Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Фізико-технічний факультет
Кафедра комп'ютерної інженерії та електроніки

Луців Дмитро Михайлович
Lutsiv Dmytro

УДК 004:056.55

Спеціальність 123 «Комп'ютерна інженерія»

Кваліфікаційна робота
на здобуття освітнього ступеня бакалавр

Розробка прикладної програми з використанням криптографічного захисту інформації

Development of application using cryptography information security

Науковий керівник:
Професор Запухляк Р.І.
Рецензент:
Доц. Ліщинський І.М.

Івано-Франківськ
2020

Формат	Поз.	Позначення	Найменування	К-ть	Прим.
А4			Пояснювальна записка	66	

Змн.	Арк.	№ докум.	Підпис	Дата	123.КІ-41.17		
Розробив		Луців Д. М.			Специфікація	Літ.	Арк. 2 / Аркушів 1
Перевірив		Запухляк Р. І.					
Н. Контр.							
Затвердив							

АНОТАЦІЯ

В бакалаврській дипломній роботі розроблено захищений месенджер для миттєвого обміну повідомленнями між людьми. Додаток працює на операційній системі Android. Швидкий обмін повідомленнями забезпечується безкоштовним рішенням компанії Google - Firebase Cloud Messaging, а повідомлення зберігаються в хмарному сховищі Firebase Database.

Захист повідомлень гарантує процес наскрізного шифрування з використанням шифрування повідомлень алгоритмом AES. Криптографічний алгоритм на сьогоднішній день неможливо зламати і тому це є ідеальним вибором для шифрування.

В роботі описано методи збереження конфіденційності інформації та існуючі аналоги безпечних додатків для обміну повідомленнями. Проведено програмну реалізацію додатку шифрування та обміну повідомлень.

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		ЛуцівД.М.			Анотація	*Літ.*	*Арк.* 3 / *Аркушів* 1
Перевірив		Запухляк Р. І.					
Н. Контр.		.					
Затвердив							

ABSTRACT

In the bachelor's thesis has developed a secure messenger for instant messaging between people. The application runs on the Android operating system. Fast messaging is provided by Google's free solution - Firebase Cloud Messaging, and messages are stored in the Firebase Database cloud storage.

Message security guarantees the End-to-End Encryption process using AES algorithm encryption of messages. The cryptographic algorithm cannot be hacked for today, so it is an ideal choice for encryption.

The thesis describes methods for maintaining the confidentiality of information and existing analogues of secure messaging applications. Software implementation of the application, encryption and messaging has been carried out.

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		ЛуцівД. М.			Abstract	*Літ.*	*Арк.* 4 / *Аркушів* 1
Перевірив		Запухляк Р. І.					
Н. Контр.		.					
Затвердив							

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Фізико-технічний факультет
Кафедра «Комп’ютерної інженерії та електроніки»

Пояснювальна записка
до кваліфікаційної роботи на тему:

Розробка прикладної програми з використанням криптографічного захисту інформації

					123.КІ-41.17		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		Луців Д. М.			. Пояснювальна записка	*Літ.*	*Арк.* 5 / *Аркушів* 63
Перевірив		*Запухляк Р. І.*					
Н. Контр.		.					
Затвердив							

ЗМІСТ

ВСТУП

Кожного дня величезна кількість людей обмінюються повідомленнями через мобільні додатки. Так звані android месенджери є одними з найпопулярніших програм, які установлені на смартфонах користувачів. На сьогоднішній час безпека особистих даних та конфіденційності інформації, яка передається такими додатками, викликає все більше занепокоєння.

Щоб вирішити проблему необхідно якось шифрувати дані користувачів. Ідеальним способом збереження конфіденційних даних в такому випадку постає використання наскрізного шифрування. Такий метод шифрування практично не навантажує пристрій користувача і не займає великої кількості часу. На сьогоднішній день правильно реалізоване наскрізне шифрування неможливо.

Великою проблемою є те, що в багатьох популярних месенджерах такого шифрування взагалі немає, або не увімкнене за замовчуванням. Тому, питання створення безпечних аналогів таких додатків з схожим зручним функціоналом та дизайном є більш ніж актуальним.

Метою роботи є розробка програмного продукту android месенджеру з алгоритмом наскрізного шифрування.

Об'єкт дослідження: алгоритми шифрування та забезпечення конфіденційності даних під час надсилання, технології хмарного збереження інформації.

Предмет дослідження: проектування та розробка android додатку.

1. Постановка задачі та огляд аналогічних програмних продуктів, призначених для обміну повідомленнями

1.1. Дослідження принципів забезпечення безпеки приватної інформації на мобільних пристроях

Смартфони, по своїй суті, мають погану приватність. Вони є пристроями для відстеження, які завжди підключені до стаціонарних вишок і GPS-супутників. Увесь час, активне відстеження cookie файлів браузера, ідентифікатори реклами та статистики використання стежать за кожною людиною. Тож немає такого поняття, як ідеально безпечний і насправді приватний смартфон чи будь-який інший мобільний пристрій.

Смартфони в сучасному світі використовуються для зберігання особистої інформації, для комунікації з іншими людьми, отримання даних з мережі Інтернет у вигляді фото, відео, аудіо, тощо. Особливу стурбованість викликає безпека особистої та ділової інформації, яка зараз зберігається на пристроях.

Все більше користувачів та компаній використовують месенджери і соціальні мережі для спілкування, планування та організації роботи своїх працівників, а також для особистого життя. В компаніях такі введені технології викликають серйозні зміни в організації та методах обміну інформацією, через це вони і стали джерелом потенційно нових ризиків. Справді, смартфони зберігають все більший обсяг даних про користувача та людей чи компаній з якими він спілкується, доступ до таких даних повинен бути тільки у власника пристрою для захисту конфіденційності користувачів й інтелектуальної власності компаній.

Також смартфони, як і комп’ютери, є частими об'єктами атак. Ці атаки застосовуються на слабкі сторони, що притаманні смартфонам, і можуть надходити з режиму зв'язку. Наприклад, служба коротких повідомлень (SMS, також текстові повідомлення), послуги мультимедійних повідомлень (MMS), Wi-Fi, Bluetooth GSM та багато інших, фактично глобальний стандарт

мобільного зв'язку. Також існують атаки, що спрямовані на вразливості програмного забезпечення в браузері або операційній системі. А деяке шкідливе програмне забезпечення покладається на слабкі знання середньостатистичного користувача, який не дуже просунутий з інформаційною безпекою.

Контрзаходи щодо безпеки розробляються та застосовуються до смартфонів постійно, від налаштувань безпеки в програмних рівнях, до розповсюдження застережливої інформації для кінцевих користувачів. Розроблені хороші практики, яких необхідно дотримуватись на всіх рівнях розробки - від дизайну інтерфейсу, до розробки операційних систем, проміжних програмних шарів і додатків, що необхідно завантажувати з мережі Інтернет.

З кожним роком у світі все більше і більше людей користуються смартфонами. За даними німецького інтернет-порталу статистики Statista [1], на березень 2020 року, нараховується 4.78 млрд унікальних користувачів мобільних телефонів, з яких 3.5 млрд користуються смартфонами (рис. 1.1).

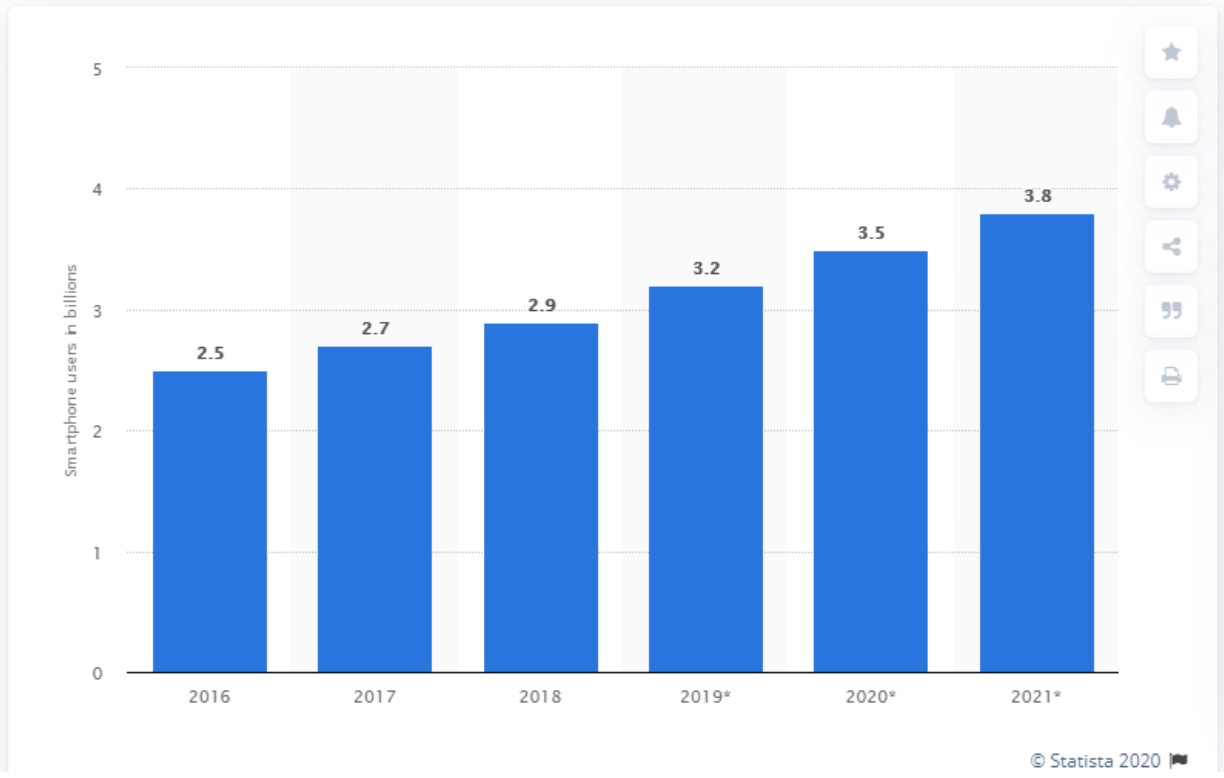


Рис.1.1. Статистика унікальних користувачів смартфонів (у млрд).

Ця статистика говорить про необхідність захисту інформації, що зберігається на мобільних пристроях, оскільки зловмисників, що хочуть отримати конфіденційну інформацію, у корисних для себе цілях, колосальна кількість.

1.1.1. Безпека даних, які зберігаються на мобільному пристрої

Ми довіряємо своїм телефонам, оскільки вони дозволяють нам підтримувати зв'язок з іншими людьми, зберігати наші паролі та дозволяють встановлювати цікаві та корисні програми. Саме тому смартфони під постійною загрозою. Зловмисники хочуть отримати нашу особисту інформацію, наприклад номер мобільного телефону або адресу проживання. Вони також орієнтуються на паролі та PIN-коди в надії отримати доступ до банківських рахунків.

В першу чергу, звичайно, потрібно уберегти свій пристрій від шкідливих додатків та вірусів. Згідно з посиланням на некомерційну правозахисну організацію Electronic Frontier Foundation (EFF) [2] це може статись з пристроєм завдяки користувачу, який сам того не знаючи, інсталює шкідливе програмне забезпечення (далі ПЗ), або завдяки злочинцю, який встановить ПЗ, використовуючи "дірки" в додатках або просто отримавши фізичний доступ до смартфону (з огляду на розміри сучасних пристроїв, такий варіант цілком реальний).

В якості загального захисту від вірусів можна використовувати мобільні антивіруси. У таблиці 1.1. на основі інформації із загальнодоступної універсальної інтернет-енциклопедії "Вікіпедія" [3] порівнюються мобільні антивіруси компаній Avast, Avira, Dr. Web, ESET, Kaspersky Lab, McAfee, 360 Mobilesafe, Norton для захисту від небезпечних додатків для операційної системи Android. Окрім, наведених в таблиці функцій, всі антивіруси мають можливість захисту пристрою в режимі реального часу та перевірку пристрою за потребою користувача.

Таблиця 1.1.

Порівняльна таблиця для мобільних антивірусів

ПЗ	Брандмауер	Веб-захист	Керування дозволами програми	Віддалене керування пристроєм	Фільтрація дзвінків та смс
Avast Mobile Security	Так	Так	Так	Так	Так
Avira Free Android Security	Ні	Ні	Ні	Ні	Так
Dr. Web Mobile Security Suite	Так	Так	Ні	Так	Так
ESET Mobile Security	Ні	Частково, тільки антифішинг	Так	Так	Так
Kaspersky Mobile Security	Ні	Так	Ні	Так	Так
McAfee Mobile Security	Ні	Так	Так	Так	Так
360 Mobilesafe	Так	Так	Так	Так	Так
Norton Mobile Security	Ні	Так	Ні	Так	Так

Вразливості встановленого по замовчуванню ПЗ, на жаль, може виправити тільки сам розробник, користувач в такому випадку нічого не може зробити. Все, що в його силах, це своєчасно оновлювати встановлене ПЗ та ОС до останніх версій.

З іншого боку, щоб обмежити доступ до пристрою і даних які в ньому зберігаються, необхідно увімкнути блокування екрану. Воно може бути у вигляді пін-коду, графічного паролю. Використання сканування відбитків пальців та ідентифікації обличчя також є відмінним варіантом, оскільки це швидше і простіше, ніж запам'ятовування коду розблокування. Також можна захистити паролем усі мобільні додатки, які містять особисті дані, такі як банківські додатки, електронна пошта і таке інше.

Звичайно, все це не забезпечує максимального захисту і завжди необхідно бути трохи уважним по відношенню до своїх дій з мобільними пристроями, особливо в людних місцях.

1.1.2 Інформація про місцезнаходження пристрою

Інформація про місцезнаходження людини є особистою інформацією і також з деякою точністю визначається за допомогою мобільних пристроїв, які отримують і відправляють постійно сигнали в мережу. Є принаймні 4 способи дізнатися місце розташування пристрою [4]:

1. Визначити місце за стільниковими вежами.

Усі сучасні мобільні оператори можуть вирахувати, де знаходиться телефон конкретної людини, поки він працює і зареєстрований в мережі. Можливість це робити є результатом того, як побудована мобільна мережа, і цей спосіб називається тріангуляцією [5]. Для цього їм необхідно визначити потужність сигналу від мобільного пристрою на різних вишках. Точність залежить від багатьох факторів, наприклад, використовуваних технологій і кількості стільникових веж в районі абонента, але зазвичай в місці розташування пристрою таким способом можна визначити з точністю до кількох десятків метрів. Не можна сховатися від такого відстеження, якщо ваш мобільний телефон увімкнено та в ньому активна сім-карта, вона передає сигнали в мережу оператора. Хоча зазвичай лише мобільний оператор може здійснювати такий спосіб відстеження (рис. 1.2), також змусити оператора передати дані про місцезнаходження користувача (в режимі реального часу або як запис в певний час) можуть правоохоронні органи.

