ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка

до магістерської роботи
на тему:

«ТЕХНОЛОГІЇ ЗАХИСТУ ІНФОРМАЦІЇ ТА ПРОТИДІЇ СПАМУ В СЕРВІСІ TELEGRAM»

Виконав: студент 6 курсу, групи БСДМ-62
Спеціальності 125 Кібербезпека
Освітньо-професійної програми «Інформаційна та кібернетична безпека»
(шифр і назва спеціальності)
Олійник Є.О.
(прізвище та ініціали)
Керівник Довженко Н.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтролер Чумак Н.С.
(прізвище та ініціали)

КИЇВ – 2021

РЕФЕРАТ

Текстова частина магістерської роботи: 66 сторінок, 29 рисунків, 2 таблиці, 26 джерел.

Об'єкт дослідження – сервіс для миттєвого обміну повідомленнями Telegram.

Предмет дослідження – технології захисту інформації та протидії спаму в сервісі Telegram.

Мета роботи – розробити алгоритм функціонування власного боту для захисту інформації та протидії спаму в сервісі Telegram.

Методи дослідження – опрацювання літератури за даною темою, теорія інформації, практичне тестування програмного забезпечення.

В роботі описано особливості налаштувань безпеки в сервісі для миттєвого обміну повідомленнями Telegram. Досліджено протоколи для безпечного обміну повідомленнями, які забезпечують наскрізне шифрування повідомлень.

Розглянуто приклади і методи захисту від небажаного трафіку в сервісі Telegram. Визначено, що причиною отримання спаму не завжди є добре спланована атака. Іноді самі учасники групи або користувачі каналу ненавмисно додають бота, відкриваючи таким чином простір для активності.

Продемонстровано варіанти створення власного бота за допомогою наявних прикладів, які вже існують в середовищі сервісу Telegram. При цьому зазначено, що ці боти можуть бути вразливі до спаму та фішингу через тих розробників, які над ними працювали. Приведено алгоритм розробки власного боту, а також особливості реєстрації його на власному ПК. В цій реалізації не виникає сумнівів щодо виконання розробленим ботом всіх тих функцій, які потрібні користувачам. Адже достовірність кожної із команд може бути гарантована саме розробником.

Галузь використання – кібербезпека.

TELEGRAM, ІНФОРМАЦІЯ, БОТИ, СПАМ, ФІШИНГ, МЕСЕНДЖЕР, СЕРВІС, КІБЕРАТАКА, БЕЗПЕКА, АЛГОРИТМ, МЕТОД, ШИФРУВАННЯ.

ABSTRACT

Master's thesis: 66 pages, 29 figures, 2 tables, 26 sources.

Object of research – the service for instant messaging Telegram

Subject of research – the information protection technologies and anti-spam in the Telegram service.

The aim of research – to develop the operation algorithm of a bot to protect information and counteract spam in the Telegram service.

Research methods – elaboration of literature on the topic, information theory, practical software testing.

The paper describes the features of security settings in the service for the instant messaging Telegram. Protocols for secure messaging that provide end-to-end message encryption have been explored.

Examples and methods of protection against unwanted traffic in the Telegram service are considered. It is determined that the reason for receiving spam is not always a well-planned attack. Sometimes the group members themselves or channel users inadvertently add a bot, thus opening up space for activity.

Options for creating a bot using existing examples that already exist in the Telegram service environment are demonstrated. It is noted that these bots may be vulnerable to spam and phishing due to the developers who worked on them. The algorithm development of own bot, and also features of its registration on own personal computer is resulted. In this implementation, there is no doubt that the developed bot performs all the functions that users need. After all, the authenticity of each of the teams can be guaranteed by the developer.

Field of use – cybersecurity.

TELEGRAM, INFORMATION, BOTS, SPAM, FISHING, MESSAGE, SERVICE, CYBERATRACK, SECURITY, ALGORITHM, METHOD, ENCRYPTION.

