірку числа на простоту, що є досить ресурсозатратним процесом, було розроблено додатковий інструмент для генерації великих псевдовипадкових простих чисел. Для цього було розроблено доволі простий застосунок із інтерфейсом користувача для зручного користування та із трьома методами. Функція isProbablyPrime відповідає за реалізацію тесту Міллера-Рабіна для перевірки згенерованого числа на простоту. Допоміжна функція powMod розроблена для піднесення числа до степені за модулем, число степінь та модуль надходять як вхідні параметри. Функція isPrimeDefault реалізована лише для тестування генерації просто числа та реалізує звичайний підхід до перевірки числа на простоту. Функція generateNum відповідає за генерацію випадкового простого числа, вхідні параметри визначають довжину згенерованого числа, яка вказується у бітах. Функція isProbablyPrime перевірки на простоту викликається у функції generateNum генерації числа та повертає значення типу bool. При поверненому значенні true завершується генерація числа та виводиться згенероване число у текстове поле інтерфейсу користувача. За допомогою цього інструменту, генерується одразу два числа, а саме для модуля та для самого числа, яке буде підноситись до степені. Основні функціональні можливості засобу реалізовані у наступному файлі, а саме call_screen.dart. Цей файл також містить два класи, основним виступає клас, який наслідує стандартний клас State для віджиту Flutter. Основні поля класу, які взаємодіють між собою для встановлення з’єднання між користувачами є такими:

– socket – зберігає дескриптор сокету із класу для взаємодії із сигнальним сервером;
– _localRTCVideoRenderer – об’єкт класу RTCVideoRenderer, який зберігає посилання на локальні аудіо- та відео-потоки для відтворення даних користувачам;
– _remoteRTCVideoRenderer – об’єкт класу RTCVideoRenderer, в який записуються посилання на аудіо- та відео-потоки, яке отримується від віддаленого користувача;
– _localStream – об’єкт класу MediaStream для зберігання локальних медіа даних, які надходять з пристроїв введення;
– _rtcPeerConnection – об’єкт класу RTCPeerConnection, який зберігає усі параметри Peer-to-peer з’єднання між користувачами та виконує надсилання локальних медіа даних та івентів та встановлення обробника подій для отримання віддалених даних;
– основні змінні типу bool стосовно керування медіа даних, а саме isAudioOn, isVideoOn, isFrontCameraSelected.

Основним методом цього класу є _setupPeerConnection, який відповідає за ініціалізацію з’єднання, встановлення обробників подій, яка відбувається при отримані віддалених треків, отримання локальних медіа даних за допомогою стандартного об’єкту navigator та методу getUserMedia, в який передається об’єкт, який визначає які потоки треба зчитати, а саме аудіо- або відео-потік. Також, реалізується розподіл локального стріму на треки, які потім надсилаються віддаленому користувачу за допомогою об’єкту _rtcPeerConnection та його методу addTrack, який приймає необхідний трек та стрім, до якого належить цей трек. Отримані локальні стріми у змінній _localStream записуються у поле об’єкту _localStreamRenderer, а саме srcObject. Також, цей метод реалізує присвоєння параметрів RTCP з’єднанню за допомогою присвоєння локального та віддаленого SDP-пакетів за допомогою методів setLocalDescription та setRemoteDescription об’єкту _rtcPeerConnection. Після цього виконується обмін медіа даними між співрозмовниками.

Робота застосунку розпочинається із файлу main, який містить основну функцію, яка викликає клас WebRTCCallingApp, який наслідує основний стандартний клас фреймворку Flutter, а саме StatelessWidget, цей клас просто зазначає стандартні стилі для певних віджетів, віджет, з якого розпочинається інтерфейс користувача та посилання на сигнальний сервер. Перший віджет з якого розпочинається інтерфейс користувача описано у файлі home_screen.dart. Даний файл містить два класи HomeScreen, клас, який наслідує стандартний StatefulWidget та _HomeScreenState, який наслідує стандартний клас State<HomeScreen>. Основні методи для роботи даного віджету описано у другому класі. Основні методи для реалізації під'єднання до сокету та для виведення повідомлення про помилку при з'єднанні. Після того, як користувач з'єднується із сокетом, на екрані з'являється екран підключення до дзвінку з можливістю зробити вихідний виклик за певним ID іншого користувача. На початку рендерінгу скріна встановлюється обробник подій на вхідний виклик, який записує віддалений SDP-пакет у змінну incomingSDPOffer та з'являється спливаюче вікно для того, щоб прийняти або відхилити виклик. Для реалізації криптографічного захисту було розроблено два класи. Клас DiffieHellmanService реалізує основні обчислення для обміну та генерації спільного секрету. Цей клас містить три метода та п'ять полів для виконання усіх необхідних обчислювальних дій для генерації публічного секрету, а саме такі:

- поле a – містить згенероване секретне число, яке виступає у ролі степені;
- поле g – зберігає велике просте число, яке попередньо згенероване за допомогою додаткового інструменту для генерації великих простих чисел;
- поле p – зберігає велике просте число, яке виступає у ролі модуля в обчисленнях. Також, попередньо згенероване допоміжним інструментом;
- поле public – змінна для зберігання публічного локального числа;
- поле secretKey – змінна для зберігання згенерованого спільного секрету після всіх обчислень та обмінів інформацією;
- метод generateNum – генерує секретне число a;

– метод calculatingPublic – обчислює публічне число для надсилання віддаленій стороні;
– метод calculatingSecret – обчислює спільний секрет на основі отриманого публічного числа від віддаленої сторони та числа a.

Клас AES256Service розроблений для реалізації шифрування да розшифрування даних за допомогою алгоритму AES-256. Це клас містить одне поле та шість методів для коректної роботи шифрування. Ці поля та методи є такими:

– поле secretKey – містить секретний ключ для шифрування та розшифрування;
– метод generateSecretKey – генерує секретний ключ на основі спільного секрету, який обчислений за допомогою протоколу Diffie Hellman;
– метод aesEncrypt – реалізує шифрування, вхідними параметрами є дані для шифрування, секретний ключ та вектор ініціалізації;
– метод aesDecrypt – реалізує розшифрування, вхідні параметри такі самі, як і у попереднього методу;
– метод bin2hex – реалізує перетворення масиву бітів у строку із значеннями цих бітів у шістнадцятковій системі числення;
– метод hex2bin – реалізує перетворення строки із значеннями у шістнадцятковій системі числення у масив байтів;
– метод strToBin – реалізує перетворення певного тексту у масив байтів.

Код, основних класів та файлів у яких описано основні функціональні можливості розробленого програмного засобу для захищеної інтернет-телефонії наведено у додатку А.

Після розробки основного функціоналу програмного засобу для інтернет-телефонії, необхідно провести декілька типів тестувань та визначити вади розробленого засобу, якщо вони є.

4.3 Блокове тестування модуля захищеного зв’язку

Після того, як було розроблено засіб захищеної інтернет-телефонії, необхідно зробити блокове тестування. Спочатку необхідно протестувати додатково розроблений інструмент для генерації великих простих чисел. Для цього було розроблено декілька unit-тестів для основного функціоналу та декілька unit-тестів для перевірки заглушки при введені неможливої довжини числа у бітах.

Для основного функціоналу було розроблено шість тестів, по два для кожної функції. Всі тести згруповані у групу під назвою Prime checking tests. Перший тест перевіряє основну функцію генерації простого числа. Спочатку генерується просте число та записується у змінну number, після цього згенероване число перевіряється функцією, яка розроблена лише для тестування. Очікується результат true. Назва даного тесту Test prime number generation function через кому вказано чи позитивний має бути результат та номер тесту. Оскільки, це псевдовипадкове число, було виконано п’ять разів цей тест для більш точної перевірки.

Другий тест розроблений для перевірки функції, яка реалізує тест Міллера-Рабіна для перевірки числа на простоту. Цей тест використовується два рази, перший раз вхідним параметром для перевірки є дійсно просте число і очікуваний результат true. Другий раз вхідним параметром для функції є натуральне число і очікуваний результат false.

Оскільки було використано функцію для перевірки числа на простоту звичайним методом, її також необхідно протестувати. Третій тест повторює попередній, але тестує іншу функцію, а саме isDefaultPrime. Цей тест також було запущено два рази із позитивним результатом, тобто true, та з негативним, тобто false. Назва цього тесту Test for default function prime checking та через кому вказано який результат очікується отримати, позитивний чи негативний. Результати тестування цієї групи проілюстровано на рисунку 4.1.