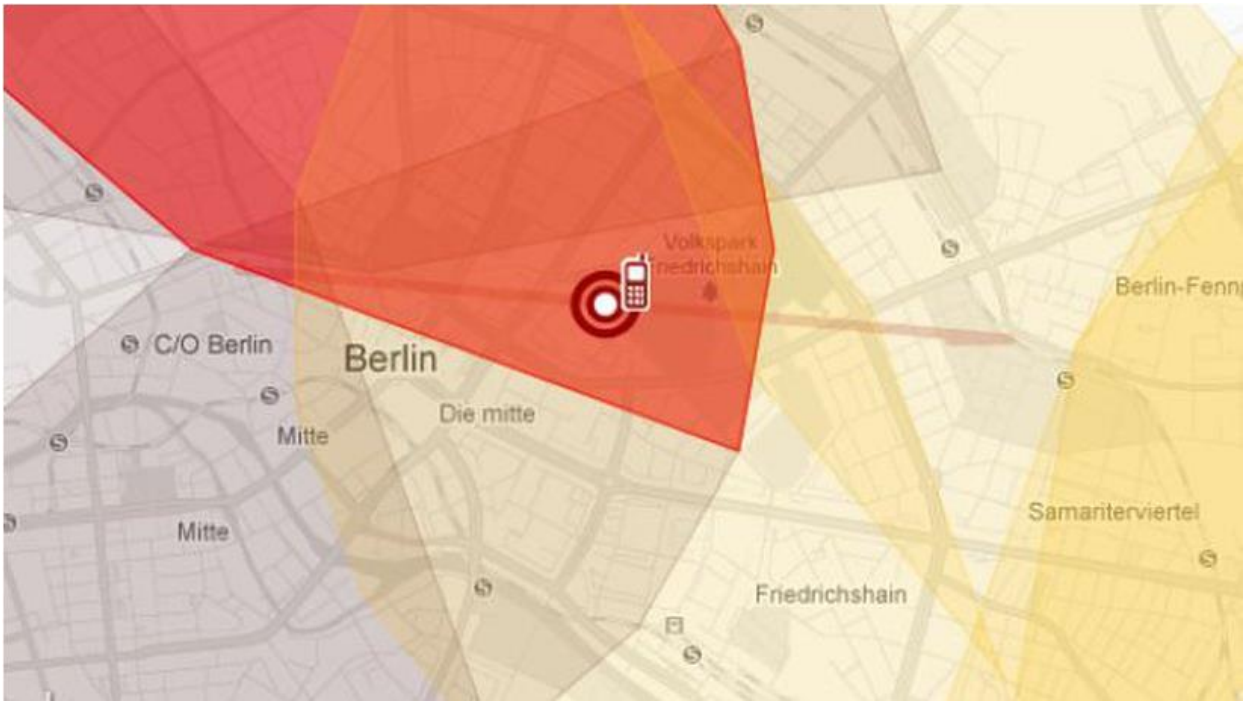


Рис. 1.2. Визначення місцезнаходження людини методом тріангуляції.

2. IMSI-пастка, або симулятор стільникового сайту.

Такі пристрої використовуються урядом чи іншими технічно розвиненими організаціями, які можуть дозволити собі техніку високого рівня складності. IMSI (International Mobile Subscriber Identity) посилається на номер ідентифікатора міжнародного мобільного абонента, який ідентифікує SIM-карту конкретного абонента. Принцип дії такий, що пастка "прикидається" справжньою стільникового вежею, а мобільні пристрої, що розташовані близько до неї, підключаються і передають дані через неї (рис.1.3). Таким чином, ці пристрої дозволяють стежити не тільки за місцем розташування, а й за контактами підключених людей. В даний час немає надійної оборони від усіх IMSI-пасток.

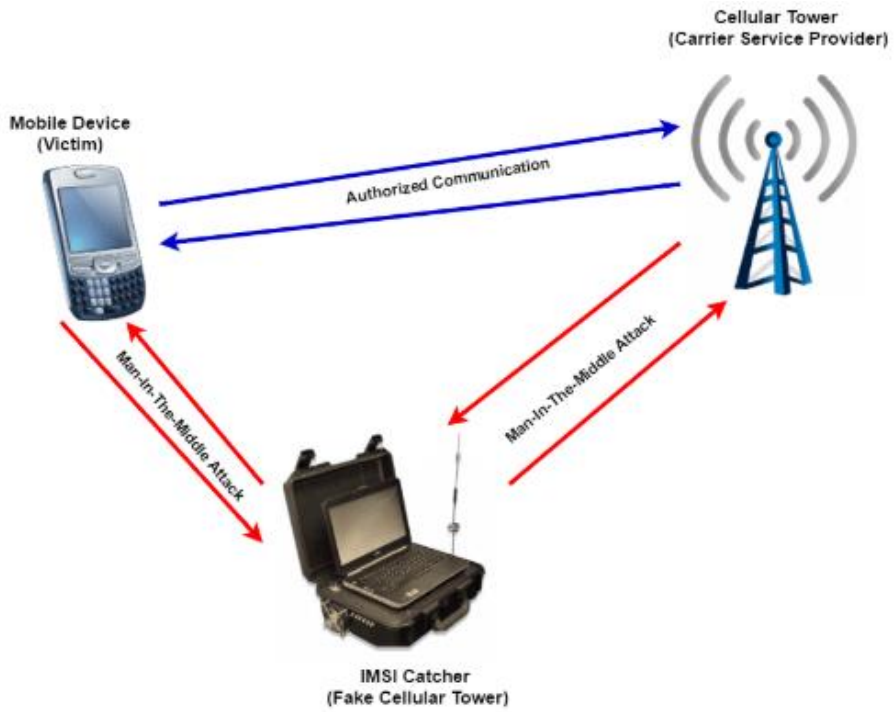


Рис. 1.3. IMSI пастка.

3. Відслідковування сигналів Wi-Fi і Bluetooth.

Всі сучасні мобільні пристрої, включаючи навіть розумні годинники оснащені цими радіопередавачами. Всі передавачі мають свою унікальну MAC-адресу, яку не можна змінити на більшості мобільних пристроїв. Відмінність від GSM-мереж полягає в тому, що радіус дії Wi-Fi і Bluetooth істотно менший, тобто треба знати точне місце розташування жертви, а також MAC-адреси передавачів, які встановлені в її мобільному пристрої. До плюсів такого способу варто віднести те, що місце розташування можна визначити з більшою точністю або, наприклад, можна визначати, коли особа входить в будівлю і коли виходить з неї.

4. Витік даних про місцезнаходження під час веб-серфінгу.

Сучасні мобільні пристрої пропонують самі визначити своє місце розташування, часто використовуючи GPS-сигнал або за допомогою інших сервісів, які допомагають визначити місце розташування пристрою на основі даних від стільникових веж або Wi-Fi-мереж. Додатки можуть отримувати

доступ до цих даних з дозволу користувача. Деякі додатки після цього відправляють дані про місцезнаходження користувача в мережу, завдяки чому у третіх осіб з'являється можливість стежити за місцем розташування інших людей.

Інформація про місця, в яких знаходяться пристрої, може служити не тільки для того, щоб безпосередньо визначити місце розташування людини, але, і щоб визначати, які відносини можуть пов'язувати різних людей. Наприклад, з великою ймовірністю за цими даними можна з’ясувати, хто відвідував певну зустріч, чи хто був на певному протесті, або спробувати визначити конфіденційне джерело журналіста.

На жаль, звичайні люди не в змозі як-небудь протистояти першим двом способам (можливо тільки використовувати одноразові телефони), але користувачі завжди можуть перевіряти, чи вимкнений Wi-Fi і Bluetooth поза межами свого будинку, а також ретельно стежити за додатками, яким вони надають доступ до даних про своє місцезнаходження. У цьому, до речі, можуть допомогти антивіруси, розглянуті в розділі 1.1.1.

1.1.3 Безпека даних, яка передається між мобільними пристроями в мережі Інтернет

Люди можуть обмінюватись текстовими повідомленнями, зображеннями і т.п., наприклад, через соціальні мережі, послуги хмарного зберігання та електронну пошту.

Хоча й існує багато способів обміну та передачі файлів в Інтернеті , але не так багато способів зробити це надійно, навіть при обміні великими файлами. Багато існуючих програмних платформ можуть пропонувати додаткові функції, такі як захист паролем документів, але це не стосується всіх файлів або папок, які ви можете надіслати. Потрібна певна форма шифрування, щоб забезпечувати приватність всіх спільних файлів.

Це може бути надзвичайно важливим для бізнесу, його зазвичай не розглядають як щось важливе - але все частіше хакерські атаки, що викрадають особисті та конфіденційні дані навіть великих компаній, означають, що великій кількості людей необхідно захистити свої дані в мережі.

Існує ряд методів, якими розробники можуть скористатися безпосередньо, такі як PGP, GPG та E2EE, які використовують методи криптографії з відкритим ключем, а також SSH та SFTP.

Ситуація з безпекою змінюється, якщо людина починає використовувати додатки для передачі даних в інтернеті. В даному випадку, ступінь захищеності переданої інформації залежить від засобів шифрування, які використовує додаток. Ці засоби шифрування можуть бути побудованими на основі будь-яких доступних способів шифрування. Важливою умовою якісного криптозахисту додатку є присутність наскрізного шифрування і відсутність можливості у розробника і власників серверів переглядати передані дані.

Зробивши висновок з усього вищесказаного, я продовжив роботу саме над передачею даних в мережі Інтернет за допомогою месенджерів для смартфонів, так як саме ця область в сфері захисту особистої інформації найбільше залежить від вибору програмного забезпечення і надає можливість забезпечити необхідний рівень захисту.

1.2. Аналіз забезпечення конфіденційності в месенджерах

1.2.1. Вимоги до безпечного мобільного месенджеру

Вибір додатку для обміну повідомленнями частіше за все залежить від критеріїв, які мають різну важливість для різних мас користувачів, а також мають певний вплив на певний рівень безпеки. Такими критеріями є:

1. При використовуванні мобільного інтернету постачальник послуг не повинен мати доступ до даних, які передаються його сервісами.

2. Забезпечення безпечного сеансу або TLS. Безпечний сеанс - це унікальний ключ для кожного сеансу. Необхідно слідкувати за тим, щоб спілкування відбувалось з потрібною людиною, і ніяка третя особа не змогла прочитати повідомлення.
3. Повідомлення повинні бути зашифровані для підтримки безпеки та конфіденційності.
4. Не дозволяється обмінюватися повідомленнями, якщо користувачі не є друзями. Тобто вони впевнені, що спілкуються саме з тою особою.

1.2.2. Опис та принцип роботи наскрізного шифрування

Наскрізне шифрування (End-to-End Encryption) - це захищений і приватний спосіб спілкування, коли єдиний, хто може отримати доступ до даних - це відправник і конкретний одержувач. Використання повноцінного шифрування запобігає хакерам чи небажаним стороннім особам отримати доступ до повідомлень або файлів на сервері [5].

У правильному наскрізному шифруванні воно відбувається на рівні пристрою. Тобто, кожне повідомлення або файл шифрується до того, як він покине телефон чи комп'ютер, і не розшифровується, поки не досягне свого місця призначення. Як результат, хакери не можуть отримати доступ до даних на сервері і на проміжних станах, оскільки вони не мають приватних ключів для розшифровки даних. Натомість ключі зберігаються у окремого користувача на своєму пристрої, що значно ускладнює доступ до даних особи.

Захист такого типу шифрування забезпечується створенням пари публічно-приватних ключів. Цей процес, також відомий, як асиметрична криптографія, використовує окремі криптографічні ключі для блокування та шифрування повідомлення. Відкриті ключі широко розповсюджуються та використовуються для блокування чи шифрування повідомлення. Приватні ж

ключі відомі тільки власнику і використовуються для розблокування або розшифрування повідомлення.

В наскрізному шифруванні певний програмний продукт створює публічний і приватний криптографічні ключі для співрозмовника.

Щоб наглядно зрозуміти, як працює таке шифрування давайте змоделюємо ситуацію. Користувач 1 і користувач 2 зареєструвались в системі обміну повідомленнями. За допомогою певного способу шифрування і математики для них генеруються по парі публічно-приватних ключів.

Для того, щоб перший користувач надіслав зашифроване повідомлення він запитує відкритий ключ другого користувача і шифрує своє повідомлення за допомогою отриманого публічного ключа. Тоді другий користувач отримує зашифроване повідомлення, бере свій приватний ключ на своєму пристрої, розшифровує повідомлення і бачить його так, як перший користувач хотів надіслати.

Повідомлення від першого користувача до другого може пройти по дорозі через кілька серверів електронних пошт чи інших хостингових серверів. Навіть якщо компанії, які володіють сервером, спробують прочитати повідомлення, вони не зможуть цього зробити, оскільки наскрізне шифрування забезпечує безпеку повідомлення і без приватного ключа іншого користувача неможливо отримати вміст повідомлення. Лише користувач два зможе розшифрувати повідомлення, оскільки він єдиний має відповідний приватний ключ, який може розшифрувати повідомлення.

Якщо користувач два хоче відповісти, він просто повторює процес, шифруючи своє повідомлення користувачу один, використовуючи його відкритий ключ.

Таке шифрування надійно передає дані між кінцевими точками. Однак зловмисник може використати хитрий прийом: стати на стороні сервера і видати себе за одержувача, замінивши відкритий ключ одержувача на свій. Таким чином повідомлення шифрується ключем, відомим для зловмисника.

Однак, використовуючи систему аутентифікації, такої проблеми можна уникнути. Автентичність забезпечується тим, що користувач один робить цифровий підпис повідомлення іншому за допомогою свого приватного ключа. Коли користувач два отримує повідомлення від першого, то він може перевірити цифровий підпис отриманого повідомлення, використовуючи його відкритий ключ. Оскільки цифровий підпис базується на приватному ключі, користувач один є єдиним, хто міг створити цей підпис. Таким чином, не існує способу підробляти такий вид ідентифікації відправника.

В сучасних додатках і системах ці перевірки виконуються автоматично і настільки швидко, що користувачі не підозрюють про такі механізми безпеки їх даних.

1.3 Порівняльний аналіз існуючих android месенджерів

На початку 2020 року доступні десятки програм для обміну повідомленнями, які позиціонують себе, як захищені. Переглянувши масу доступних програм для безпечного обміну повідомленнями, що відповідають наступним основним характеристикам:

- Незалежність від великих технологічних компаній
- Наскрізне шифрування (End-to-End Encryption)
- Підтримка різних платформ (доступність на різних пристроях)
- Кілька типів обміну інформації (текстова, голосова, зображення, відео і т.д.)

Для аналізу я вибрав 4 такі мобільні месенджери:

1. Signal
2. Telegram
3. Viber
4. WhatsApp

В таблиці 1.2. представлено результат порівняння.

Таблиця 1.2.

Порівняльна таблиця месенджерів

Назва месенджеру	**Наскрізне шифрування (End-to-End Encryption)**	**Тимчасові ключі шифрування**	**Чи відкритий вихідний код для незалежного огляду?**	**Чи використовує додаток TLS / Noise для шифрування мережевого трафіку?**	**Чи можна отримати доступ до старих повідомлень в разі взлому аккаунта?**	**Чи добре задокументовано протокол безпеки?**
Signal	+	+	+	+	-	+
Viber	+	+	-	+	-	-
WhatsApp	+	+	-	+	+	+
Telegram (в режимі секретного чату)	+	+	+	-	+	+

Список популярних месенджерів які не підтримують наскрізне шифрування, чи воно не увімкнене по замовчуванню: Discord, Facebook Messenger, ICQ, Line, Messages (Apple), Skype, Snapchat, Telegram, Viber, WeChat.

1.3.1 Signal

Історія цього мобільного додатку почалася в 2010-му році із створення двох додатків: TextSecure для захисту кореспонденції і RedPhone для здійснення захищених голосових викликів по мережі Інтернет. З самого початку розробники позиціонували своє програмне забезпечення як конфіденційне і здатне захистити особисті дані від атак посередника шляхом наскрізного шифрування. У 2013-му році TextSecure і RedPhone були об'єднані в один додаток - Signal. Одночасно додаток анонсували для iOS.