ЗМІСТ

ВСТУП

Актуальність дослідження. За останні роки тенденція використання мобільних додатків для спілкування зросла і стала стандартним методом спілкування. Почали з’являтися нові програми для обміну повідомленнями, які намагалися замінити традиційні SMS, але створення їх з урахуванням безпеки та конфіденційності не було важливим для розробників.

Популярні інструменти для миттєвого обміну повідомленнями, які використовуються останніми роками, не підтримують наскрізне шифрування, а лише стандартне шифрування від клієнта до сервера, що надає постачальникам послуг доступ до більш конфіденційних даних, ніж необхідно.

Кінцеві користувачі почали використовувати безпечні мобільні додатки для обміну повідомленнями після справи Сноудена, однак вирішальним стало впровадження не лише надійного спілкування між користувачами, але й хороший користувальницький софт із зручними для розуміння властивостями.

Нові безпечні мобільні додатки, такі як Viber, Whatsapp та Telegram, прості у використанні кінцевими користувачами, коли їм потрібен лише номер телефону для створення облікового запису, а програми генерують та обмінюються криптографічними ключами у фоновому режимі, без жодної взаємодії з користувачем.

Новою проблемою, що поступово виокремилася і в сучасних версіях програм для миттєвого обміну повідомленнями, стала проблема спаму. Це сталося через неефективне функціонування державної системи забезпечення безпеки користувачів таких додатків та необізнаність самих користувачів в умовах розвитку цифровізації та диджиталізації. Джерелом більшості кібератак в сучасних месенджерах є спам, адже вони розраховані саме на неуважність та необізнаність користувачів.

Об’єкт дослідження – сервіс для миттєвого обміну повідомленнями Telegram.

Предмет дослідження – технології захисту інформації та протидії спаму в сервісі Telegram.

Мета роботи – розробити алгоритм функціонування власного боту для захисту інформації та протидії спаму в сервісі Telegram.

Відповідно до мети наукового дослідження були поставлені та розв’язані наступні завдання:

- описано особливості налаштувань безпеки в сервісі для миттєвого обміну повідомленнями Telegram;
- досліджено сервіс миттевого обміну повідомленнями Telegram;
- досліджено протоколи для безпечного обміну повідомленнями, які забезпечують наскрізне шифрування повідомлень.
- досліджено джерела спаму;
- продемонстровано варіанти створення власного бота за допомогою наявних прикладів, які вже існують в середовищі сервісу Telegram.

Результати, досягнуті в процесі роботи дозволяють автоматизувати процеси підтримки прийняття рішень в управлінні безпекою групп користувачів.

Методи дослідження – опрацювання літератури за даною темою, теорія інформації, практичне тестування програмного забезпечення.

1 АНАЛІЗ ОСОБЛИВОСТЕЙ ФУНКЦІОНУВАННЯ СЕРВІСУ TELEGRAM

1.1. Виокремлення об’єкту дослідження

Telegram - це програма для миттєвого обміну повідомленнями, орієнтована на швидкість та безпеку користувачів. Головною особливістю цього сервісу є одночасне використання на декількох пристроях - всі повідомлення безперешкодно синхронізуються на будь-якій кількості телефонів, планшетів чи комп’ютерів.

За допомогою Telegram користувачі можуть постійно обмінюватися повідомленнями, фотографіями, відео та файлами будь-якого типу (doc/docx, pdf, zip, rar, mp3/mp4, jpeg тощо), а також створювати групи до 200 000 людей або канали для трансляції необмеженій аудиторії. На додаток до цього підтримується сервіс шифрування голосових дзвінків.