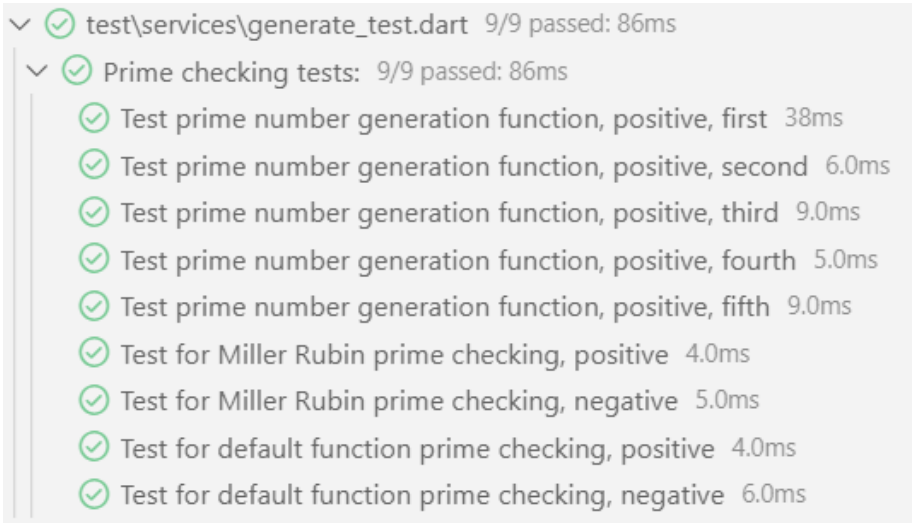


Рисунок 4.1 – Результати тестування групи Prime checking tests

Далі було проведено тестування інтерфейсу користувача, а саме спливаюче повідомлення про помилку, яке з’являється при введені занадто великого числа для довжини числа, яке буде генеруватись. Для цього тестування було використано основні методи класу WidgetTester із стандартної бібліотеки flutter_test, а саме pumpWidget, enterText, pump та tap, усі вони відповідають за взаємодію із віджетами для тестування. Спочатку вводиться певна довжина у поле для вводу, яке знайдене за певним ключем, далі натискається кнопка генерації чисел і перевіряється чи існує віджет спливаючого повідомлення про помилку. Було проведено два тести із позитивним та негативним результатами. Результати тестування наведено на рисунку 4.2.

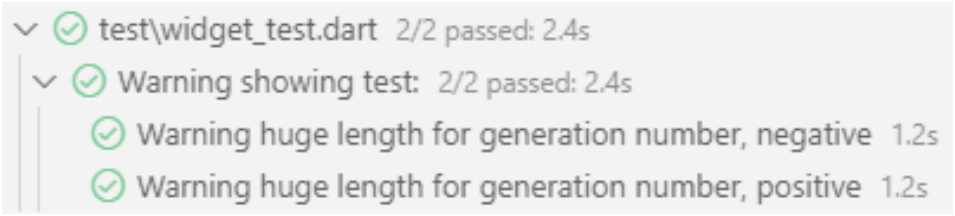


Рисунок 4.2 – Результати тестування групи Warning showing tests

Увесь код тестування додаткового інструменту для генерації великих простих чисел наведено у додатку Б.

Далі було проведено тестування основних функцій криптографічного захисту у модулі захищеного зв’язку. Розроблено тести для перевірки коректності

виконання функцій, які обчислюють публічне число та секретний ключ для протоколу Diffie-Hellman. Ці тести було об’єднано у групу під назвою Diffie Hellman generated joint secret tests. Для кожної функції було розроблено тест, спочатку тестується функція для обчислення публічного числа, а саме calculatingPublic. Вхідними даними під час тестування будуть тестові числа, розрахунки для яких було проведено вручну. Спочатку ми викликаємо функцію, яку тестуємо та отримуємо обраховане публічне число, далі порівнюємо отримане число із число, яке було результатом обрахунків вручну, результат порівняння записується у змінну типу bool. Для даної функції проведено два тести із позитивним та негативним результатами, а саме true та false в кінці виконання.

Для функції обчислення спільного секрету, а саме calculatingSecret, було розроблено подібний тест до попереднього. Змінюється функція для виклику та додається один вхідний параметр, а саме публічне число, яке отримується від віддаленої сторони. Результати обчислення порівнюються із результатами, які було отримано від ручного обрахування спільного секрету. Результати тестування наведено на рисунку 4.3.

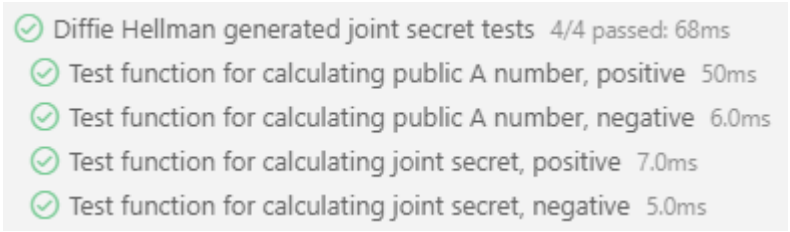


Рисунок 4.3 – Результати тестування для класу DiffieHellmanService

Для тестування методів класу AES256Service було розроблено два тести для перевірки методу шифрування та розшифрування. Тестові значення для перевірки було взято з офіційного сайту NIST [46]. Для початку було створено тест для перевірки методу aesEncrypt, для цього передаємо усі значення у функцію та порівнюємо результат із зашифрованим текстом із тестових значень. Для перевірки на негативний результат, змінимо один символ у зашифрованому тексту тестових значень. Для тестування методу розшифрування було створено подібний тест до

попереднього, різниця лише у функції, яка тестується, а саме aesDecrypt. Результати тестування проілюстровано на рисунку 4.4.

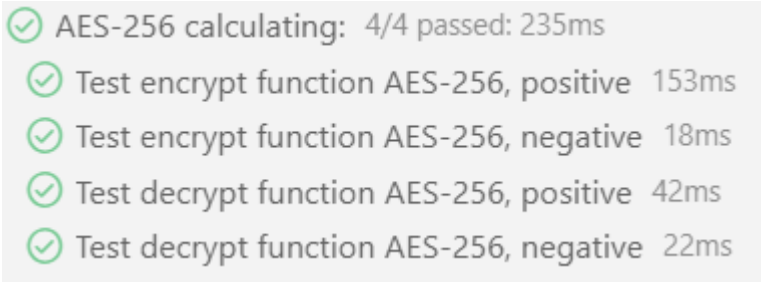


Рисунок 4.4 – Результати тестування для класу AES256Service

Увесь код тестування основних функціональних можливостей модуля захищеного зв’язку наведено у додатку В.

В результаті проведеного блокового тестування було перевірено коректність роботи основних функціональних можливостей розробленого засобу, на основі цих тестів було об’єднано роботу окремих блоків та розроблено модуль захищеного зв’язку без вад у коді. Після блокового тестування необхідно перейти до інтеграційного тестування розробленого засобу.

4.4 Інтеграційне тестування модуля захищеного зв’язку

Після того, як було проведено блокове тестування, необхідно провести інтеграційне тестування для того, щоб перевірити роботу усіх модулів у взаємодії.

На початку виконання модуля встановлення захищеного зв’язку, користувач отримує унікальний ID, який формується на етапі реєстрації у модулі автентифікації. За цим ідентифікатором встановлюється зв’язок між користувачами. Однак, для тестування було створено віджет, який дає можливість ввести тестовий ідентифікатор, результат роботи тестового віджету наведено на рисунку Г.1. Спочатку користувач вводить свій ID та натискає на кнопку для підключення до сигнального серверу, при неправильно введеному ідентифікаторі, з’являється попередження про це і користувач не може підключитись до сигнального серверу, приклад спливаючого попередження наведено на рисунку

Г.2. При вдалому підключенні, користувач переходить на інший екран, який дає змогу ввести ідентифікатор користувача, з яким необхідно встановити з’єднання, на даному етапі також є валідація стосовно коректного id. Результати роботи валідації та цього скріна наведено на рисунках Г.3 та Г.4. Також, на цьому екрані, при вхідному виклику, з’являється блок зверху екрана. Цей блок реалізує можливість відповіді на вхідний виклик, або ж відхилення. Також, на цьому блоці вказується, від кого надходить вхідний виклик, а саме ID віддаленого користувача. Приклад роботи даного блоку наведено на рисунку Г.5. Після цього, користувач переходить на екран дзвінку, де очікує прийняття та під’єднання віддаленого користувача до виклику, або ж відхилення виклику від віддаленого користувача. Робота програми у режимі очікування відповіді від віддаленого користувача наведено на рисунку Г.6. Також, цей скрін дозволяє керувати передаванням аудіо- та відео-потоків, а саме вимикати або вмикати відео-зв’язок та аудіо-зв’язок, також можна змінювати камеру, яка є ресурсом відео-потоку, а саме фронтальна камера або основна, при наявності двох, або більше пристроїв введення. Також, є можливість завершити з’єднання та спілкування із користувачем, під час натиснення цієї кнопки, відбувається перехід користувача на попередній екран. Усі ці кнопки керування винесені в окрему панель, яка розташована внизу екрану пристрою. Локальне відео виводиться у маленьке віконце, яке розташоване у правому нижньому кутку із невеликими відступами від правої та нижньої сторін екрану пристрою користувача. Віддалене відео розтягнуте на весь екран для більшої інформативності, оскільки ми хочемо дізнатись саме відео- та аудіо-інформацію від віддаленого користувача, а не локального. Кінцевий результат встановлення захищеного зв’язку проілюстровано на рисунках 4.5 та 4.6.

Всі інші етапи роботи модуля встановлення захищеного зв'язку наведено у додатку Г. Отже, після проведеного блокового та інтеграційного тестування, необхідно зробити висновки про дієздатність та коректність роботи розробленого модуля для встановлення захищеного зв'язку.