Розробником додатків є компанія Open Whisper Systems. Вона має свій репозиторій на порталі GitHub [6], куди викладає всі вихідні коди своїх проектів. Це робиться для того, щоб спільнота незалежних розробників могла

інспектувати код на наявність вразливостей. Також будь-хто може зібрати додаток з вихідного коду і перевірити, чи відрізняється він від того, що поширюється за допомогою офіційних магазинів додатків. Ця організація відома ще тим, що всі її продукти рекомендував Едвард Сноуден.

Для шифрування використовується Signal Protocol. Раніше цей протокол носив назву TextSecure Protocol із зрозумілих причин.

Загальний принцип дії протоколу Signal Protocol:

Згідно з документацією[7], ключі генеруються за допомогою найшвидшої еліптичної кривої Curve25519 (рис.1.4). Ця крива є криптографічним протоколом, що дозволяє двом сторонам, які мають пари відкритий/закритий ключі на еліптичних кривих, отримати загальний секретний ключ, використовуючи незахищений від прослуховування канал зв'язку. При установці додатку генеруються Identity Key Pair - довгострокова пара ключів, Signed Pre Key - середньострокова пара ключів, завірена Identity Key Pair, і One-Time Pre Keys - партія одноразових ключів. При реєстрації користувача на сервері відправляються публічні частини згенерованих ключів. Секретні ключі залишаються тільки на пристрої.

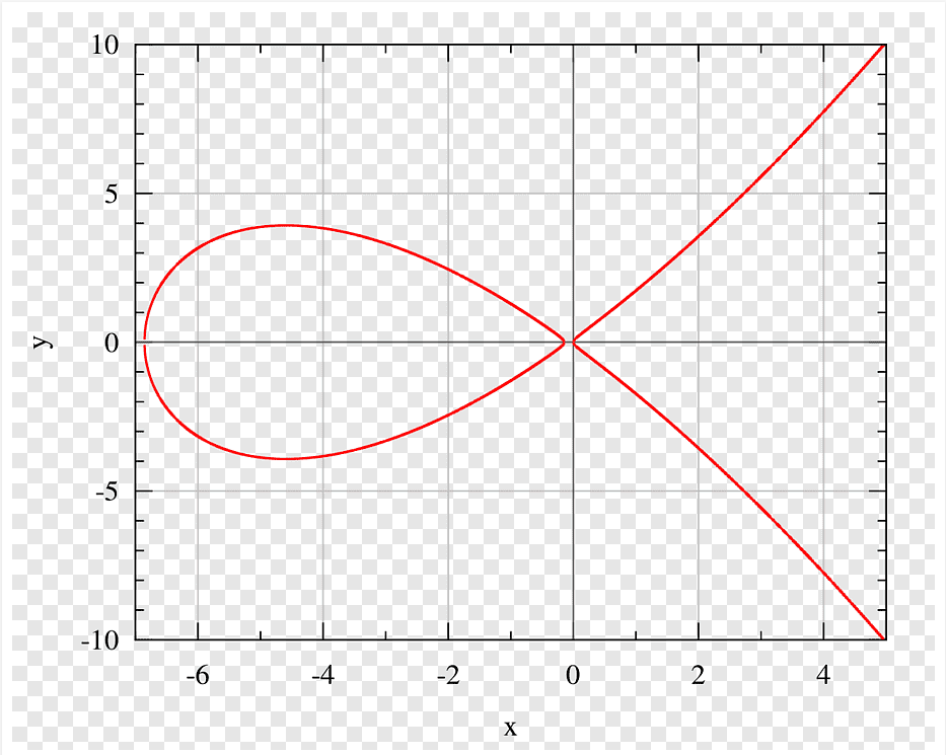


Рис.1.4. Еліптична крива Curve25519.

Для зв'язку двох користувачів по захищеному каналу для початку необхідно ініціалізувати сесію. Після старту сесії немає необхідності її перезапускати. Це потрібно тільки в разі переустановлення програми або при зміні мобільного пристрою. При ініціалізації сесії відбувається наступна послідовність дій, описана в документації [7]:

1) Ініціатор запитує Identity Key Pair, Signed Pre Key і One-Time Pre Key у одержувача повідомлень.

2) Сервер висилає запитані ключі, при цьому видаляючи One-Time Pre Key, так як ключі такого типу одноразові (якщо One-Time Pre Keys одержувача були вислані і не доповнені, то One-Time Pre Key повертається на сервер).

3) Ініціатор зберігає отримані ключі, пов'язуючи їх з одержувачем повідомлень: Identity Key як $I_{recipient}$, Signed Pre Key як $S_{recipient}$ і One-Time Pre Key як $O_{recipient}$.

4) Ініціатор генерує пару тимчасових ключів за допомогою Curve25519, $E_{initiator}$.

5) Identity Key ініціатора позначимо як $I_{initiator}$.

6) Після цього генерується майстер-ключ (ECGH, Elliptic curve Diffie-Hellman - протокол Діффі-Хеллмана на еліптичних кривих) master_secret = ECDH ($I_{initiator}$, $S_{recipient}$) || ECDH ($E_{initiator}$, $I_{recipient}$) || ECDH ($E_{initiator}$, $S_{recipient}$) || ECDH ($E_{initiator}$, $O_{recipient}$). Якщо One-Time Pre Key відсутній, то остання дія опускається.

7) Ініціатор використовує HKDF (функція формування ключів) для створення Root Key і Chain Keys з master_secret.

В результаті ініціалізації сесії генерується Root Key - 32-байтне значення, яке використовується для генерації Chain Key, що теж є 32-байтним значенням і використовується для генерації Message Key. Message Key - 80-байтне значення, яке використовується для шифрування змісту повідомлень.

Після установки довгострокової сесії, ініціатор може відразу ж починати відправляти повідомлення одержувачу. На початку всіх повідомлень буде передаватися інформація, яка необхідна одержувачу для установки сесії. Щоб встановити сесію, одержувач виконує такі завдання [7]:

1) Одержувач вирахує master-secret, використовуючи свої секретні ключі і ключі, отримані в повідомленнях.

2) Одержувач видалить One-Time Pre Key, використаний ініціатором.

3) Одержувач використовує HKDF для обчислення Root Key і Chain Keys з master_secret.

Кожне повідомлення шифрується новим Message Key, який обчислюється в такий спосіб:

1) Message Key = HMAC-SHA256 (Chain Key, 0x01).

2) Після першого кроку Chain Key оновлюється за формулою Chain Key = HMAC-SHA256 (Chain Key, 0x02).

Таким чином, Signal Protocol позбавлений недоліків завдяки тому, що всі необхідні ключі: і довготривалі, і одноразові - зберігаються на сервері. Це дозволяє домогтися легкості використання і швидкості дії.

В цілому, месенджер Signal володіє великою кількістю плюсів, при цьому має відкритий вихідний код на платформі GitHub[8]. Відкритість вихідного коду в свою чергу дозволяє говорити про дійсно надійні методи шифрування, перевірених незалежними експертами.

1.3.2 Telegram

Цей додаток створено Павлом Дуров, засновником соціальної мережі "ВКонтакте". З самого створення розробники говорили про те, що відмінною рисою даного месенджера є наявність шифрування. Чати в цій програмі діляться на звичайні і секретні. Далі я буду розглядати секретні чати, так як тільки вони в даній програмі підтримують наскрізне шифрування.

Для шифрування даних розробники використовують протокол MTProto (рис.1.5).

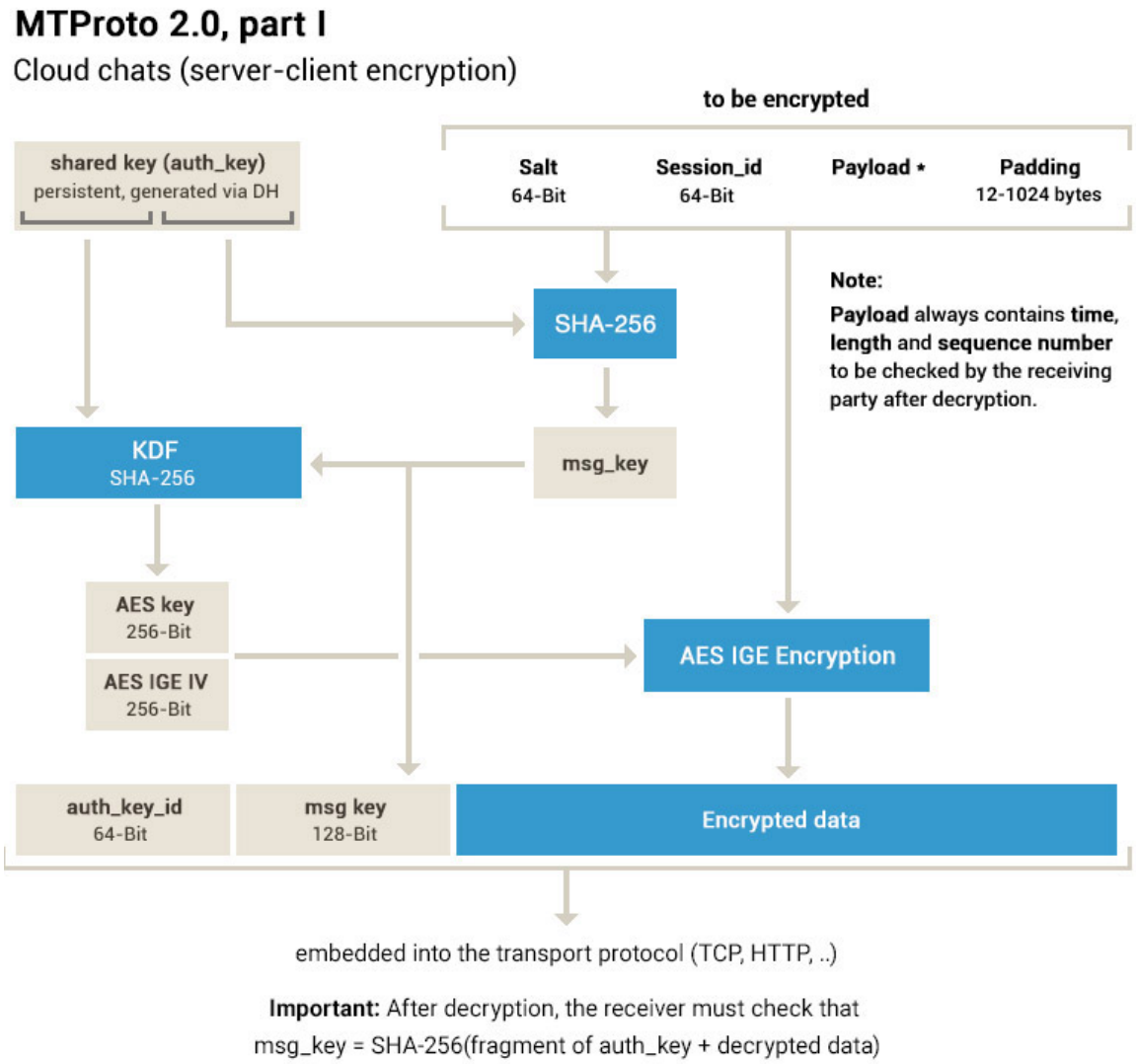


Рис. 1.5. Протокол шифрування Telegram – MTProto.

Розглянемо протокол шифрування даних в секретних чатах, який наведений у документації [9]:

1. Ініціатор запускає метод messages.getDhConfig, щоб згенерувати параметри для протоколу Діффі-Хеллмана: просте число p і елемент високого порядку g.
2. Ініціатор обчислює випадкове 2048-бітне число a, обчислює g_a: = pow (g, a) mod dh_prime і запускає messages.requestEncryption, щоб відправити запит одержувачу.

3. Одержувач приймає запит, оновлює параметри для протоколу Діффі-Хеллмана, обчислює випадкове 2048-бітове число b за тими ж правилами, що й ініціатор обчислив a.
4. Після прийняття запиту одержувач обчислює кінцевий загальний ключ за формулою key = (pow (g_a, b) mod dh_prime), потім обчислює g_b: = pow (g, b) mod dh_prime і запускає messages.acceptEncryption.
5. Після отримання g_b ініціатор також обчислює кінцевий загальний ключ за формулою key = (pow (g_b, a) mod dh_prime).
6. Далі за допомогою SHA-1 (алгоритм криптографічного хешування) обчислюються ключі шифрування:
   a) Обчислюється msg_key шляхом вибору 128 біт з SHA-1 від тексту повідомлення;
   b) x = 0;
   c) sha1_a = SHA1 (msg_key + substr (key, x, 32));
   d) sha1_b = SHA1 (substr (key, 32 + x, 16) + msg_key + substr (key, 48 + x, 16));
   e) sha1_c = SHA1 (substr (key, 64 + x, 32) + msg_key);
   f) sha1_d = SHA1 (msg_key + substr (key, 96 + x, 32));
   g) aes_key = substr (sha1_a, 0,8) + substr (sha1_b, 8,12) + substr (sha1_c, 4,12);
   h) aes_iv = substr (sha1_a, 8,12) + substr (sha1_b, 0,8) + substr (sha1_c, 16,4) + substr (sha1_d, 0,8).

Дані шифруються 256-бітовим ключем aes_key і 256-бітовим вектором ініціалізації aes_iv, використовуючи алгоритм шифрування AES-256 в режимі розширення невизначеного спотворення (infinite garble extension, IGE).

7. Відбиток ключа шифрування key_fingerprint і msg_key додаються до початку масиву байтів, після чого запускається messages. sendEncrypted.

8. Для розшифровки алгоритм з пункту 6 відтворюється в зворотному порядку.

Нові ключі в даному протоколі формуються в двох випадках: якщо старими були зашифровані 100 повідомлень або якщо старим ключам більше тижня і за допомогою них зашифрували хоча б одне повідомлення. Для формування нових ключів існує окремий протокол, описаний на сайті розробників [10] і мало чим відрізняється від наведеного вище.

Варто відзначити корисну особливість секретних чатів - в їх налаштуваннях можна вказати час автоматичного видалення повідомлень. Наприклад, поставити таймер на 10 секунд і тоді кожне повідомлення буде автоматично видалятися через 10 секунд після прочитання його іншим користувачем.

Вихідні тексти клієнтської частини даного додатка представлені на порталі GitHub [11]. Таким чином, клієнтську частину також можуть інспектувати незалежні експерти. Тексти програм серверної частини залишаються закритими і публікуватися не збираються.

З усього вищесказаного можна зробити висновок про те, що секретні чати в месенджері Telegram надають можливість організувати дійсно захищений канал зв'язку для обміну повідомленнями між користувачами.

1.3.3 Viber

Хоч не дуже і відомий на Заході, Viber - це один з найпопулярніших додатків для безпечного обміну повідомленнями у світі. За даними 99firms.com [12], Viber має понад 1 мільярд (1 000 000 000) користувачів у всьому світі . Цікаво, що більшість цих користувачів розташовані в пострадянських країнах, порівняно мало користувачів у таких місцях, як США або Європейський Союз.

Авторами Viber є Ігор Магазинник і Тальмон Марко. Реліз месенджеру був в грудні 2010 року. Додаток написаний за допомогою таких мов як: Java, C, Python, C++, Objective-C.

З виходом оновлення до версії 6.0 Viber отримав підтримку наскрізного шифрування для всіх чатів, секретні чати із аналогічними функціями, як у Telegram, налаштуваня таймеру автоматичного видалення повідомлень, постійне відстеження скріншотів і їх заборона, захист від пересилань та копіювання повідомлень і тому подібне.

На відміну від Telegram, у Viber увімкнене наскрізне шифрування за замовчуванням для всіх нових чатів.

Взагальному, з огляду на слова розробників Viber, шифрування повідомлень та даних у месенджері виконуються протоколом шифрування Signal Protocol про який детально описано вище.

Протягом декількох років тільки Viber пропонував послуги голосових та відео -дзвінків, перш ніж WhatsApp і Telegram почали надавати подібні голосові та відео послуги. Зважаючи на його попередні розробки, зрозуміло, що Viber може запропонувати кращу якість голосу із меншим рівнем шуму, незважаючи на всю пропускну здатність, порівняно з WhatsApp та Telegram. Viber також пропонує два режими якості голосу: нормальний і HD режим, останній дуже хороший для мереж з високою пропускною здатністю. Мабуть, найбільш унікальною особливістю є Viber Out, що дає змогу користувачам Viber контактувати з людьми, які не користуються сервісом.

1.3.4 WhatsApp

За різними оцінками WhatsApp є найпопулярнішим додатком для миттєвого обміну текстовими повідомленнями для мобільних та десктопних платформ, також з підтримкою голосового і відеозв’язку. Компанія WhatsApp була куплена компанією Facebook в жовтні 2014 року. Програма була розроблена на мові Erlang. Перша її версія була випущена в 2009 році.

Спочатку додаток був платним: перший рік – без оплати, всі решта по 1-2 долари за рік.

WhatsApp використовує модифікований відкритий мережевий протокол для швидкого обміну повідомленнями Extensible Messaging and Presence Protocol (XMPP). Під час установки створюється аккаунт на сервері s.whatsapp.net, який в якості імені користувача використовує номер мобільного телефону. Android версія автоматично використовує в якості пароля MD5-хеш від зміненого ідентифікатора IMEI, а iOS версія використовує MD5-хеш від MAC-адреси смартфону.

У версіях для персонального комп'ютера і веб є особливість, яка не дозволить використовувати систему, якщо на мобільному пристрої немає зв'язку з інтернетом. Також не можна бути активним одночасно у веб версії і версії для персонального комп'ютера.

В березні 2016 року, в ньому з'явилась підтримка наскрізного шифрування, що дозволяє зарахувати його до захищених, хоча спочатку його так не позиціонували. Єдиним недоліком залишається те, що вихідні коди цього месенджера закриті, і оцінити надійність шифрування не представляється можливим. Проте, можливо оцінити організацію цього шифрування, яка детально і ясно викладена в документації [13].

Як протокол шифрування використовується Signal Protocol, що розроблений організацією Open Whisper Systems. Базовий принцип його дії описаний в розділі 1.3.1. Цей протокол наскрізного шифрування призначений для запобігання доступу сторонніх осіб до відкритого доступу до повідомлень або дзвінків. Більше того, навіть якщо ключі шифрування з пристрою користувача будуть коли-небудь фізично змінені, їх більше не можна буде використовувати для дешифрування раніше переданих повідомлень.

Можна окремо виділити процедуру шифрування файлів і мультимедіа (відео, аудіо, фото і файли), так як вони теж зашифровані як і текстові повідомлення. Процес описано в документації [13]:

1. Користувач WhatsApp, що надсилає повідомлення ("відправник"), генерує ефемерний 32 байтний AES256 ключ і ефемерний 32 байтний ключ HMAC-SHA256.
2. Відправник шифрує прикріплений файл ключем AES256 в режимі CBC з випадковим IV, потім додає MAC-адресу шифротексту за допомогою HMAC-SHA256.
3. Відправник вивантажує зашифрований прикріплений файл на сервер.
4. Відправник передає одержувачу звичайне зашифроване повідомлення, яке містить: ключ шифрування, ключ HMAC, хеш SHA256 шифрованого файлу та вказівник на місце на сервері.
5. Одержувач розшифровує повідомлення, витягує зашифрований файл з серверу, перевіряє хеш SHA256, перевіряє MAC та розшифровує файл.

Таким чином, згідно з офіційною документацією, в мобільному додатку WhatsApp реалізована система наскрізного шифрування з використанням тимчасових ключів і з можливістю підтвердження особи співрозмовника, шляхом порівняння цих самих ключів. За рахунок застосування наскрізного шифрування інші користувачі і ті, хто мають доступ до серверів не повинні мати доступ до даних користувачів і їх чатів. Все це в теорії повинно забезпечувати високий рівень безпеки переданих особистих даних.

2. Аналіз технологій для проектування android додатку

2.1. Постановка задачі мобільного додатку

Не так давно виявилось, що величезна компанія Facebook має доступ до повідомлень користувачів, надіслані через їх додаток Messenger. Компанія заявила, що аналізує повідомлення лише для тренування інших систем. Проте схожа ситуація була і з Google, який раніше читав особисті електронні листи для кращої персоналізації реклами. Один популярний месенджер Telegram, про нього детальніше йшлось в розділі 1.3.2, піддається тиску з сторони влади різних країн, які б хотіли знати, про що спілкуються користувачі.

Здається, що в наш час майже неможливо використовувати будь-який спосіб зручного спілкування з повною приватністю.

Сказати, що певні месенджери забезпечують справжню "конфіденційність" означає, що користувачі повинні довіряти компанії, яка розробила цю систему. Але практично завжди немає вагомих причин для цього. Більшість сучасних засобів комунікації є "закритим джерелом", тобто вихідний код цих продуктів ніколи не публікується і незалежні дослідники не можуть його перевірити. Це означає, що навіть якщо команда, що стоїть за такими месенджерами, не має наміру оприлюднювати особисту інформацію користувачів, немає гарантії, що хакери чи інші зловмисники не будуть використовувати помилки програмістів або недоліки безпеки для отримання доступу до таких даних.

Ще одне питання випливає з підходів до розробки ПЗ. В даний час більшість месенджерів використовують застарілі технології, такі як модель P2P, яка не може приховувати IP-адреси людей.

Але найбільша проблема полягає в тому, що "безпека та конфіденційність" використовуються виключно для маркетингових цілей, тоді як насправді корпорації, які стоять за популярними месенджерами, взагалі не бажають конфіденційності для своїх користувачів. Ось чому вам

потрібно використовувати свій номер телефону або обліковий запис соціальних мереж для входу, а не дозволяти додатку доступу до вашого списку контактів, щоб почати спілкуватися. Усі ці дані потім будуть передані на віддалені сервери, і в масах мало хто знає, як вони будуть використовуватися.

Навіть Telegram, який позиціонує себе як найбезпечніший месенджер у світі, абсолютно неанонімний. Він централізований і може бути заблокований, що вже траплялося зі службою в Ірані та Росії. Як результат, його користувачів можуть змусити використовувати інші популярні та ще менш анонімні засоби зв'язку.

Безкоштовно та безпечно не завжди означає приватно. Навіть дійсно безпечні месенджери, які використовують потужні технології шифрування, все ще можуть бути неанонімними, збираючи та обмінюючись особистою інформацією користувачів. Сьогоднішня структура ринку дозволяє послугам обміну повідомленнями заробляти гроші, продаючи дані користувачів або показуючи рекламу.

2.1.1. Ціль і призначення мобільного додатку

Мета цього проекту - представити сучасну систему для чатів, яка має на меті забезпечити простий у використанні та ефективний функціонал разом із корисними послугами. Система має на меті бути достатнім середовищем для соціальних цілей та розваг.

Основні цілі можна підсумувати наступними пунктами:

- Створення легкого мобільного додатку зі стильним зовнішнім виглядом.
- Створення надійної платформи обміну повідомленнями з можливістю надання необхідних функцій соціальної мережі. Особливості включають групи чату, завантаження зображень, налаштування облікового запису та можливість підтримувати приватну спільноту друзів.

- Заборонено переписування між користувачами, якщо вони не є друзями.

2.2. Технології для розробки android додатку

Android - це мобільна операційна система яка вільно поширюється, базується на модифікованій версії ядра Linux та іншого ПЗ, в основному, для пристроїв з сенсорним екраном. Розробляється та поширюється компанією Google [14]. Android був створений Open Handset Alliance, який очолює Google. Стартував вихід ОС Android 1.0 23 вересня 2008 року і називався Astroboy. З кожним роком виходили нові оновлення операційної системи, які додавали новий функціонал. Кожна головна версія була названа в алфавітному порядку в честь різноманітних солодощів: «Кекс», «Пончик», «Еклер» та інші. На сьогоднішній час, останньою версією ОС є Android 10 «Q», що вийшов 3 вересня 2019 року.

Мобільний додаток – це певне ПЗ, що призначене для використання смартфонами, планшетами, навіть деякими розумними годинниками, та іншими мобільними пристроями. Вони можуть бути завантажені через браузер або офіційні онлайн-магазини – Play Store, App Store та інші, деякі найпопулярніші додатки вже встановлені на пристрої по замовчуванню.

Програми, для мобільних пристроїв, розробляються за допомогою програм для створення ПЗ на Android (SDK) і, таких мов програмування, як Kotlin або Java. Java може легко поєднуватися з C/C++ , разом з вибором режимів їх сумісності, які дозволяють покращити підтримку C++. В 2017 році мова Kotlin була оголошена, як аналог Java. Мова Go також підтримується, хоча з обмеженим набором інтерфейсів прикладного програмування (API).

2.2.1. Вибір середовища для розробки додатку

Для розробки мобільних додатків Google дає можливість вибору багатьох інструментів для розробки Android SDK, який постачається з необхідними компонентами та емуляторами. Доступно кілька графічних середовищ для розробки - Eclipse, NetBeans і IntelliJ IDEA, та все ж Android Studio є більш перспективним. Він почав свою історію з 2014 року.

Перша версія Android Studio 1.0 була представлена в грудні 2014 року. На сьогоднішній день остання стабільна версія програми – AS 3.6 (12 лютого 2020 року).

У поточній стабільній версії надаються такі функції:

- Підтримка системи автоматичного збирання додатків Gradle.
- Виправлення кількох однакових частин коду програми одночасно.
- Інструменти для збільшення продуктивності, ефективності використання сторонніх компонентів та сумісності різних версій плагінів і бібліотек.
- Інтеграція ProGuard та можливості цифрового підпису програм.
- Допомога створення дизайну та компонентів на основі вбудованих шаблонів Android.
- Потужний редактор XML - макетів, що надає змогу розробникам графічно редагувати елементи інтерфейсу вікон додатку.
- Можливість створення програм для ОС Android Wear.
- Тісна інтеграція з Google Cloud Platform, яка робить можливим імплементацію з Firebase Cloud Messaging (раніше "Google Cloud Messaging") і Google App Engine.
- Вибір таких налаштувань віртуального пристрою Android (емулятор), як версія ОС від 4.2 до останньої, практично будь-які діагоналі, розширення і співвідношення сторін екрану, що дозволяє максимально гнучко налаштовувати інтерфейс для смартфонів користувачів.

Сторонні бібліотеки для розширення функціоналу додаються в проект за допомогою директиви пакету Gradle. Велика кількість сторонніх інструментів розробки SDK для Android розповсюджується за допомогою менеджера пакунків під назвою maven.

2.2.2. Вибір мови програмування

Java була типовою мовою написання програм для Android з моменту запровадження платформи Android у 2008 році. Java - це об'єктно-орієнтована мова програмування, яка була розроблена компанією Sun Microsystems у 1995 році (зараз вона належить Oracle). Це популярна мова, позиціонувалась, як чиста об'єктно-орієнтована мова (порівняно з C++) і швидко була прийнята розробниками платформи Android. Java спочатку компілюється в "байт-код", який інтерпретується під час виконання базовою віртуальною машиною Java, що працює в ОС.

У 2017 році в Google оголосили, що з їх боку буде відбуватись підтримка Kotlin як альтернативної мови першого класу для програмування Android. Kotlin взаємодіє з Java, і всі бібліотеки Java можна викликати з Kotlin. Котрово кажучи, Котлін - охайніша форма Java. На рівні виконання, Kotlin компілюється в Java Bytecode. І все-таки, станом на 2020 рік, Котлін є кращою мовою для розробки Android, відповідно до Google.

Не зважаючи на зручніший і кращий аналог, в проекті використовувалась мова програмування Java, у зв’язку з набагато складнішою інтерпретацією певного коду. Для мови Kotlin на даний момент немає доступної підтримки аналогів реалізації наскрізного шифрування. В майбутньому, якщо така реалізація існуватиме, буде вагома причина переробити додаток мовою Kotlin.

3. Розробка програмного продукту

3.1. Реалізація програмної частини додатку

Розробка додатку відбувалась мовою Java в графічному середовищі Android Studio 3.6. Допоміжні бібліотеки та плагіни, що використовувались в процесі розробки:

Android Maven Plugin - використовується для створення додатків для операційної системи Android, а також для побудови власних бібліотек.

- Jacoco – плагін для тестування коду.
- Picasso - потужна бібліотека для завантаження та кешування зображень.
- OkHttp – бібліотека для роботи з HTTP протоколами. Ефективний клієнт для роботи з HTTP.
- LovelyDialog – бібліотека з набором приємних на вигляд шаблонів діалогів програми з користувачем.

Основні частини коду програми можна побачити в додатку.

3.1.1. Опис архітектури

Використана архітектура призначена для додатку месенджера і базується на типі клієнт-сервер. На стороні клієнта, коли користувач вперше запускає програму, то він або вибирає вхід, або реєстрацію. На стороні бекенду база даних чату складається з 3 основних вузлів: друзі, повідомлення, користувачі (рис. 3.1).

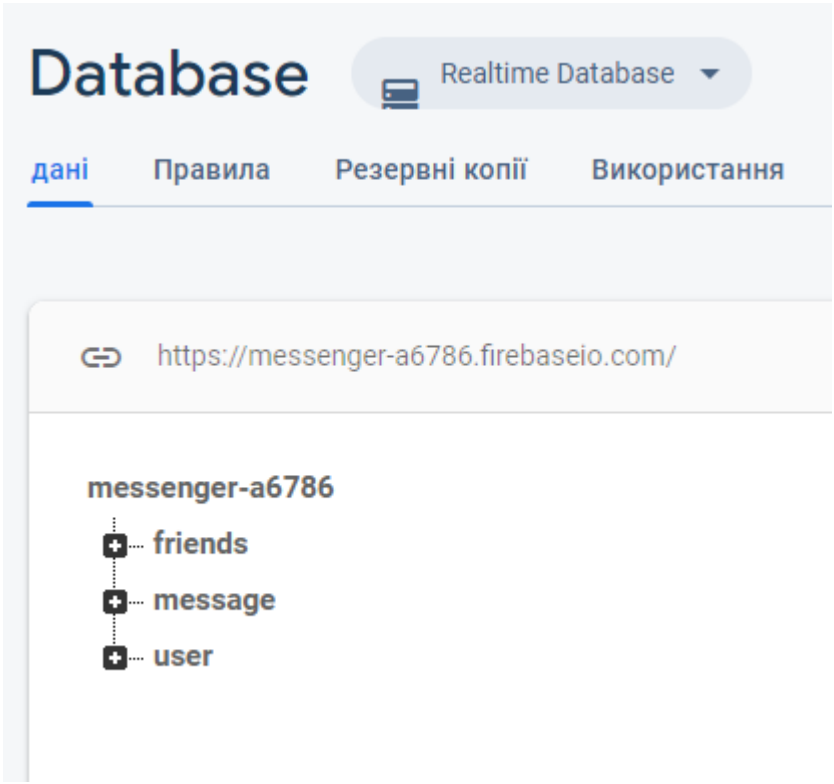


Рис.3.1. Вигляд бази даних додатку.