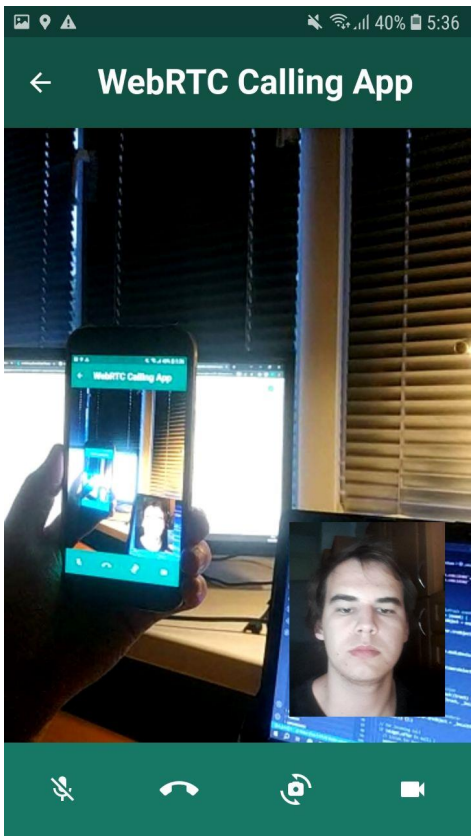


Рисунок 4.5 – Екран встановленого зв’язку зі сторони А

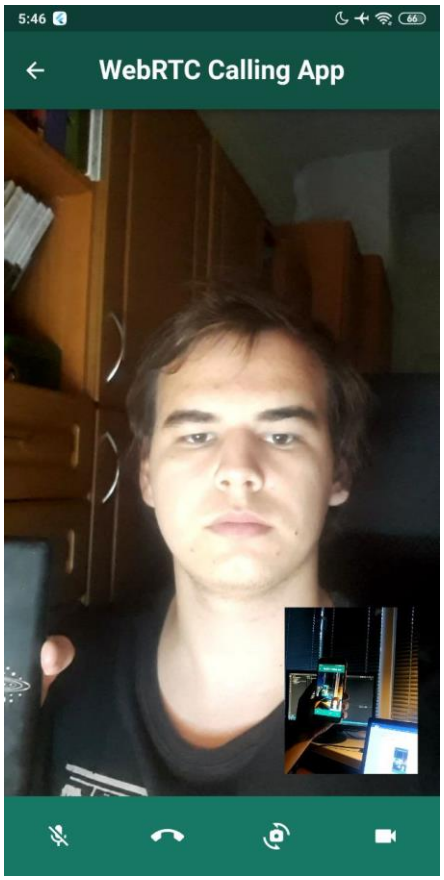


Рисунок 4.6 – Екран встановленого зв'язку зі сторони Б

4.5 Висновки з розділу

У даному розділі було проведено обґрунтування засобів, які були використані у розробці модуля встановлення захищеного зв’язку. Це обґрунтування, дало змогу підвищити продуктивність розробки програмного забезпечення шляхом компонування усіх засобів для розробки разом та розподілу усіх завдань для розробки на виконання коректним засобам. Після вдалого обґрунтування та розробки засобу, було наведено основні семантичні одиниці коду програмного забезпечення. Наведення та опис семантичних одиниць, дало змогу краще продемонструвати можливості у сфері розробки програмного забезпечення та більш детально пояснити механізм роботи розробленого засобу, вже на описі реально розробленого коду із використаними бібліотеками та фреймворками. Після детального огляду розроблених блоків та модулів програмного забезпечення, було проведено блокове тестування у автоматичному вигляді. Проведення unit-тестування дало змогу протестувати розроблені блоки окремо без взаємодії зі всіма іншими, що досить важливо для подальшої розробки та компонування окремих блоків у модулі для реалізації більш складних функціональних можливостей засобу. Після цього виду тестування було проведено інтеграційне тестування, яке дало змогу перевірити роботу усього модуля встановлення захищеного зв’язку у цілому та визначити певні вади про виконані певних функцій.

Отже, в результаті проведеного тестування, у розробленому програмного засобі не було виявлено значних вад, які унеможливлюють встановлення захищеного з’єднання для інтернет-телефонії, що дає змогу стверджувати, що розробка цього модуля пройшла успішно.

ВИСНОВКИ

Виконаний в роботі аналіз використання інтернет-телефонії, як способу спілкування між людьми як під час трудової діяльності, так і під час приватного спілкування, дозволив довести актуальність забезпечення безпеки в цьому способі зв'язку. Це пояснюється ти, що основні засоби інтернет-телефонії, які використовуються найбільше на даний момент, це WhatsApp, Signal, Zoom, Skype. Однак подальший аналіз дозволив виявити, що в зазначених засобах існують випадки несанкціонованого витоку конфіденційної інформації користувачів із власних серверів, що створює великий ризик для конфіденційності інформації користувачів.

Саме тому, дана робота була присвячена розробці засобу захищеної інтернет-телефонії, а саме модуля захищеного зв'язку, в якому використовується Peer-to-Peer зв'язок без зберігання конфіденційної інформації користувачів на серверах, що унеможливлює несанкціонований витік з серверів.

Для досягнення мети було проаналізовано засоби інтернет-телефонії та виконано декомпозицію їх основних структурних елементів та алгоритмів роботи, що стало основою їх порівняльного аналізу. Для забезпечення безпеки в модулі, що розробляється в межах цієї частини комплексної бакалаврської роботи було здійснено аналіз методів передавання медіа даних, який на основі варіантного аналізу дозволив обґрунтувати вибір технології WebRTC. Оскільки математично доведену стійкість засобів кібербезпеки можливо досягти лише шляхом використання методів криптографії, було проведено порівняльний аналіз методів шифрування, який дозволив визначити AES-256, як найбільш перспективний. Було розроблено структуру засобу та окремих його основних блоків та модулів, що дозволило здійснити керування складністю процесу проектування шляхом декомпозиції задачі. В наслідок цієї декомпозиції було виділено модулі, які співпрацюють для забезпечення стійкого захисту. Також, розроблено узагальнений алгоритм роботи цілого засобу та алгоритми основних його окремих модулів та блоків, що дозволило ефективно оцінити оптимально необхідні обчислювальні та

мережеві ресурси для реалізації даного алгоритму у засобі. Розроблено програмний засіб, який реалізує захищений зв’язок між користувачами для обміну медіа даними, такими як аудіо- та відео-потоками. Проведено блокове тестування, що дало змогу перевірити коректність роботи усіх окремих розроблених блоків, що досить важливо для подальшої розробки, оскільки потрібно буде здійснити компонування цих блоків у модулі для реалізації більш складних функціональних можливостей засобу. Також, було проведено інтеграційне тестування розробленого засобу, що дозволило здійснити перевірку уже повного модуля встановленого захищеного зв’язку та переконатись у коректній роботі розробленого засобу та можливостях, які реалізують функціональні можливості.

Отже, за допомогою виконання усіх перерахованих вище завдань, було мінімізовано ризики несанкціонованого виливу конфіденційної інформації користувачів із серверів, які належать компанії розробника програмного забезпечення, що і було висвітлено, як основна проблема сучасних існуючих засобів інтернет-телефонії. Розроблений модуль для встановлення захищеного зв’язку, може бути використаний для впровадження у систему JetIQ для впровадження онлайн занять. Також, архітектуру розробленого модуля для інтернет-телефонії можна зробити більш масштабовану за допомогою впровадження конференц-зв’язку. Обрані алгоритми та методи криптографічного захисту та передавання медіа даних дозволяють таке вдосконалення розробленого модуля.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Козак О. М., Каплун В. А. Засіб для захисту програмного забезпечення від несанкціонованого копіювання і дослідження: наук.-техн. конф. факультету інформаційних технологій та комп'ютерної інженерії. 2022р. : тези доповідей. Вінниця: ВНТУ, 2011. С. 3;
2. Комп'ютерна програма «SKMProtect» : а. с. 117287 Україна / О. М. Козак. № с202300812 ; опубл. 20.03.2023;
3. Комп'ютерна програма "Тестування множини непарних складених чисел на приналежність до Кармайклових чисел" : а. с. 95986 Україна / О. М. Козак. № 96945 ; заявл. 08.01.2020 ; опубл. 11.02.2020;
4. Що таке IP-телефонія і як це працює. URL: https://itel.ua/articles/shho-take-ip-telefonija-i-jak-ce-pracjuye (дата звернення: 21.05.2023);
5. WhatsApp. *WhatsApp.com*. URL: https://www.whatsapp.com/?lang=uk (accessed: 21.05.2023);
6. Ходєва С. WhatsApp і дані користувачів. Чи виправдано обурення новими правилами конфіденційності?. *ms.detector.media*. URL: https://ms.detector.media/kiberbezpeka/post/26404/2021-01-15-whatsapp-i-dani-korystuvachiv-chy-vypravdano-oburennya-novymy-pravylamy-konfidentsiynosti/ (дата звернення: 21.05.2023);
7. Про наскрізне шифрування | Довідковий центр WhatsApp. *WhatsApp Help Center*. URL: https://faq.whatsapp.com/820124435853543/?locale=uk_UA (дата звернення: 25.05.2023);
8. Documentation. *Signal Messenger*. URL: https://signal.org/docs/ (accessed: 21.05.2023);
9. Specifications >> The X3DH Key Agreement Protocol. *Signal Messenger*. URL: https://signal.org/docs/specifications/x3dh/ (accessed: 21.05.2023);
10. Specifications >> The Double Ratchet Algorithm. *Signal Messenger*. URL: https://signal.org/docs/specifications/doubleratchet/ (accessed: 21.05.2023);