Сервер користувача керує обліковими даними користувачів (рис.3.2).

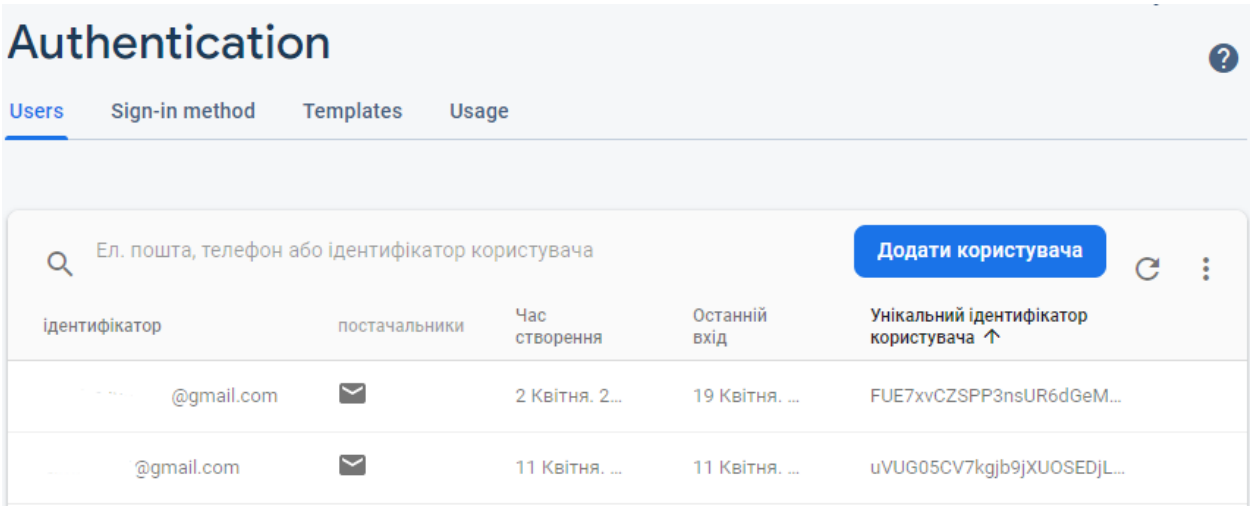


Рис.3.2. Користувачі.

Сервер повідомлень обробляє повідомлення між користувачами за допомогою Firebase Cloud Messaging (FCM). Якщо користувач надсилає

повідомлення в автономному режимі, то вони будуть тимчасово зберігатися у черзі FCM протягом певного періоду часу, і коли він підключиться до мережі, ці повідомлення відправляться і будуть видалені з черги. Схема архітектури показана на рис.3.3.

Рис.3.3. Схема архітектури додатку.

3.1.2. Реєстрація облікового запису

Під час першого запуску додатку автоматично налаштовується захищене місце для зберігання локальних ключів - Keystore, щоб ускладнити їх вилучення з пристрою сторонніми особами чи іншими програмами [15].

Електронна пошта та ім’я користувача є унікальними. Ім'я, електронна адреса та пароль необхідні для реєстрації нового акаунта. Після введення необхідної інформації – відбувається перевірка існування пошти. Мінімальна кількість символів паролю повинна бути не менше шести - пароль шифрується за допомогою алгоритму XSalsa20 [16], після чого облікові дані користувачів надсилаються на сервер. Після перевірки сервер генерує унікальний ідентифікатор (токен), який виступає як ідентифікатор

користувача. Після цього надходить підтвердження про успішну реєстрацію в клієнтській програмі, а інформація про клієнта зберігається у хмарній базі даних.

Додаток генерує набір ключів:

a) Ключ для шифрування пароля.
b) Пара відкритих ключів для обчислення ключа сеансу.
c) Симетричний ключ для шифрування / дешифрування повідомлень.

3.1.3. Вхід за допомогою облікового запису

Для автентифікації користувача необхідні електронна пошта та пароль. Після введення інформації про автентифікацію пароль шифрується, після чого облікові дані користувачів надсилаються на сервер. Сервер перевіряє, чи є електронна пошта та пароль дійсними. Після перевірки, веб-маркер JSON (JWT) [17] створюється та відправляє клієнту для його зберігання. Коли клієнт подає запит пізніше, JWT передається із запитом. Сервер перевіряє JWT, якщо він дійсний, запит обробляється (рис.3.4).

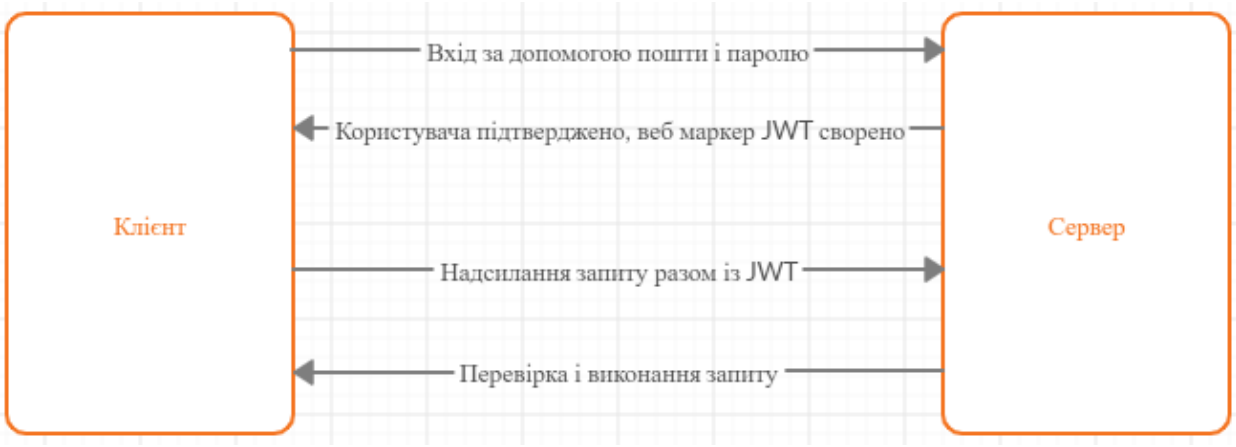


Рис.3.4. Підтвердження даних користувача для входу.

3.1.4. Система для обміну повідомленнями Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) - це послуга, яка полегшує обмін повідомленнями між мобільними та серверними частинами додатку. Він

заснований на сервісах Google Play, які підтримують крос-платформу (iOS, Android та Web). Це безкоштовний сервіс, який дозволяє надсилати легкі повідомлення з сервера на пристрої [18]. Це ефективно економить ресурс акумулятора користувача, уникаючи запитів на сервер про нові повідомлення з певною періодичністю. Він підтримує протокол TLS для забезпечення безпеки даних, що проходять через інформаційний канал.

При початковому запуску програма отримує наступне:

1. Додаток підключається до сервера FCM і реєструє себе.
2. Після успішної реєстрації FCM надає токен реєстрації пристрою. Цей токен реєстрації унікально ідентифікує кожен пристрій.
3. Додаток надсилає токен реєстрації на сервер, щоб зберегти його в базі даних MongoDB.

Наведені вище етапи показані на рис. 3.5.

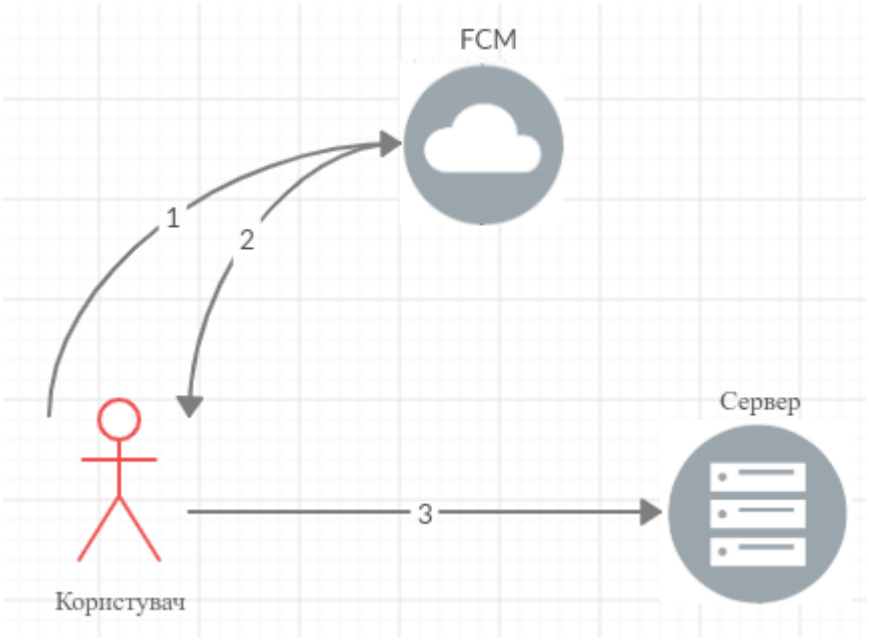


Рис.3.5. Схема зв’язків додатку, серверу і FCM.

Коли сервер надсилає push-сповіщення, він надсилає запит FCM, який надсилає push-повідомлення, а також токен реєстрації. FCM ідентифікує

цільовий пристрій за допомогою токена реєстрації, після чого починає надсилати дані.

3.1.5. Налаштування ключа сеансу

Додати користувачів до списку контактів можна у списку зареєстрованих користувачів, де показані їх імена, зображення профілю та короткий опис.

Наприклад, перший користувач називається Аліса, а другий - Боб. Коли відправляється запит на додавання в друзі, ім'я Боба вводиться Алісою, її відкритий ключ витягається із сховища, після чого запит надсилається на сервер.

Коли запит отримано, він виглядає як сповіщення (рис. 3.6). Якщо Боб прийняв запит про дружбу, його приватний ключ вилучається відкритим ключем Аліси для обчислення ключа сеансу за допомогою Еліптичної кривої Діффі-Гелмана (ECDH) над кривою Curve25519 [19] і хешування результату за допомогою алгоритму HSalsa20 [16], тоді ключ сесії зберігається в хмарному сховищі (рис. 3.7). Врешті-решт, підтверджений запит надсилається з його відкритим ключем на сервер для доставки Алісі.

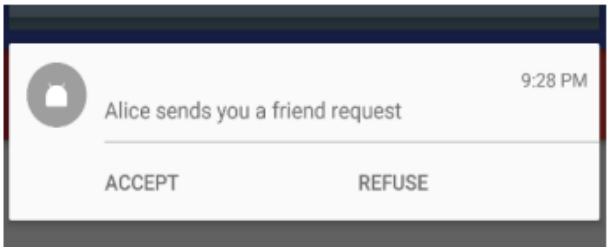


Рис.3.6. Отримано запит на додавання в друзі.

Після отримання повідомлення про підтвердження запиту відбуваються ті ж самі вищеописані кроки. Ключ сеансу обчислюється за допомогою приватного ключа Аліси та відкритого ключа Боба, після чого він зберігається у хмарному сховищі для подальшого використання.

Ключ сеансу однаковий для обох сторін, це сильна сторона Елліптичної кривої Діффі-Хеллмана (ECDH), і тому легко уникнути так званої атаки через людину посередині. Принцип цієї атаки в перехопленні даних в проміжку їх передачі з одного пристрою на сервер, чи інший пристрій.

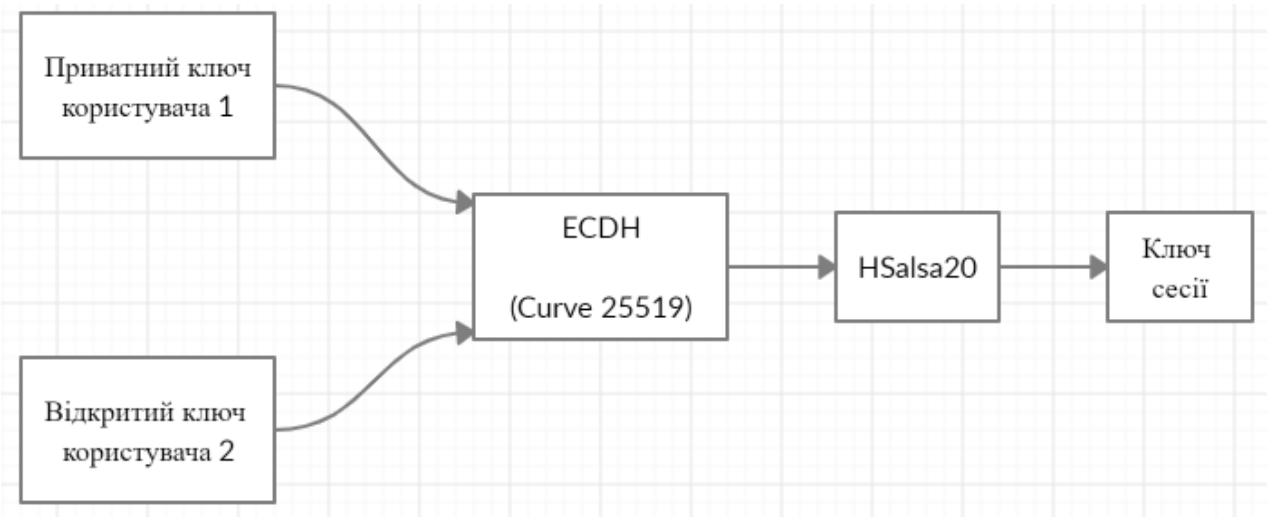


Рис.3.7. Принцип додавання користувачів в друзі.

3.1.6 Обмін повідомленнями, реалізація шифрування

Додаток використовує наскрізне шифрування, щоб забезпечити більшу безпеку та захистити конфіденційність користувачів. Повідомлення, які зберігаються в базі даних, зберігаються в зашифрованому вигляді. Процеси шифрування та дешифрування відбуваються на стороні клієнта у самій програмі перед завантаженням та відправленням Firebase. Наприклад, повідомлення «Гарного дня» в базі даних виглядає як на рис.3.8.

Рис.3.8. Вигляд повідомлення у базі даних.

Повідомлення, надіслане в базу даних, зберігається в зашифрованому вигляді, а всі шифрування та дешифрування виконуються на стороні клієнта в самій програмі перед відправленням до Firebase.

Кожного разу, коли повідомлення готується до відправлення, генерується секретний ключ з початкового 32-байтного ключа за допомогою алгоритму AES в режимі ECB з PKCS5, доповнюється результат зашифрованого байтного повідомлення, а потім кодується у вигляді 64-символьного рядка і нарешті відправляється до бази даних а для дешифрування використовується зворотна операція, виконана декодуванням останнього кроку у формат рядка UTF8.

Шифрування проводиться за допомогою алгоритму AES в режимі ECB. AES був прийнятий урядом США і зараз використовується у всьому світі. Він замінює стандарт шифрування даних (DES). Через велику різноманітність ключів та високу складність в обчислювальних процесах брутфорс атаки не загрожують безпеці AES. Хоча існують деякі теоретичні атаки проти AES, їх все ж неможливо здійснити. Незважаючи на те, що обчислювальна потужність комп'ютерів подвоюється кожні два роки, пройдуть десятиліття, перш ніж AES стане обчислювально незахищеною.