11. Specifications >> The XEdDSA and VXEdDSA Signature Schemes. *Signal Messenger*. URL: https://signal.org/docs/specifications/xeddsa/ (accessed: 21.05.2023);
12. Signal Messenger: Speak Freely. *Signal Messenger*. URL: https://signal.org/uk/ (accessed: 21.05.2023);
13. Looking back at how Signal works, as the world moves forward. *Signal Messenger*. URL: https://signal.org/blog/looking-back-as-the-world-moves-forward/ (accessed: 21.05.2023);
14. Skype | Спілкуйтеся в безкоштовних відеовикликах, хоч де ви будете. *Skype | Stay connected with free video calls worldwide*. URL: https://www.skype.com/uk/ (дата звернення: 22.05.2023);
15. Microsoft. *Microsoft*. URL: https://www.microsoft.com/uk-ua (accessed: 25.05.2023);
16. Чи використовує Skype шифрування? | Підтримка Skype. *Skype Support for All products | Skype Support*. URL: https://support.skype.com/uk/faq/FA31/chi-vikoristovuie-skype-shifruvannya (дата звернення: 22.05.2023);
17. Як дарувати гроші на рахунку Skype друзям і родичам | Підтримка Skype. *Skype Support for All products | Skype Support*. URL: https://support.skype.com/uk/faq/FA12197/yak-daruvati-groshi-na-rakhunku-skype-druzyam-i-rodicham (дата звернення: 22.05.2023);
18. Онлайновий номер телефону | Skype-номер | Skype. *Skype | Stay connected with free video calls worldwide*. URL: https://www.skype.com/uk/features/online-number/ (дата звернення: 22.05.2023);
19. One platform to connect | Zoom. Zoom. URL: https://zoom.us/ (accessed: 22.05.2023);
20. End-to-end (E2EE) encryption for meetings. *Zoom*. URL: https://support.zoom.us/hc/en-us/articles/360048660871 (accessed: 25.05.2023);
21. Video Conferencing, Web Conferencing, Webinars, Screen Sharing. *Zoom*. URL: https://zoom.us/download (accessed: 25.05.2023);

22. Setting up advanced chat encryption. *Zoom Support.* URL: https://support.zoom.us/hc/en-us/articles/207599823 (accessed: 22.05.2023);
23. Vonage business. Business Phone, VoIP, Communication APIs, Contact Center | Vonage. URL: https://www.vonage.co.uk/ (accessed: 08.06.2023);
24. HTTP Live Streaming | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/http-live-streaming (accessed: 22.05.2023);
25. Introduction to the Real-time Transport Protocol (RTP) - Web APIs | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Intro_to_RTP (accessed: 22.05.2023);
26. WebRTC. WebRTC. URL: https://webrtc.org/?hl=en (accessed: 23.05.2023);
27. Introduction to WebRTC protocols - Web APIs | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols (accessed: 23.05.2023);
28. Standards – MPEG. *MPEG – The Moving Picture Experts Group*. URL: https://www.mpeg.org/standards/MPEG-DASH/ (accessed: 23.05.2023);
29. Daemon J., Rijmen V. The Design of Rijndael: The Advanced Encryption Standard. 2nd ed. Springer, 2020. 300 p;
30. Baksi A. Classical and Physical Security of Symmetric Key Cryptographic Algorithms. Springer Singapore Pte. Limited, 2021. 300 p;
31. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. Чинний від 2016-03-01. Вид. офіц. 228 с;
32. News Release 050710. Wayback Machine. URL: https://web.archive.org/web/20190719142326if_/ntt.co.jp/news/news05e/0507/050720.html (accessed: 26.05.2023);
33. Data encryption standard (DES) | Set 1 - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/ (accessed: 26.05.2023);

34. Lehtinen S. Diffie-Hellman key exchange: from mathematics to real life. LAP LAMBERT Academic Publishing, 2012. 92 p;
35. Pointcheval D. Asymmetric Cryptography: Primitives and Protocols. Wiley & Sons, Incorporated, John, 2023. 304 p;
36. WebRTC API - Web APIs | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API (accessed: 10.06.2023);
37. TURN server. *MDN Web Docs*. URL: https://developer.mozilla.org/ru/docs/Web/API/WebRTC_API/Protocols#turn (accessed: 30.05.2023);
38. Flutter_webrtc 0.9.32. *Pub.dev*. URL: https://pub.dev/packages/flutter_webrtc (accessed: 31.05.2023);
39. Socket_io_client 2.0.2. *Pub.dev*. URL: https://pub.dev/packages/socket_io_client (accessed: 31.05.2023);
40. Pointycastle 3.7.3. *Pub.dev*. URL: https://pub.dev/packages/pointycastle (accessed 31.05.2023);
41. Number theory - primality tests. Applied Cryptography Group | Stanford University. URL: https://crypto.stanford.edu/pbc/notes/numbertheory/millerrabin.html (accessed: 31.05.2023);
42. What is java? - java programming language explained - AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/what-is/java/?nc1=h_ls (accessed: 09.06.2023);
43. What is JavaScript? - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (accessed: 09.06.2023);
44. Dart programming language. Dart programming language | Dart. URL: https://dart.dev/ (accessed: 09.06.2023);

45. Flutter - Build apps for any screen. Flutter - Build apps for any screen. URL: https://flutter.dev/ (accessed: 09.06.2023);
46. NIST Computer Security Resource Center | CSRC. URL: http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip (accessed: 12.06.2023).

ДОДАТКИ

Додаток А. Код основних функціональних можливостей засобу

Код файлу signalling.service.dart основного засобу

```
import 'dart:developer';
import 'package:socket_io_client/socket_io_client.dart';

class SignallingService {
  Socket? socket;

  SignallingService._();
  static final instance = SignallingService._();

  init({required String websocketUrl, required String selfCallerID}) {
    socket = io(websocketUrl, {
      "transports": ['websocket'],
      "query": {"callerId": selfCallerID}
    });

    socket!.onConnect((data) {
      log("Socket connected !!");
    });

    socket!.onConnectError((data) {
      log("Connect Error $data");
    });

    socket!.connect();
  }

  disconnect() {
    socket!.close();
  }
}
```

Код файлу generate.service.dart додаткового інструменту

```
import 'dart:math';

bool isProbablyPrime(int number, int k) {
  if (number == 2 || number == 3) {
    return true;
  }

  if (number < 2 || number % 2 == 0) {
    return false;
  }

  int r = 0;
  int d = number - 1;

  while (d % 2 == 0) {
```

```
    r++;
    d = d ~/ 2;
  }

  for (int i = 0; i < k; i++) {
    int a = 2 + Random().nextInt(number - 3);
    int x = powMod(a, d, number);

    if (x == 1 || x == number - 1) {
      continue;
    }

    bool isWitness = false;

    for (int j = 0; j < r - 1; j++) {
      x = powMod(x, 2, number);

      if (x == 1) {
        return false;
      }

      if (x == number - 1) {
        isWitness = true;
        break;
      }
    }

    if (!isWitness) {
      return false;
    }
  }

  return true;
}

int powMod(int base, int exponent, int modulus) {
  int result = 1;

  while (exponent > 0) {
    if (exponent % 2 == 1) {
      result = (result * base) % modulus;
    }

    base = (base * base) % modulus;
    exponent = exponent ~/ 2;
  }

  return result;
}

bool isPrimeDefault(int number) {
  if (number < 2) {
```

```
    return false;
  }

  if (number == 2 || number == 3) {
    return true;
  }

  if (number % 2 == 0 || number % 3 == 0) {
    return false;
  }

  int sqrtNumber = sqrt(number).toInt();

  for (int i = 5; i <= sqrtNumber; i += 6) {
    if (number % i == 0 || number % (i + 2) == 0) {
      return false;
    }
  }

  return true;
}

int generateNum(int bitLength) {
  int min = 1;
  int max = 1;

  for(int i = 0; i < bitLength; i++) {
    max *= 2;
    if(i < bitLength - 1) {
      min *= 2;
    }
  }

  while(true) {
    int result = min + Random().nextInt(max - min);
    if(isProbablyPrime(result, 100)) {
      return result;
    }
  }

}
```

Код основного функціоналу файлу home_screen.dart основного засобу

```
connectToSocket({
    required String selfCallerId
  }) {
    if(textEditingController.text.length < 6) {
      _showAlertNullID();
    } else {
      SignallingService.instance.init(
        websocketUrl: widget.websocketURL,
        selfCallerID: selfCallerId,
```

```
    );
    textEditingController.clear();
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (_) => JoinScreen(
          selfCallerId: selfCallerId
        )
      )
    );
  }
}
```

Код основного функціоналу файлу join_screen.dart основного засобу

```
dynamic incomingSDPOffer;
  final remoteCallerIdTextEditingController = TextEditingController();

  @override
  void initState() {
    super.initState();

    // listen for incoming video call
    SignallingService.instance.socket!.on("newCall", (data) {
      if (mounted) {
        // set SDP Offer of incoming call
        setState(() => incomingSDPOffer = data);
      }
    });
  }

  // join Call
  _joinCall({
    required String callerId,
    required String calleeId,
    dynamic offer,
  }) {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (_) => CallScreen(
          callerId: callerId,
          calleeId: calleeId,
          offer: offer,
        ),
      ),
    );
  }
```