3.1.7 Реалізація серверної частини

Серверна сторона базується на Node JS [20] та нереляційній базі даних MongoDB [21]. Node JS швидкий, здатний обробляти велику кількість одночасних з'єднань з високою пропускною здатністю, що еквівалентно високій масштабованості. MongoDB та Node JS часто використовуються разом через використання ними файлів JSON, тому не потрібно витрачати час на перетворення даних між ними, що полегшує взаємодію між собою. Крім того, MongoDB підтримує протокол TLS для забезпечення безпеки даних, що проходять через інформаційний канал (рис. 3.9).

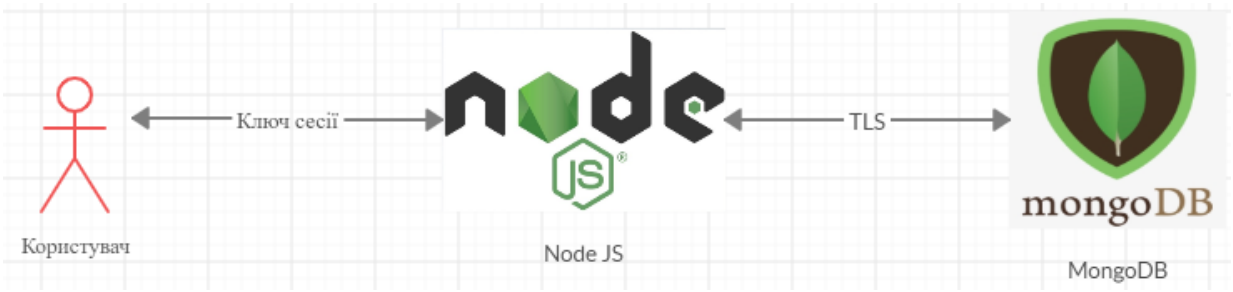


Рис.3.9. Серверна частина додатку.

Для виконання запиту клієнта проходить кілька кроків, такі як:

1. Спочатку необхідно запустити підключення до MongoDB, а потім запустити Node JS. На цьому етапі сервер готовий прийняти запити користувача.
2. Коли користувач надсилає запит, сервер отримує HTTP-запит у форматі JSON. Після цього запит виконується.
3. Запит HTTP порівнюється з базовим, якщо він збігається, тоді передається далі Express Framework.
4. Express Framework отримує HTTP-запит і спрямовує його до конкретної кінцевої точки, яка відповідає запиту. У разі невідповідності жодному з маршрутів відобразиться помилка. В іншому випадку він буде переданий на контролер, який виконує необхідну операцію.
5. Відбувається надсилання запиту до бази даних MongoDB за допомогою сервісу mongoose для обробки операцій.
6. Коли дані отримуються з бази даних MongoDB і виконуються необхідні операції, Node JS отримує відповідь, після чого надсилає її клієнту.

Наведені вище етапи показані на рис. 3.10.

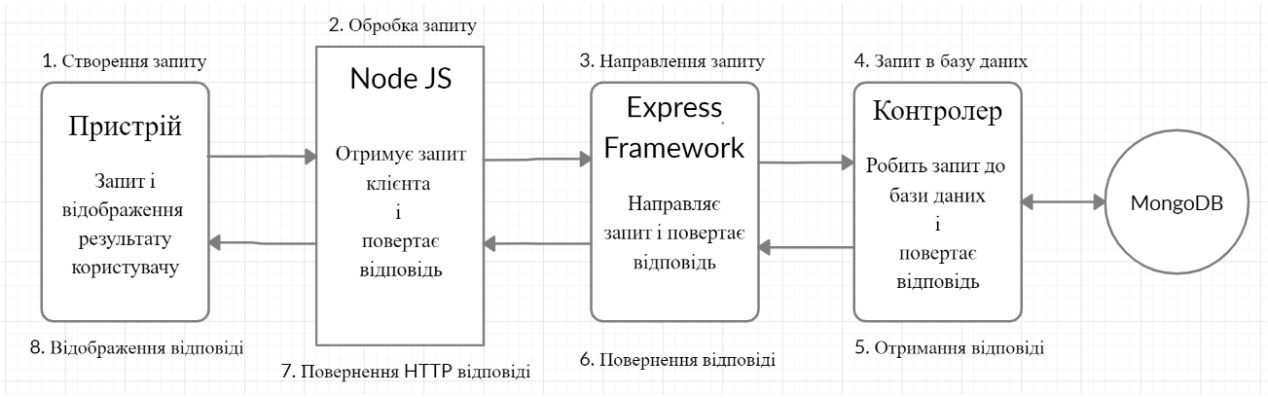


Рис. 3.10. Етапи запиту клієнта до бази даних.

У хмарній базі даних є 3 основних вузли (рис.3.11, 3.12, 3.13):

1. **Користувач**

   Містить інформацію про користувача, таку як:

   • електронна пошта

   • ім’я користувача

   • короткий опис

   • останнє повідомлення

   • фото (зберігається закодовано у 64 базовому байтовому масиві)

   • маркер (Це рядок, що надається користувачеві, коли він входить у систему та використовується у випадку повідомлення)

2. **Повідомлення**

   • ідентифікатор відправника

   • ідентифікатор одержувача

   • ідентифікатор чату

   • текст

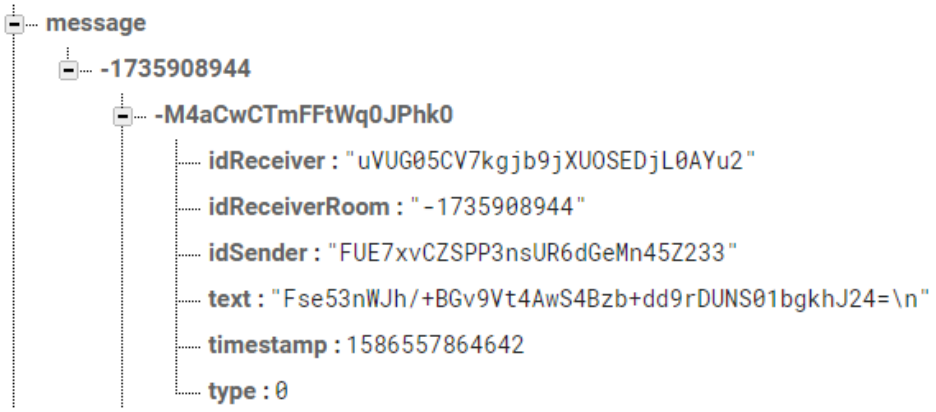
   • часова відмітка

3. **Друзі**

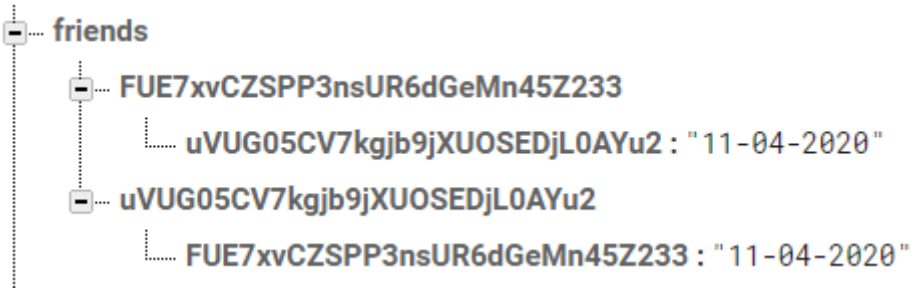
Статус дружби зберігається двічі, що означає для будь-яких двох друзів Боба і Аліси зберігається так: якщо Боб додав друга Алісу і Аліса додала друга Боба, то вони вважатимуться друзями і зможуть відправляти один

одному повідомлення, а також зберігається дата, коли вони стають друзями.

3.11. Користувачі



3.12. Повідомлення



3.13. Друзі

3.2. Реалізація клієнтської частини додатку

Наступна діаграма(рис.3.14) показує графічний інтерфейс додатку і способи переміщення між вікнами. На діаграмі також можна побачити як

можна переміщатись між вікнами додатку і як вони взаємодіють між собою та базою даних.

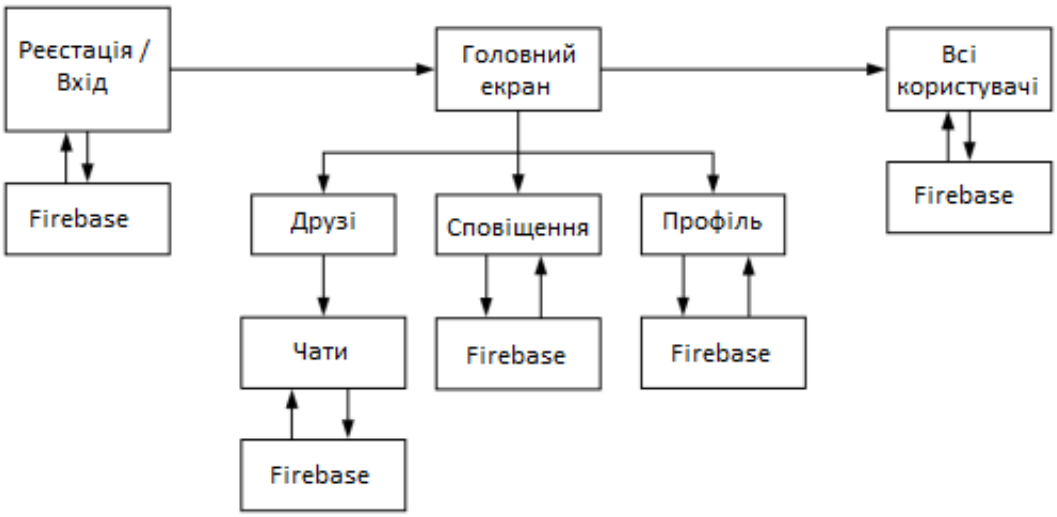


Рис.3.14. Схема вікон програми, які можуть побачити користувачі.

3.2.1. Реєстрація

При першому запуску додатку перед користувачем відкриється екран з полями для вводу існуючих даних. Так як користувач ще не має облікового запису, йому потрібно відкрити екран для реєстрації(рис.3.15). Це надзвичайно легкий і зручний процес. Щоб створити новий обліковий запис, користувач повинен ввести наступні дані:

1. Ім’я – ім’я яке будуть бачити інші користувачі. Може бути який-небудь набір символів.
2. Електронна пошта – при вводі відбувається перевірка на правильність стандартного вигляду символів електронної пошти. Вона повинна бути унікальною для кожного користувача у системі. Після натискання кнопки реєстрація відбувається перевірка, чи взагалі існує така адреса.
3. Пароль – повинна бути прихованою комбінацією букв і чисел. Приймаються паролі не менше 8 символів.
4. Перевірка паролю – необхідно повторити введення паролю для перевірки на правильність.

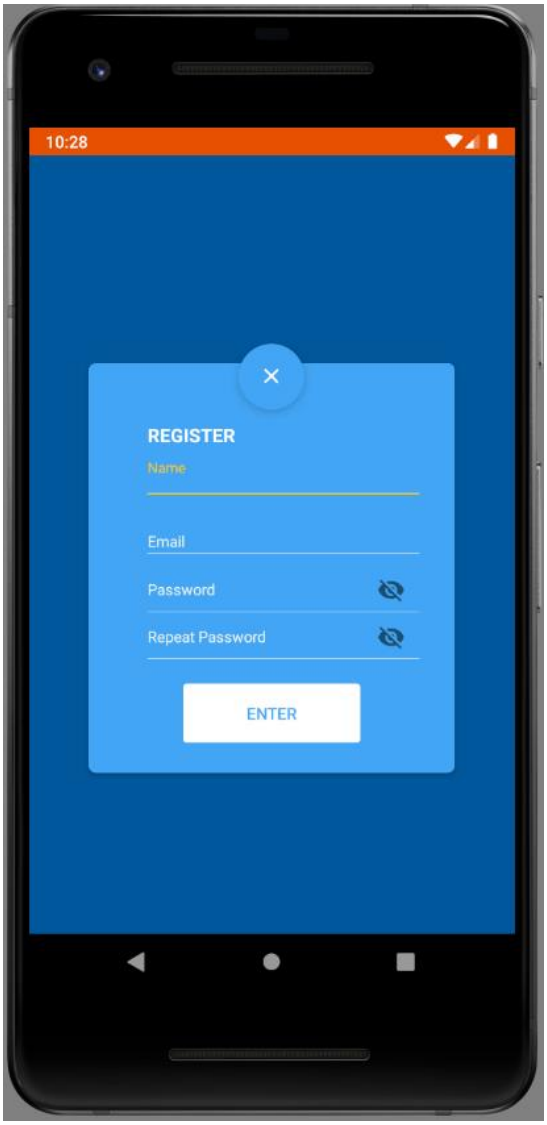


Рис.3.15. Екран реєстрації користувача.

3.2.2. Вхід

Це початковий екран за замовчуванням, який з’являється користувачеві після запуску додатку. Зазвичай користувач повинен ввести свою електронну адресу та пароль, які він вказав під час створення облікового запису, а потім натиснути кнопку ENTER, показану на рисунку 3.16.

Цей екран переміщає користувача на сторінку його облікового запису після натискання кнопки та після підтвердження облікових даних.

Користувач, можливо, забув свій пароль або вказав неправильний пароль під час процесу реєстрації, у цьому випадку він може натиснути параметр Забули пароль. Наступний пункт детальніше описує цей процес.

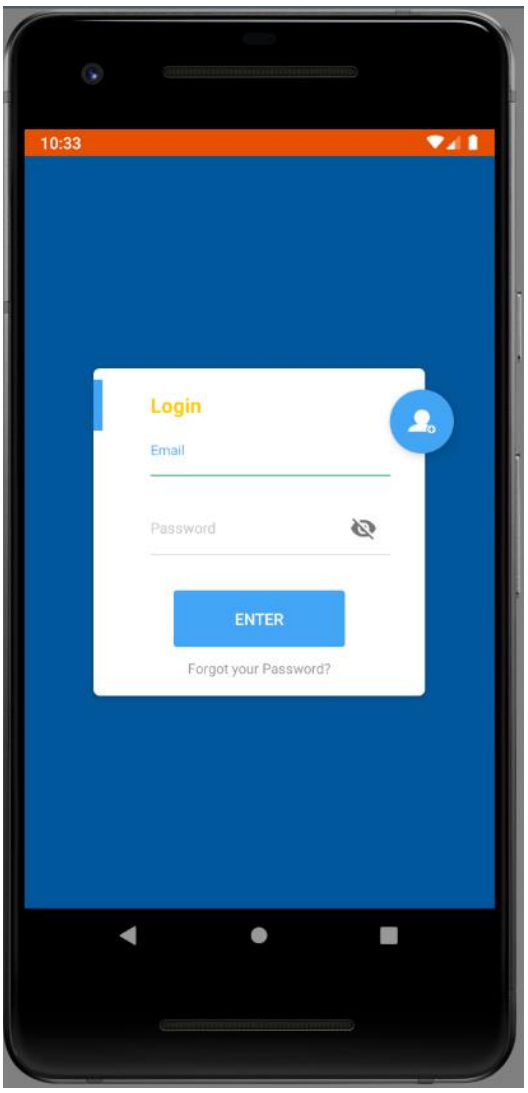


Рис.3.16. Екран входу.

3.2.3. Відновлення паролю

Користувач, який з певних причин не пам’ятає пароль, може перейти на цей екран. Він повинен просто ввести адресу електронної пошти від свого облікового запису та натиснути кнопку RESET PASSWORD, як показано на рисунку 3.17.

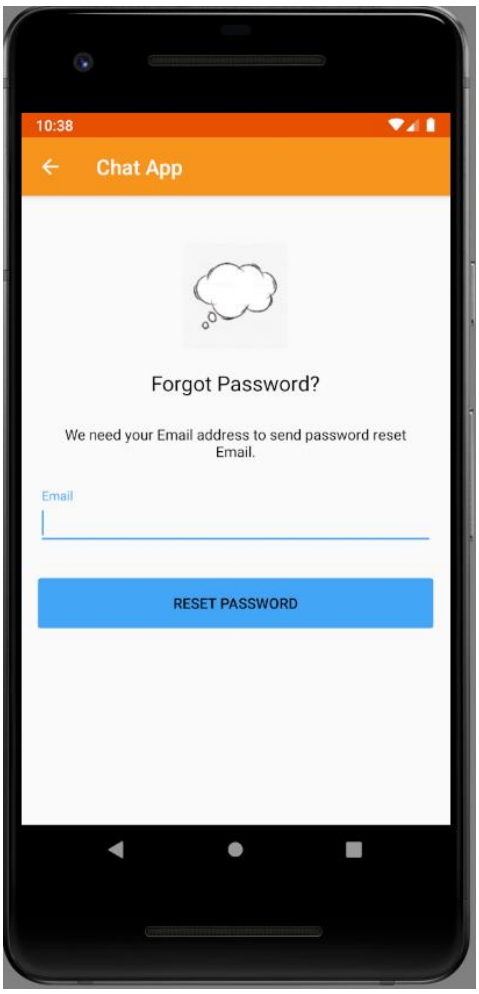


Рис.3.17. Екран відновлення паролю.

Після цього користувачу повинен прийти лист для відновлення паролю, як показано на рис.3.18.

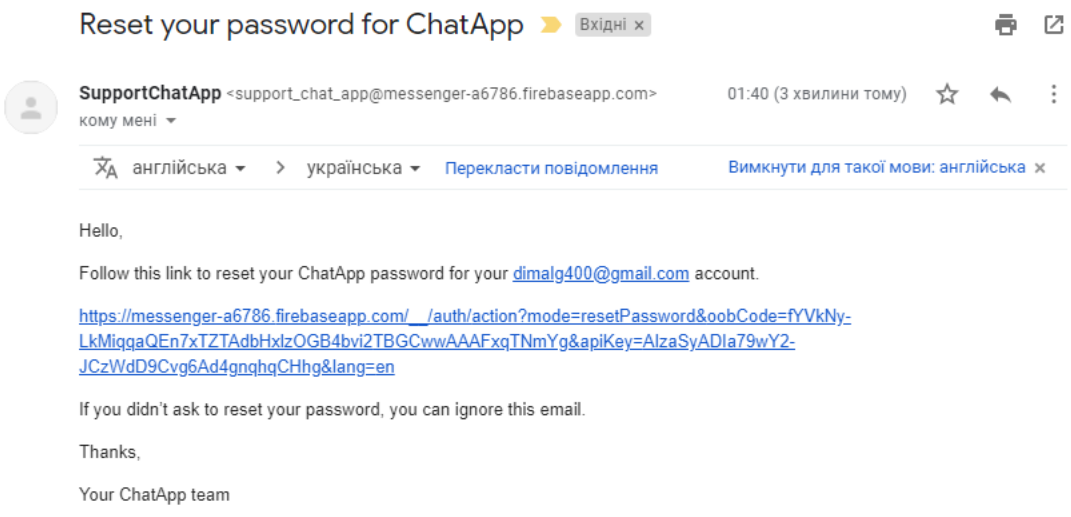


Рис.3.18. Лист з посиланням для відновлення паролю аккаунту.

В електронному листі додається посилання, по якому слід перейти, щоб скинути пароль. Якщо користувач отримує аналогічний електронний лист не відновлюючи свій пароль, це означає, що інший користувач, що, напевно, забув свій пароль аналогічним чином, і вказав помилкову адресу електронної пошти. У цьому випадку краще ігнорувати цей електронний лист.

3.2.4. Чати

Після входу в обліковий запис користувача додаток відображає екран контактів та чатів. На рис. 3.19 показаний такий екран. На цьому екрані відображаються друзі користувача та чати з цими друзями, ви можете переглянути розмову, натиснувши на неї.

Користувач може додати нових друзів, натиснувши кнопку додати друзів у нижній правій частині екрана.

На рис. 6.20 показані інші параметри, доступні для цього екрану:

- Знайти користувачів – натисніть цю опцію, щоб переглянути інших користувачів і отримати можливість додавати їх як нових друзів. Наступний пункт детальніше описує цей варіант.
- Вихід – Цей параметр повертає користувача до екрана входу, де він може використовувати різні облікові записи для входу або захист особистої інформації, яка може зберігатися у їхньому акаунті.

У верхній частині екрана можна знайти панель навігації з трьома доступними варіантами, що ведуть до двох основних екранів: екрани чатів та профілю.

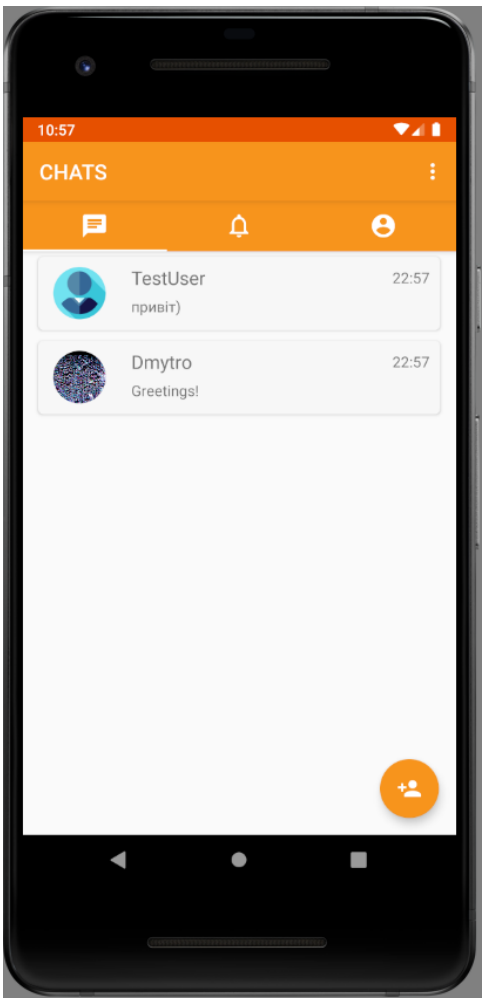


Рис. 3.19. Екран чатів.

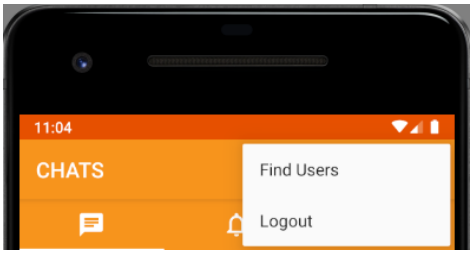


Рис.3.20. Екран чатів з опціями.

3.2.5. Знайти друзів

На рис. 3.21 показано екран «Знайти друзів». На цьому екрані користувачі можуть шукати інших користувачів у системі та додавати їх як друзів, якщо хочуть. Для кожного користувача відображаються ім’я користувача та стан для полегшення їх ідентифікації під час процесу пошуку. Користувач також може скасувати додавання друга, натиснувши кнопку «Видалити друга».

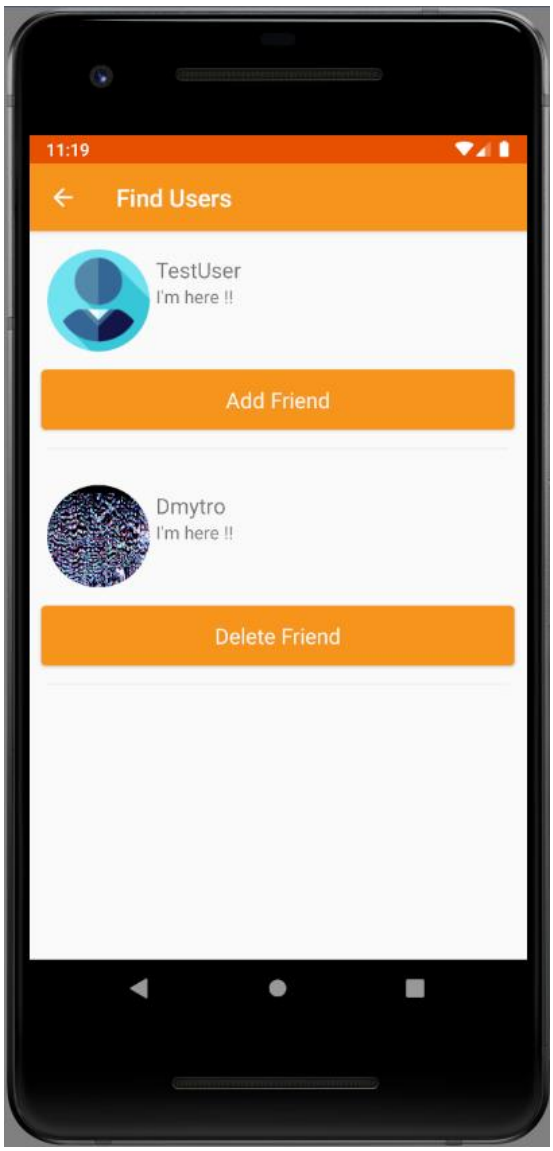


Рис.3.21. Екран «Знайти друзів»

3.2.6. Сповіщення

Для того, щоб можна було почати спілкуватись з іншим користувачем, його необхідно зробити своїм другом. Знайти іншого користувача можна в пункті «Знайти користувачів», описаним вище, і натиснути кнопку «Додати друга». Цьому користувачу прийде відповідне сповіщення, як на рис. 3.22 і після його підтвердження новий чат з’явиться в меню чатів.

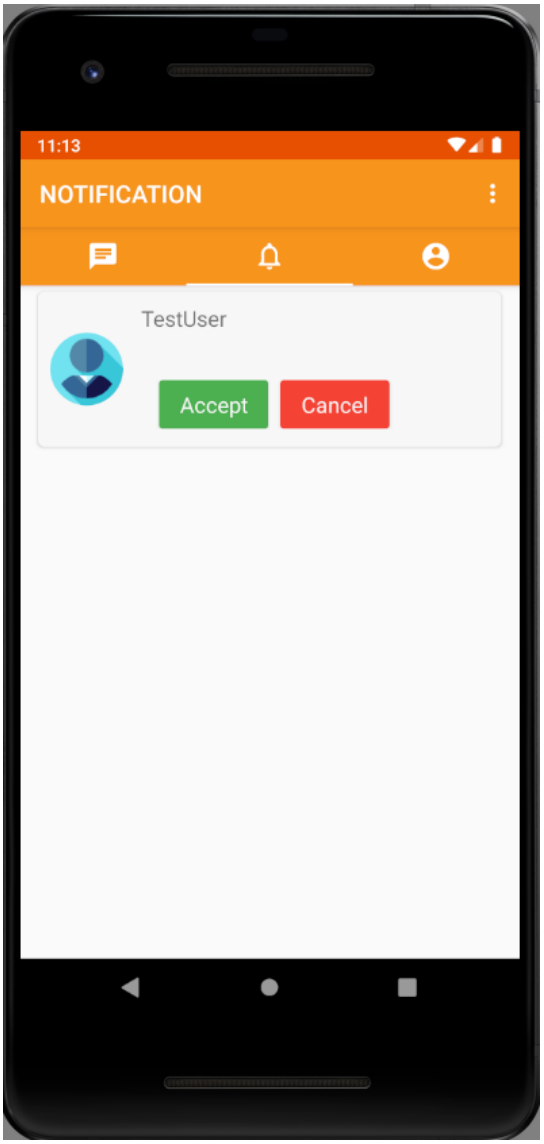


Рис.3.22. Сповіщення від іншого користувача.

3.2.7. Профіль

Екран профілю підсумовує інформацію облікового запису для конкретного користувача, як показано на рисунку 3.23. У кожного користувача є унікальний профіль, який доступний для перегляду і редагування лише їм:

- Ім'я користувача - ім'я, вибране користувачем під час реєстрації, і це ім'я, яке можуть бачити інші користувачі.
- Статус - короткий опис користувача про себе. Цей текст відображається для інших користувачів, щоб допомогти визначити користувача.

- Електронна пошта - це електронна адреса, яка використовується для створення облікового запису та пов’язана з нею.

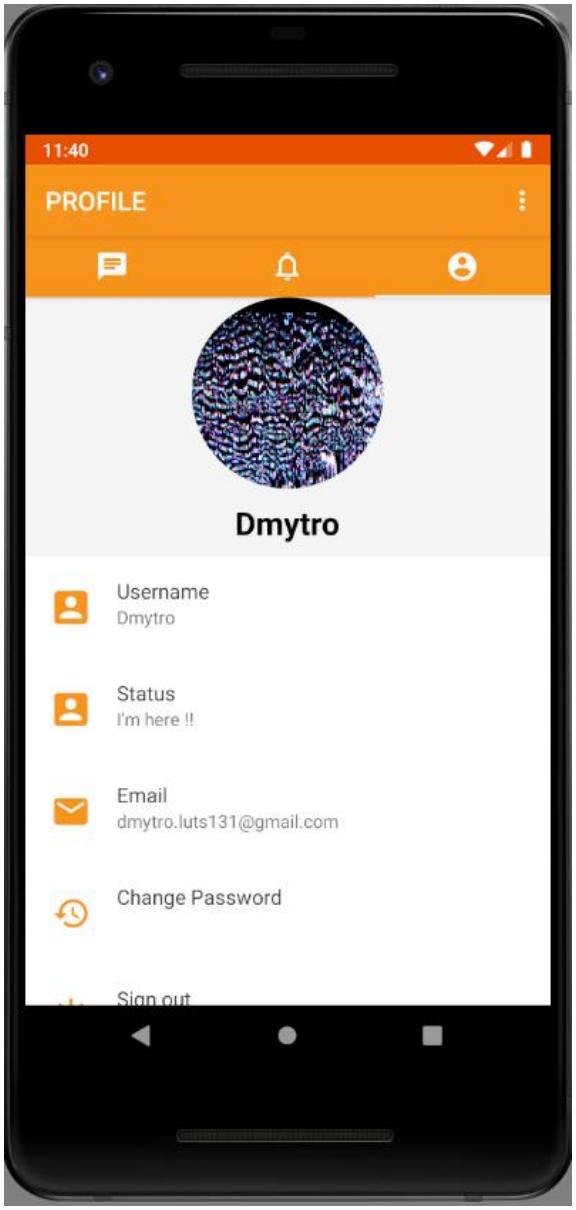


Рис.3.23. Екран профілю.

3.2.8 Чат

Користувач може вибрати один з чатів як на рис.3.24, що відображаються на екрані контактів, щоб почати спілкуватися в чаті та надсилати повідомлення. Усі чати по замовчуванню є захищеними і ніхто,

окрім самих користувачів не має доступу до вмісту повідомлень. Користувачі можуть вставляти емоджі у свої текстові повідомлення.