Код основного функціоналу файлу call_screen.dart основного засобу

```
  final socket = SignallingService.instance.socket;
```

```
final _localRTCVideoRenderer = RTCVideoRenderer();

final _remoteRTCVideoRenderer = RTCVideoRenderer();

MediaStream? _localStream;

RTCPeerConnection? _rtcPeerConnection;

List<RTCIceCandidate> rtcIceCadidates = [];

bool isAudioOn = true, isVideoOn = true, isFrontCameraSelected = true;

@override
void initState() {
  _localRTCVideoRenderer.initialize();
  _remoteRTCVideoRenderer.initialize();

  _setupPeerConnection();
  super.initState();
}

_setupPeerConnection() async {
  _rtcPeerConnection = await createPeerConnection({
    'iceServers': [
      {
        'urls': [
          'stun:stun1.l.google.com:19302',
          'stun:stun2.l.google.com:19302'
        ]
      }
    ]
  });

  _rtcPeerConnection!.onTrack = (event) {
    _remoteRTCVideoRenderer.srcObject = event.streams[0];
    setState(() {});
  };

  _localStream = await navigator.mediaDevices.getUserMedia({
    'audio': isAudioOn,
    'video': isVideoOn
        ? {'facingMode': isFrontCameraSelected ? 'user' : 'environment'}
        : false,
  });

  _localStream!.getTracks().forEach((track) {
    _rtcPeerConnection!.addTrack(track, _localStream!);
  });

  _localRTCVideoRenderer.srcObject = _localStream;
  setState(() {});
```

```
if (widget.offer != null) {
  // listen for Remote IceCandidate
  socket!.on("IceCandidate", (data) {
    String candidate = data["iceCandidate"]["candidate"];
    String sdpMid = data["iceCandidate"]["id"];
    int sdpMLineIndex = data["iceCandidate"]["label"];

    _rtcPeerConnection!.addCandidate(RTCIceCandidate(
      candidate,
      sdpMid,
      sdpMLineIndex,
    ));
  });

  await _rtcPeerConnection!.setRemoteDescription(
    RTCSessionDescription(widget.offer["sdp"], widget.offer["type"]),
  );

  RTCSessionDescription answer = await _rtcPeerConnection!.createAnswer();
  _rtcPeerConnection!.setLocalDescription(answer);
  socket!.emit("answerCall", {
    "callerId": widget.callerId,
    "sdpAnswer": answer.toMap(),
  });
}
else {
  _rtcPeerConnection!.onIceCandidate =
      (RTCIceCandidate candidate) => rtcIceCadidates.add(candidate);

  socket!.on("callAnswered", (data) async {
    await _rtcPeerConnection!.setRemoteDescription(
      RTCSessionDescription(
        data["sdpAnswer"]["sdp"],
        data["sdpAnswer"]["type"],
      ),
    );
    for (RTCIceCandidate candidate in rtcIceCadidates) {
      socket!.emit("IceCandidate", {
        "calleeId": widget.calleeId,
        "iceCandidate": {
          "id": candidate.sdpMid,
          "label": candidate.sdpMLineIndex,
          "candidate": candidate.candidate
        }
      });
    }
  });
  RTCSessionDescription offer = await _rtcPeerConnection!.createOffer();
  await _rtcPeerConnection!.setLocalDescription(offer);
  socket!.emit('makeCall', {
    "calleeId": widget.calleeId,
    "sdpOffer": offer.toMap(),
```

```
      });
    }
  }
  _leaveCall() {
    Navigator.pop(context);
  }
  _toggleMic() {
    isAudioOn = !isAudioOn;
    _localStream?.getAudioTracks().forEach((track) {
      track.enabled = isAudioOn;
    });
    setState(() {});
  }
  _toggleCamera() {
    isVideoOn = !isVideoOn;
    _localStream?.getVideoTracks().forEach((track) {
      track.enabled = isVideoOn;
    });
    setState(() {});
  }
  _switchCamera() {
    isFrontCameraSelected = !isFrontCameraSelected;
    _localStream?.getVideoTracks().forEach((track) {
      track.switchCamera();
    });
    setState(() {});
  }
```

Код файлу cryptography.service.dart

```
import 'dart:convert';
import 'dart:math';
import 'dart:typed_data';

import "package:pointycastle/export.dart";

class DiffieHellmanService {
  BigInt a = BigInt.zero;
  BigInt g = BigInt.from(2161);
  BigInt p = BigInt.from(3581);
  BigInt public = BigInt.zero;
  BigInt secretKey = BigInt.zero;

  DiffieHellmanService._();
  static final instance = DiffieHellmanService._();

  generateNum(int bitLength) {
    BigInt min = BigInt.one;
    BigInt max = BigInt.one;

    for(int i = 0; i < bitLength; i++) {
      max *= BigInt.two;
      if(i < bitLength - 1) {
```

```
      min *= BigInt.two;
    }
   }
   a = BigInt.from(Random().nextDouble() * (max.toDouble() - min.toDouble()) + min.toDouble());
  }

  calculatingPublic(int A) {
   BigInt res = BigInt.one;
   for(int i = 0; i < A; i++) {
    res *= g;
   }
   res %= p;
   public = res;
  }

  calculatingSecret(BigInt remotePublic, int A) {
   BigInt res = BigInt.one;
   for(int i = 0; i < A; i++) {
    res *= remotePublic;
   }
   res %= p;
   secretKey = res;
  }
}

class AES256Service {
  String secretKey = "";

  AES256Service._();
  static final instance = AES256Service._();

  generateSecretKey(BigInt jointSecret) {

  }

  Uint8List aesEncrypt(Uint8List key, Uint8List iv, Uint8List paddedPlaintext) {
   if (![128, 192, 256].contains(key.length * 8)) {
    throw ArgumentError.value(key, 'key', 'invalid key length for AES');
   }
   if (iv.length * 8 != 128) {
    throw ArgumentError.value(iv, 'iv', 'invalid IV length for AES');
   }
   if (paddedPlaintext.length * 8 % 128 != 0) {
    throw ArgumentError.value(
        paddedPlaintext, 'paddedPlaintext', 'invalid length for AES');
   }

   final cbc = CBCBlockCipher(AESEngine())
    ..init(true, ParametersWithIV(KeyParameter(key), iv));

   final cipherText = Uint8List(paddedPlaintext.length);
```

```
  var offset = 0;
  while (offset < paddedPlaintext.length) {
    offset += cbc.processBlock(paddedPlaintext, offset, cipherText, offset);
  }
  assert(offset == paddedPlaintext.length);

  return cipherText;
}

Uint8List aesDecrypt(Uint8List key, Uint8List iv, Uint8List cipherText) {
  if (![128, 192, 256].contains(key.length * 8)) {
    throw ArgumentError.value(key, 'key', 'invalid key length for AES');
  }
  if (iv.length * 8 != 128) {
    throw ArgumentError.value(iv, 'iv', 'invalid IV length for AES');
  }
  if (cipherText.length * 8 % 128 != 0) {
    throw ArgumentError.value(
        cipherText, 'cipherText', 'invalid length for AES');
  }

  final cbc = CBCBlockCipher(AESEngine())
    ..init(false, ParametersWithIV(KeyParameter(key), iv));

  final paddedPlainText = Uint8List(cipherText.length);

  var offset = 0;
  while (offset < cipherText.length) {
    offset += cbc.processBlock(cipherText, offset, paddedPlainText, offset);
  }
  assert(offset == cipherText.length);

  return paddedPlainText;
}

String bin2hex(Uint8List bytes, {String? separator, int? wrap}) {
  var len = 0;
  final buf = StringBuffer();
  for (final b in bytes) {
    final s = b.toRadixString(16);
    if (buf.isNotEmpty && separator != null) {
      buf.write(separator);
      len += separator.length;
    }
    if (wrap != null && wrap < len + 2) {
      buf.write('\n');
      len = 0;
    }
    buf.write('${(s.length == 1) ? '0' : ''}$s');
    len += 2;
  }
  return buf.toString();
```

```
  }

  Uint8List hex2bin(String hexStr) {
    if (hexStr.length % 2 != 0) {
      throw const FormatException('not an even number of hexadecimal characters');
    }
    final result = Uint8List(hexStr.length ~/ 2);
    for (int i = 0; i < result.length; i++) {
      result[i] = int.parse(hexStr.substring(2 * i, 2 * (i + 1)), radix: 16);
    }
    return result;
  }

  strToBin(String str) {
    return utf8.encode(str);
  }
}
```

Додаток Б. Код тестування інструмента для генерації великих простих чисел

Код файлу generate_test.dart

```
import 'package:flutter_test/flutter_test.dart';
import 'package:generate_prime_app/services/generate.services.dart';

void main() {
  group("Prime checking tests", () {
    test("Test prime number generation function, positive, first", () {
      int number = generateNum(32);
      bool result = isPrimeDefault(number);
      expect(result, true);
    });

    test("Test prime number generation function, positive, second", () {
      int number = generateNum(32);
      bool result = isPrimeDefault(number);
      expect(result, true);
    });

    test("Test prime number generation function, positive, third", () {
      int number = generateNum(32);
      bool result = isPrimeDefault(number);
      expect(result, true);
    });

    test("Test prime number generation function, positive, fourth", () {
      int number = generateNum(32);
      bool result = isPrimeDefault(number);
      expect(result, true);
    });

    test("Test prime number generation function, positive, fifth", () {
      int number = generateNum(32);
      bool result = isPrimeDefault(number);
      expect(result, true);
    });

    test("Test for Miller Rubin prime checking, positive", () {
      bool result = isProbablyPrime(65537, 10);
      expect(result, true);
    });

    test("Test for Miller Rubin prime checking, negative", () {
      bool result = isProbablyPrime(36568, 10);
      expect(result, false);
    });

    test("Test for default function prime checking, positive", () {
      bool result = isPrimeDefault(65537);
```

```
    expect(result, true);
   });

   test("Test for default function prime checking, negative", () {
    bool result = isPrimeDefault(352568);
    expect(result, false);
   });
  });
}
```

Код файл widget_test.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'package:generate_prime_app/main.dart';

void main() {
  group('Warning showing test:', () {
   testWidgets('Warning huge length for generation number, positive', (WidgetTester tester) async {
    await tester.pumpWidget(const MyApp());

    await tester.enterText(find.byKey(const Key("Text Field")), "33");
    await tester.pump();
    expect(find.text('33'), findsOneWidget);
    await tester.tap(find.byKey(const Key("Generate button")));
    await tester.pump();

    expect(find.byType(AlertDialog), findsOneWidget);
   });

   testWidgets('Warning huge length for generation number, negative', (WidgetTester tester) async {
    await tester.pumpWidget(const MyApp());

    await tester.enterText(find.byKey(const Key("Text Field")), "32");
    await tester.pump();
    expect(find.text('32'), findsOneWidget);
    await tester.tap(find.byKey(const Key("Generate button")));
    await tester.pump();

    expect(find.byType(AlertDialog), findsNothing);
   });
  });
}
```

Додаток В. Код тестування основного функціоналу модуля

```
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter_webrtc_app/services/cryptography.service.dart';

void main() {
  group("Diffie Hellman generated joint secret tests", () {
    test("Test function for calculating public A number, positive", () {
      DiffieHellmanService.instance.calculatingPublic(5);
      BigInt public = DiffieHellmanService.instance.public;
      bool result = public == BigInt.from(3019);
      expect(result, true);
    });

    test("Test function for calculating public A number, negative", () {
      DiffieHellmanService.instance.calculatingPublic(5);
      BigInt public = DiffieHellmanService.instance.public;
      bool result = public == BigInt.from(12);
      expect(result, false);
    });

    test("Test function for calculating joint secret, positive", () {
      DiffieHellmanService.instance.calculatingSecret(BigInt.from(3058), 5);
      BigInt secret = DiffieHellmanService.instance.secretKey;
      bool result = secret == BigInt.from(2034);
      expect(result, true);
    });

    test("Test function for calculating joint secret, negative", () {
      DiffieHellmanService.instance.calculatingSecret(BigInt.from(3058), 5);
      BigInt secret = DiffieHellmanService.instance.secretKey;
      bool result = secret == BigInt.from(2032);
      expect(result, false);
    });
  });

  group('AES-256 calculating:', () {
    test("Test encrypt function AES-256, positive", () {
      final cipher = AES256Service.instance.aesEncrypt(

AES256Service.instance.hex2bin('00000000000000000000000000000000000000000000000000000000000000
0000000000'),
        AES256Service.instance.hex2bin('00000000000000000000000000000000'),
        AES256Service.instance.hex2bin('014730f80ac625fe84f026c60bfd547d')
      );
      final result = AES256Service.instance.bin2hex(cipher);
      expect(result, "5c9d844ed46f9885085e5d6a4f94c7d7");
    });

    test("Test encrypt function AES-256, negative", () {
      final cipher = AES256Service.instance.aesEncrypt(
```

```
AES256Service.instance.hex2bin('00000000000000000000000000000000000000000000000000000000000000000000000000'),
      AES256Service.instance.hex2bin('00000000000000000000000000000000'),
      AES256Service.instance.hex2bin('014730f80ac625fe84f026c60bfd547d')
    );
    String encryptText = AES256Service.instance.bin2hex(cipher);
    bool result = encryptText == "5c9d844ed26f9885085e5d6a4f94c7d7";
    expect(result, false);
  });

  test("Test decrypt function AES-256, positive", () {
    final cipher = AES256Service.instance.aesDecrypt(

AES256Service.instance.hex2bin('00000000000000000000000000000000000000000000000000000000000000000000000000'),
      AES256Service.instance.hex2bin('00000000000000000000000000000000'),
      AES256Service.instance.hex2bin('1bc704f1bce135ceb810341b216d7abe')
    );
    final result = AES256Service.instance.bin2hex(cipher);
    expect(result, "91fbef2d15a97816060bee1feaa49afe");
  });

  test("Test decrypt function AES-256, negative", () {
    final cipher = AES256Service.instance.aesDecrypt(