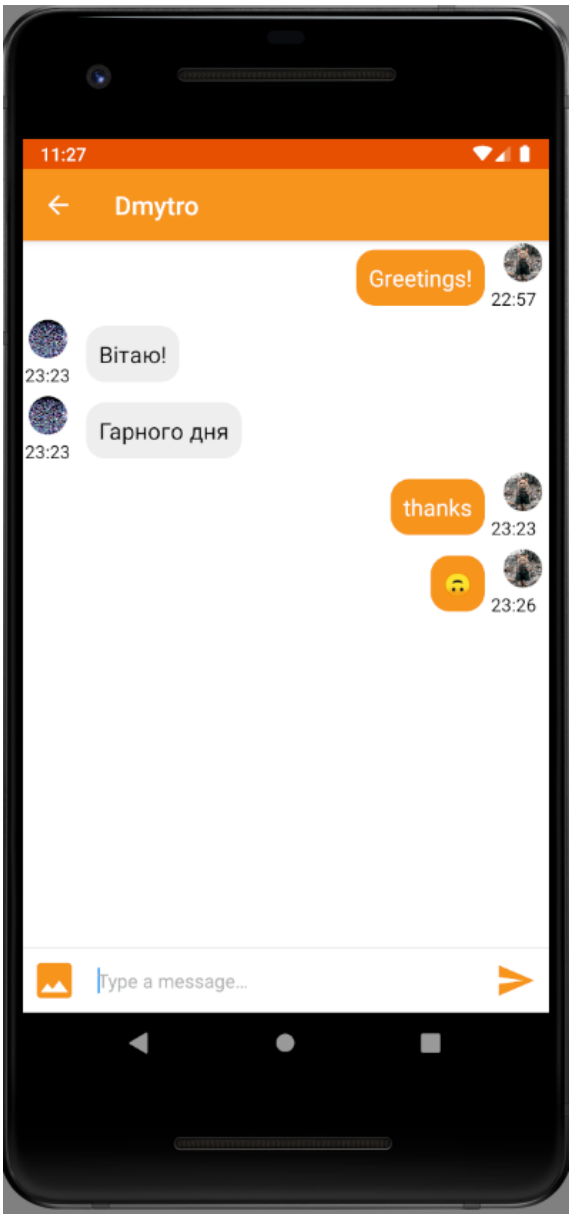


Рис.3.24. Екран чату.

Вихідний код додатку можна побачити на сторінці мого GitHub аккаунту [22].

ВИСНОВКИ

В ході виконання дипломної роботи було спроектовано та розроблено мобільний додаток для ОС Android, для захищеного спілкування між людьми в мережі Інтернет. Програмний продукт складається з клієнтської та серверної частин. Клієнтська частина має зручний інтерфейс, де можна знайти цікавого співрозмовника. В основі впроваджено стоїть спосіб безпечного спілкування між людьми, щоб ніхто інший не зміг отримати вміст повідомлення. Серверна частина додатку складається з хмарної бази даних, в якій знаходяться дані про користувачів та їх повідомлення в зашифрованому вигляді.

В результаті було досягнуто наступне:

- Досліджено ступені захищеності інших месенджерів та їх безпеку відносно користувацьких даних.
- Проаналізовано стан та можливості програмного забезпечення для створення мобільних додатків. Реалізований додаток підтримує ОС Android не нижче версії 4.4 KitKat. Середовищем для розробки було вибрано Android Studio та мова програмування Java.
- Реалізовано сучасний та ефективний функціонал. Для наскрізного шифрування вибрано алгоритм AES, який забезпечує максимально ефективну безпеку даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Statista - Number of smartphone users worldwide [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
2. Electronic Frontier Foundation (EFF) [Електронний ресурс]. – Режим доступу: https://www.eff.org/pages/secure-messaging-scorecard
3. Comparison of antivirus software [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Comparison_of_antivirus_software
4. The Problem with Mobile Phones [Електронний ресурс]. – Режим доступу: https://ssd.eff.org/en/module/problem-mobile-phones
5. Betrayed by our own data [Електронний ресурс]. – Режим доступу: https://www.zeit.de/digital/datenschutz/2011-03/data-protection-malte-spitz
6. Open Whisper Systems // GitHub. [Електронний ресурс]. – Режим доступу: https://github.com/signalapp/libsignal-service-java/wiki/Using-the-Open-Whisper-Systems-server
7. Signal Protocol documentation [Електронний ресурс]. – Режим доступу: https://signal.org/docs/specifications/xeddsa/
8. Signal android app [Електронний ресурс]. – Режим доступу: https://github.com/signalapp/Signal-Android
9. MTProto documentation [Електронний ресурс]. – Режим доступу: https://core.telegram.org/api/end-to-end
10. Secret chat Telegram [Електронний ресурс]. – Режим доступу: https://core.telegram.org/api/end-to-end/pfs
11. Telegram messenger for Android [Електронний ресурс]. – Режим доступу: https://github.com/DrKLO/Telegram
12. Viber Statistics [Електронний ресурс]. – Режим доступу: https://99firms.com/blog/viber-statistics/
13. WhatsApp encryption [Електронний ресурс]. – Режим доступу: https://scontent.whatsapp.net/v/t61.22868-

34/68135620_760356657751682_6212997528851833559_n.pdf/WhatsApp-Security-Whitepaper.pdf?_nc_sid=41cc27&_nc_ohc=1IH8I-cEDRYAX_cTOC7&_nc_ht=scontent.whatsapp.net&oh=79ccb2eb79558b44c9b5b44bf17b260d&oe=5E982093

14. Android [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Android_(operating_system)
15. “Android Keystore System | Android Developers.” [Електронний ресурс]. – Режим доступу: https://developer.android.com/training/articles/keystore.htm l.
16. D. J. Bernstein, “Extending the Salsa20 nonce,” no. Mc 152, pp. 1–14, 2011.
17. M. B. Jones, “The Emerging JSON-Based Identity Protocol Suite,” 2011.
18. “Firebase Cloud Messaging | Firebase.” [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/cloud-messaging/.
19. D. J. Bernstein, “Curve25519 : new Diffie-Hellman speed records,” vol. 25519, 2006.
20. «Node.js.» [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/.
21. «NoSQL Databases Explained | MongoDB.» [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/nosql-explained.
22. Вихідний код розробленого додатку на GitHub [Електронний ресурс]. – Режим доступу: https://github.com/GreenApple131/Android_Messenger

ДОДАТОК А

Фрагмент коду, який виконується після натискання «Register»:

```
public void onBackPressed() {
        animateRevealClose();
    }

        public void clickRegister(View view) {
            String nameString = nameEditText.getText().toString();
            String emailString = emailEditText.getText().toString();
            String passwordString = passwordEditText.getText().toString();
            String repeatPasswordString =
repeatPasswordEditText.getText().toString();
            if(validate(emailString, passwordString, repeatPasswordString)){
                Intent data = new Intent();
                data.putExtra(StaticConfig.STR_EXTRA_NAME, nameString);
                data.putExtra(StaticConfig.STR_EXTRA_USERNAME, emailString);
                data.putExtra(StaticConfig.STR_EXTRA_PASSWORD,
passwordString);
                data.putExtra(StaticConfig.STR_EXTRA_ACTION, "register");
                setResult(RESULT_OK, data);
                finish();
            }else {
                Toast.makeText(this, "Invalid email or not match password",
Toast.LENGTH_SHORT).show();
            }
        }

    private boolean validate(String emailStr, String password, String
repeatPassword) {
        Matcher matcher = VALID_EMAIL_ADDRESS_REGEX .matcher(emailStr);
        return password.length() > 6 && repeatPassword.equals(password) &&
matcher.find();
    }
```

Створення нового користувача і збереження його в базі даних:

```
private void createUser(String emailString, String passwordString, final
String nameString) {


        progressDialog.setIcon(R.drawable.ic_add_friend)
                .setTitle("Registering...")
                .setTopColor(getResources().getColor(R.color.colorPrimary))
                .show();
        mAuth.createUserWithEmailAndPassword(emailString, passwordString)
                .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        Log.d(TAG, "performFirebaseRegistration:onComplete:"
+ task.isSuccessful());
                        progressDialog.dismiss();

                        if (!task.isSuccessful()) {
```

```
                            new LovelyInfoDialog(LoginActivity.this) {
                                @Override
                                public LovelyInfoDialog
setConfirmButtonText(String text) {

findView(com.yarolegovich.lovelydialog.R.id.ld_btn_confirm).setOnClickListene
r(new View.OnClickListener() {
                                        @Override
                                        public void onClick(View view) {
                                            dismiss();
                                        }
                                    });
                                    return super.setConfirmButtonText(text);
                                }
                            }.setTopColorRes(R.color.colorAccent)
                                    .setIcon(R.drawable.ic_add_friend)
                                    .setTitle("Register false")
                                    .setMessage("Email exist or weak
password!")
                                    .setConfirmButtonText("ok")
                                    .setCancelable(false)
                                    .show();
                        } else {
                            currentUserId =
FirebaseAuth.getInstance().getCurrentUser().getUid();
                            userToken =
FirebaseInstanceId.getInstance().getToken();
                            initNewUserInfo(task.getResult().getUser(),
nameString, userToken);
                            Toast.makeText(LoginActivity.this, "Register and
Login success", Toast.LENGTH_SHORT).show();
                            Intent intent = new Intent(LoginActivity.this,
TabsActivity.class);
                            intent.putExtra("selected_index", "0");
                            startActivity(intent);
                            LoginActivity.this.finish();
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                progressDialog.dismiss();
            }
        });
    }


private void saveUserInfo() {

userReference.child(StaticConfig.UID).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                progressDialog.dismiss();
                HashMap hashUser = (HashMap) dataSnapshot.getValue();
                User userInfo = new User();
                userInfo.setName((String) hashUser.get("name"));
```

```
                userInfo.setEmail((String) hashUser.get("email"));
                userInfo.setAvatar((String) hashUser.get("avatar"));
                userInfo.setBioText((String) hashUser.get("bioText"));
                userInfo.setToken((String) hashUser.get("token"));

SharedPreferenceHelper.getInstance(LoginActivity.this).saveUserInfo(userInfo)
;
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
```

Фрагмент коду, який виконується при вході:

```
private void signIn(String emailString, String passwordString) {

        progressDialog.setIcon(R.drawable.ic_person_low)
                .setTitle("Login...")
                .setTopColor(getResources().getColor(R.color.colorPrimary))
                .show();

        mAuth.signInWithEmailAndPassword(emailString, passwordString)
                .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        Log.d(TAG, "performFirebaseLogin:onComplete:" +
task.isSuccessful());
                        progressDialog.dismiss();
                        if (!task.isSuccessful()) {
                            Log.w(TAG, "signInWithEmail:failed",
task.getException());
                            new LovelyInfoDialog(LoginActivity.this) {
                                @Override
                                public LovelyInfoDialog
setConfirmButtonText(String text) {

findView(com.yarolegovich.lovelydialog.R.id.ld_btn_confirm).setOnClickListene
r(new View.OnClickListener() {
                                        @Override
                                        public void onClick(View view) {
                                            dismiss();
                                        }
                                    });
                                    return super.setConfirmButtonText(text);
                                }
                            }
                                    .setTopColorRes(R.color.colorAccent)
                                    .setIcon(R.drawable.ic_person_low)
                                    .setTitle("Login false")
                                    .setMessage("Email not exist or wrong
password!")
```

```
                                    .setCancelable(false)
                                    .setConfirmButtonText("Ok")
                                    .show();
                        } else {
                            currentUserId =
FirebaseAuth.getInstance().getCurrentUser().getUid();
                            userToken =
FirebaseInstanceId.getInstance().getToken();

userReference.child(currentUserId).child("token").setValue(userToken);
                            saveUserInfo();
                            progressDialog.dismiss();
                            Intent intent = new Intent(LoginActivity.this,
TabsActivity.class);
                            intent.putExtra("selected_index", "0");
                            startActivity(intent);
                            LoginActivity.this.finish();


                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                progressDialog.dismiss();
            }
        });
    }
```

Фрагмент коду, який відповідає за відображення всіх користувачів:

```
private void getUsers() {

    databaseReference.addListenerForSingleValueEvent(new ValueEventListener()
{
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            Iterator<DataSnapshot> dataSnapshotIterator =
dataSnapshot.getChildren().iterator();
            while (dataSnapshotIterator.hasNext()) {
                DataSnapshot dataSnapshotChild = dataSnapshotIterator.next();
                User user = new User();

user.setName(dataSnapshotChild.child("name").getValue(String.class));

user.setEmail(dataSnapshotChild.child("email").getValue(String.class));

user.setAvatar(dataSnapshotChild.child("avatar").getValue(String.class));

user.setBioText(dataSnapshotChild.child("bioText").getValue(String.class));
                user.setUid(dataSnapshotChild.getKey().toString());

                if (!TextUtils.equals(user.getUid(),
FirebaseAuth.getInstance().getCurrentUser().getUid())) {
                    users.add(user);
                }
```

```
            }
            userListingRecyclerAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}
```

Фрагмент коду, який виконується при надсиланні повідомлення:

```
String currentUid = FirebaseAuth.getInstance().getCurrentUser().getUid();
  if (view.getId() == R.id.btn_send) {
    String content = editTextMessage.getText().toString().trim();
    if (content.length() > 0) {
      recyclerEmojiAndSmart.setVisibility(View.GONE);
      editTextMessage.setText("");
      Message message = new Message();
      message.text = CipherHandler.encrypt(content);
      message.type = Message.TEXT;
      message.idSender = currentUid;
      message.idReceiver = idFriend.get(0).toString();
      message.idReceiverRoom = roomId;
      message.timestamp = System.currentTimeMillis();
      messageReference.child(roomId).push().setValue(message);
    }

    if (dataSnapshot.getValue() != null) {
        HashMap messageMap = (HashMap) dataSnapshot.getValue();
        Message message = new Message();
        message.idSender = (String) messageMap.get("idSender");
        message.idReceiver = (String) messageMap.get("idReceiver");
        message.idReceiverRoom = (String) messageMap.get("idReceiverRoom");
        message.text = (String) messageMap.get("text");
        message.type = (long) messageMap.get("type");
        message.timestamp = (long) messageMap.get("timestamp");
        DatabaseReference user1 = usersReference.child((String)
messageMap.get("idSender"));
        DatabaseReference user2 = usersReference.child((String)
messageMap.get("idReceiver"));
        user1.child("message").setValue(message);
        user2.child("message").setValue(message);

        conversation.getMessages().add(message);
        adapter.notifyDataSetChanged();

        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(ChatActivity.this, LinearLayoutManager.HORIZONTAL,
false);
        }
}
```

Реалізація шифрування:

```
public class CipherHandler {

    private static final String SEED_KEY = "#R@VeN % Me$$@ge % 
UnBrE@K@Ble!#";
    private static final SecretKeySpec SECRET_KEY_SPEC = new 
SecretKeySpec(SEED_KEY.getBytes(), "AES");
    private static final String ALGORITHM = "AES/ECB/PKCS5Padding";
    private static Cipher cipherE, cipherD;


    public static String encrypt(final String message) {
        try {
            cipherE = Cipher.getInstance(ALGORITHM);
            cipherE.init(Cipher.ENCRYPT_MODE, SECRET_KEY_SPEC);

            byte[] encryptedBytes = cipherE.doFinal(message.getBytes());
            String base64 = Base64.encodeToString(encryptedBytes, 
Base64.DEFAULT);
            return base64;
        } catch (Exception e) {
            e.getMessage();
        }
        return message;
    }

    public static String decrypt(final String message) {
        try {
            cipherD = Cipher.getInstance(ALGORITHM);
            cipherD.init(Cipher.DECRYPT_MODE, SECRET_KEY_SPEC);
            byte[] raw = Base64.decode(message, Base64.DEFAULT);

            byte[] decryptedBytes = cipherD.doFinal(raw);
            String decryptedMessage = new String(decryptedBytes, "UTF8");
            return decryptedMessage;
        } catch (Exception e) {
            e.getMessage();
        }
        return message;
    }

}
```