AES256Service.instance.hex2bin('00000000000000000000000000000000000000000000000000000000000000000000000000'),
      AES256Service.instance.hex2bin('00000000000000000000000000000000'),
      AES256Service.instance.hex2bin('1bc704f1bce135ceb810341b216d7abe')
    );
    String encryptText = AES256Service.instance.bin2hex(cipher);
    bool result = encryptText == "912bef2d15a97816060bee1feaa49afe";
    expect(result, false);
  });
 });
}
```

Додаток Г. Результати інтеграційного тестування

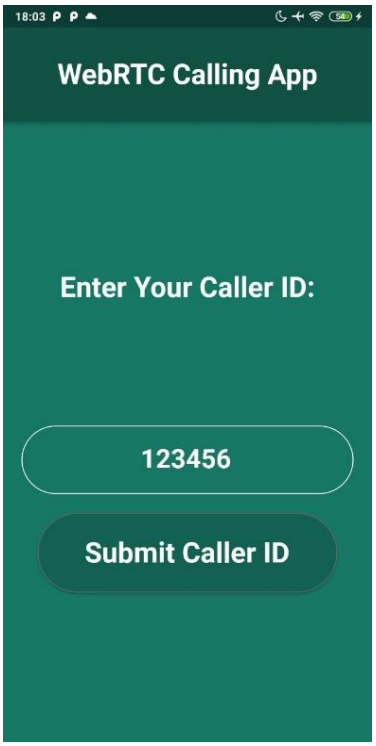


Рисунок Г.1 – Результат роботи тестового скріна для вводу id

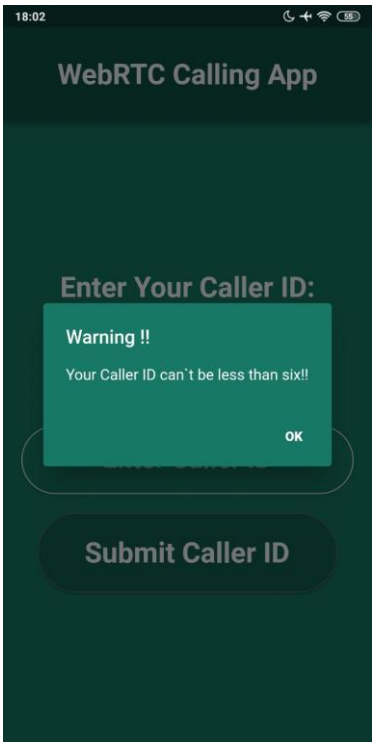


Рисунок Г.2 – Результат спливаючого повідомлення про невдалу валідацію

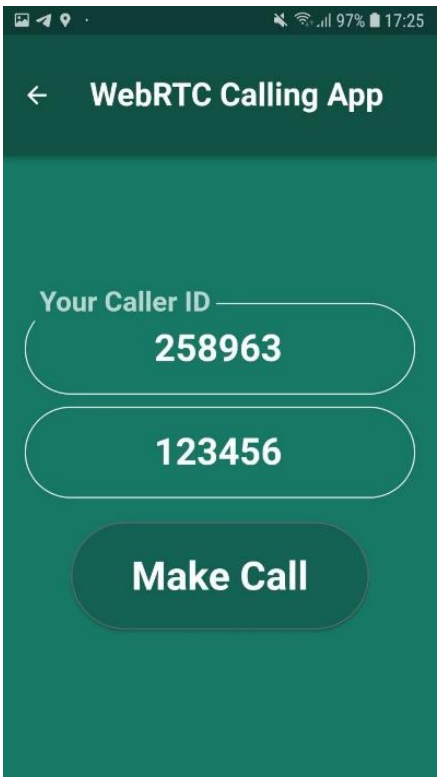


Рисунок Г.3 – Результат роботи скріна надсилання виклику

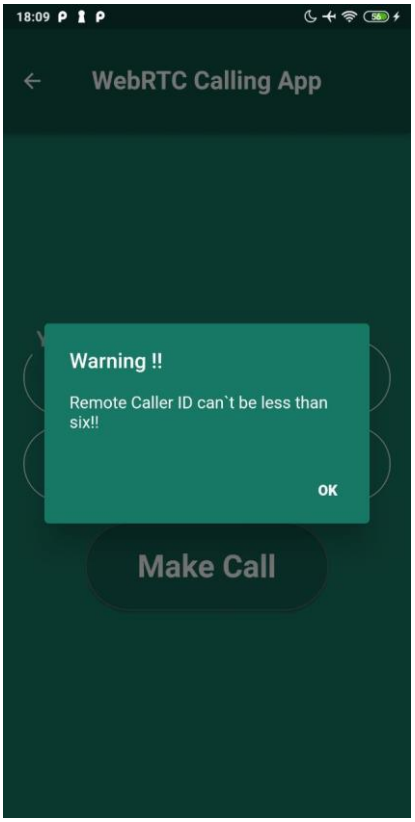


Рисунок Г.4 – Результат роботи невдалої валідації

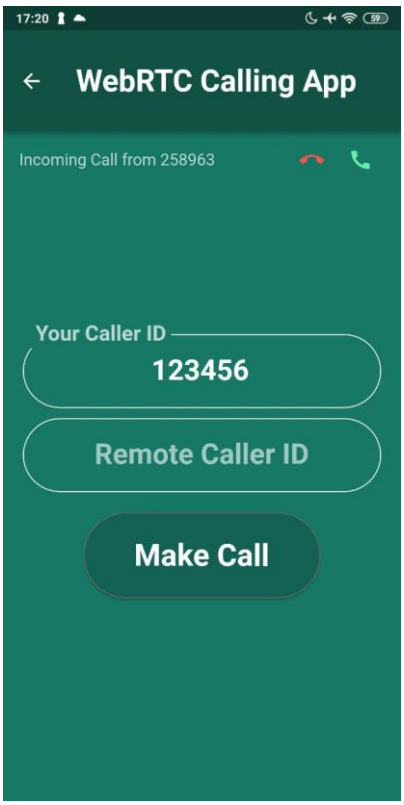


Рисунок Г.5 – Результат роботи блоку вхідного виклику

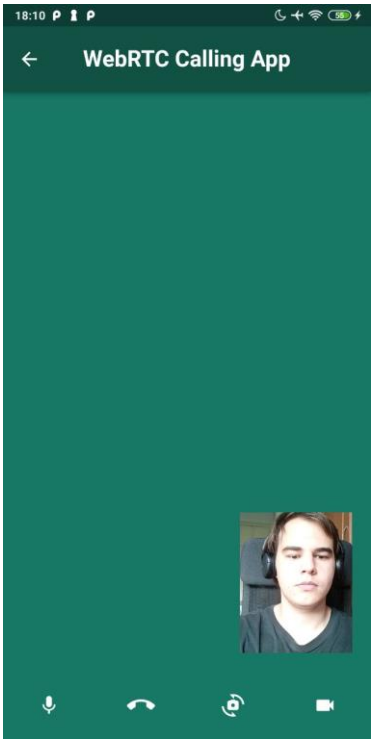


Рисунок Г.6 – Результат роботи скріна очікування

ПРОТОКОЛ ПЕРЕВІРКИ
БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб захищеної інтернет-телефонії. Частина 2. Модуль захищеного зв'язку

Автор роботи: Козак Олександр Михайлович

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 96,5%. Схожість – 3,5%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Козак О. М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Баришев Ю. В.
(підпис) (прізвище, ініціали)

ІЛЮСТРАТИВНА ЧАСТИНА

ЗАСІБ ЗАХИЩЕНОЇ ІНТЕРНЕТ-ТЕЛЕФОНІЇ. ЧАСТИНА 2. МОДУЛЬ ЗАХИЩЕНОГО ЗВ'ЯЗКУ

(Назва бакалаврської дипломної роботи)

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека

_____Олександр КОЗАК

16 червня 2023 р.

Керівник: к.т.н., доцент, доцент каф. ЗІ

_____Юрій БАРИШЕВ

16 червня 2023 р.

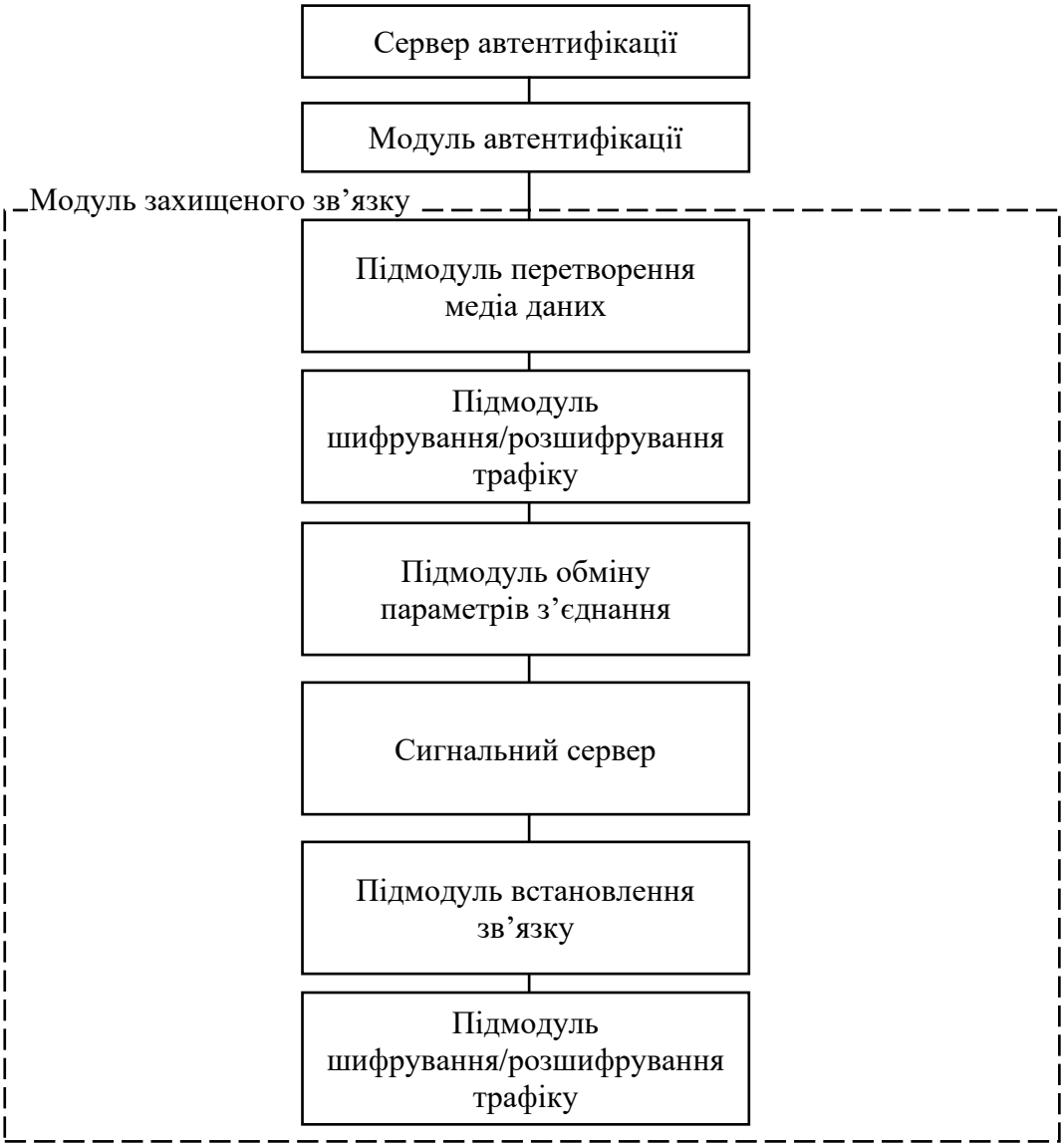
РЕЗУЛЬТАТИ ПОРІВНЯЛЬНОГО АНАЛІЗУ ЗАСОБІВ ІНТЕРНЕТ-ТЕЛЕФОНІЇ

Характеристика	WhatsApp	Signal	Skype	Zoom	Vonage
1	2	3	4	5	6
Збереження анонімності користувачів	-	+	-	-	+
Наявність відкритого коду	-	+	-	-	-
Перевірка двоетапної автентифікації	+	-	-	+	+
Інтеграція засобу для бізнесу	-	-	+	+	+
Контроль доступу до конференцій	-	-	-	+	+
Криптографічне шифрування	E2EE	E2EE	AES-256 або E2EE з обмеженнями	AES-256 або E2EE з обмеженнями	Шифрування TLS

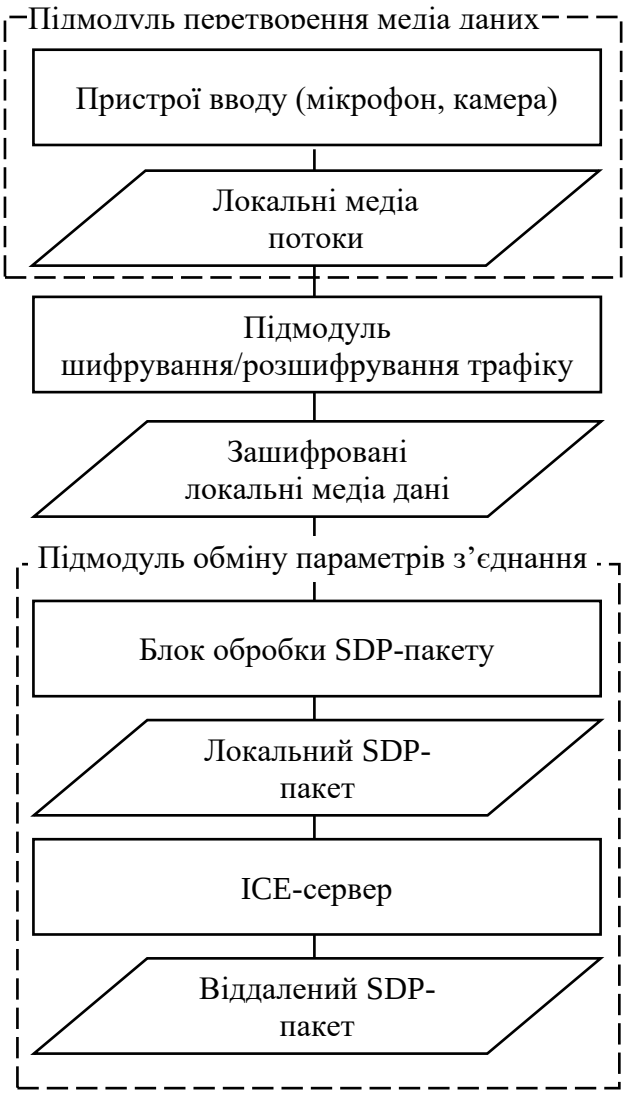
РЕЗУЛЬТАТИ ПОРІВНЯЛЬНОГО АНАЛІЗУ МЕТОДІВ ШИФРУВАННЯ

Характеристика	AES	Калина	Camellia	DES
1	2	3	4	5
Розмір блоку	128	128	128	64
Розмір ключа	128, 192, 256	128, 256, 512	128, 192, 256	56
Режими роботи	ECB, CBC, CTR, CFB, OFB	ECB, CTR, CFB	ECB, CBC, CTR, CFB, OFB	ECB, CBC, CFB
Рік створення	2001	2015	2000	1975
Швидкість шифрування	Висока	Помірна	Висока	Повільна
Безпека шифрування	Висока (стандарт США)	Висока (стандарт України)	Висока	Низька

АРХІТЕКТУРА ЗАСОБУ ЗАХИЩЕНОЇ ІНТЕРНЕТ-ТЕЛЕФОНІЇ

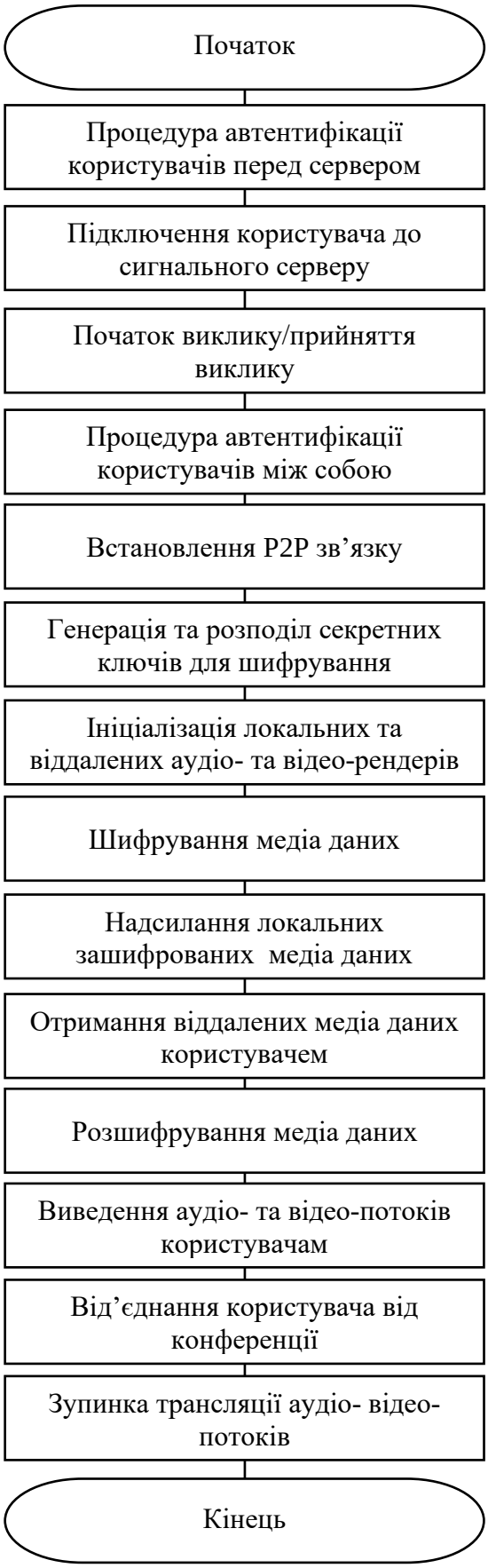


СТРУКТУРА МОДУЛЯ ЗАХИЩЕНОГО ЗВ’ЯЗКУ



СТРУКТУРА БЛОКУ ШИФРУВАННЯ ТРАФІКУ

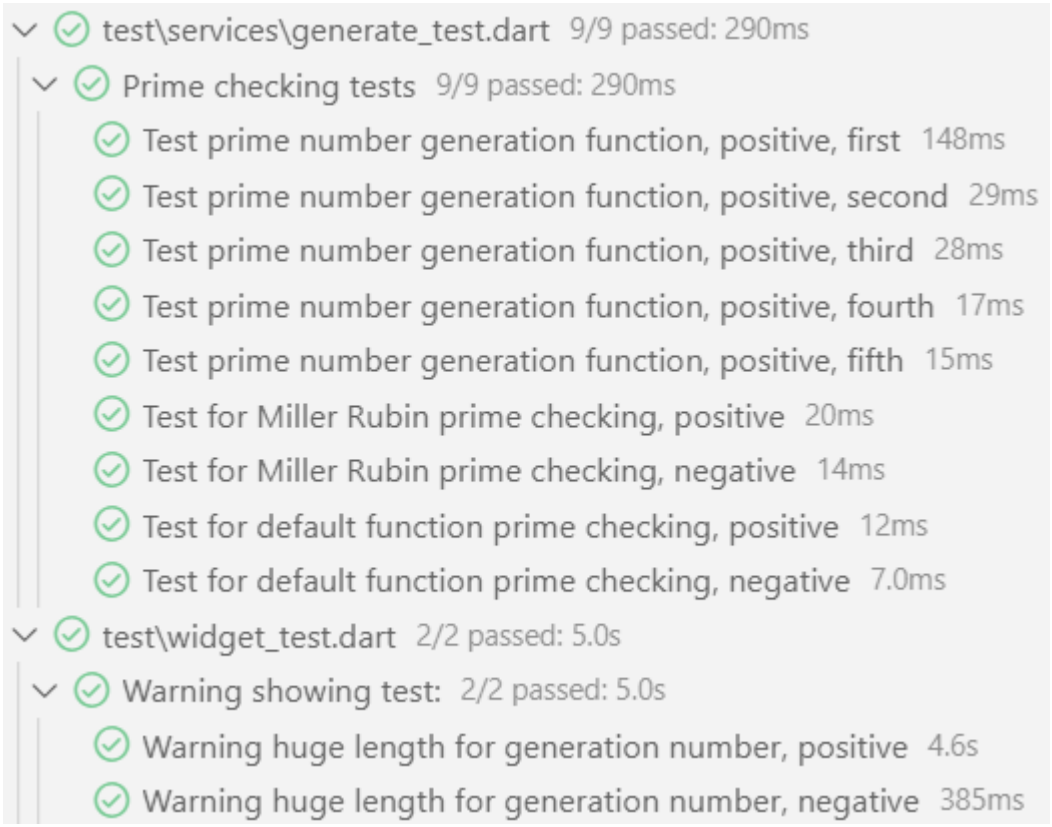
УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ ЗАСОБУ

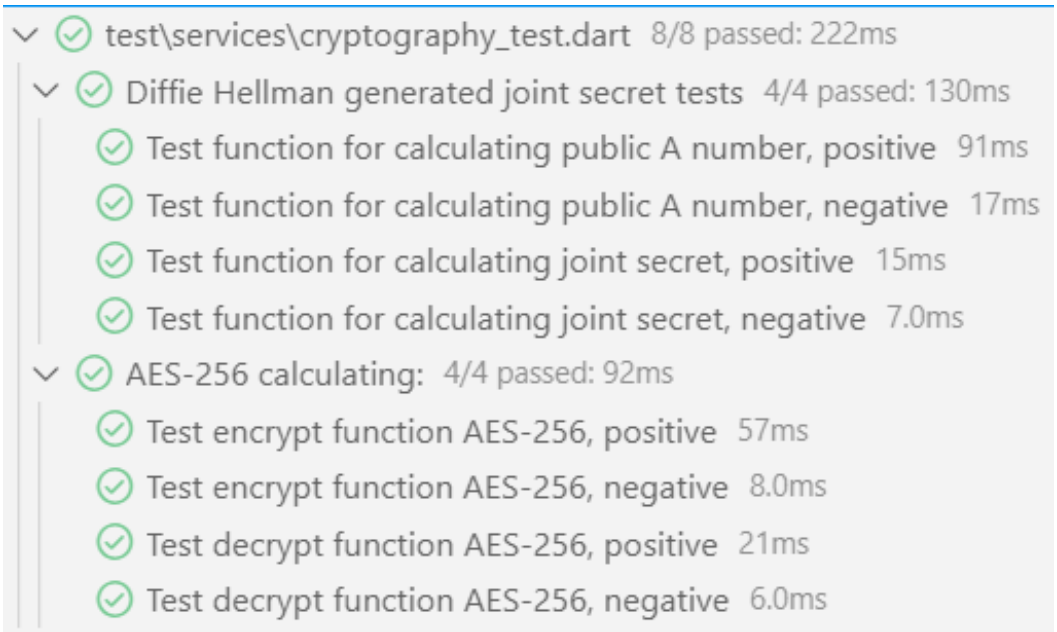


АЛГОРИТМ БЛОКУ ПЕРЕДАВАННЯ ДАНИХ

АЛГОРИТМ РОБОТИ БЛОКУ КРИПТОГРАФІЧНОГО ЗАХИСТУ

РЕЗУЛЬТАТИ БЛОКОВОГО ТЕСТУВАННЯ





РЕЗУЛЬТАТИ ІНТЕГРАЦІЙНОГО ТЕСТУВАННЯ