Офіційна статистика Telegram наступна:

- 24 березня 2019 року Telegram надав можливість повністю видаляти повідомлення у себе та співрозмовника незалежно від терміну давності;
- 14 серпня 2020 року було оголошено про запуск альфа-версії відеодзвінків у програмах Telegram для Android та iOS;
- 23 грудня 2020 року Павло Дуров заявив про те, що месенджер Telegram з 2021 року почне монетизуватися. Причиною стало зростання витрат через велику кількість користувачів (понад 350 мільйонів користувачів);
- 30 вересня 2020 року вийшло оновлення офіційних клієнтів Telegram, де було змінено інтерфейс обговорення в каналах;
- У 2020 році було додано відеодзвінки. Відеодзвінки, як і голосові дзвінки, підтримують peer-to-peer;
- У листопаді 2020 року було презентовано можливість роботи Bot API на власному сервері;

- 12 січня 2021 року Павло Дуров повідомив, що першого тижня січня Telegram перевищив 500 мільйонів активних користувачів на місяць, і кількість користувачів тільки продовжує зростати;
- У вересні 2021 року у світі під час збоїв у роботі Instagram, Facebook та Whatsapp вперше було встановлено, що кількість користувчачів за добу перевищила 570 мільйонів користувачів [1].

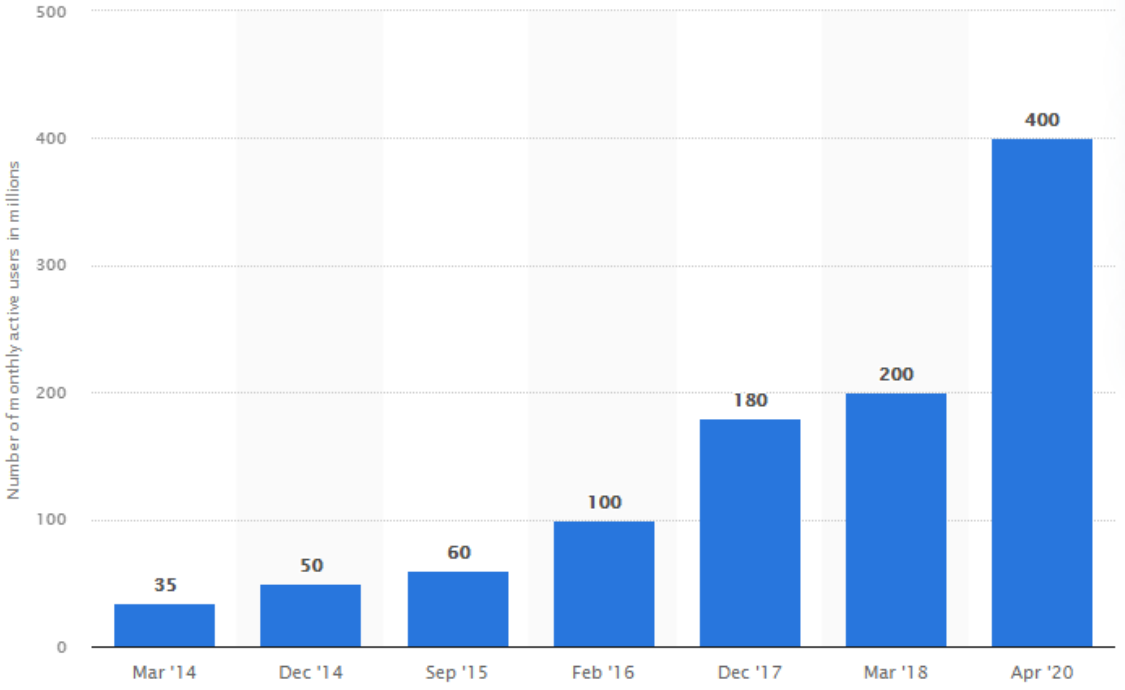

Рис.1.1. Ріст користувачів Telegram (березень 2014 року по квітень 2020 року)

З технічної точки зору спеціальні програми обміну миттєвими повідомленнями - це комп’ютерні програми, які використовують протоколи зв'язку в режимі реального часу, що забезпечують надійність і швидкість для передачі повідомлень в режимі реального часу.

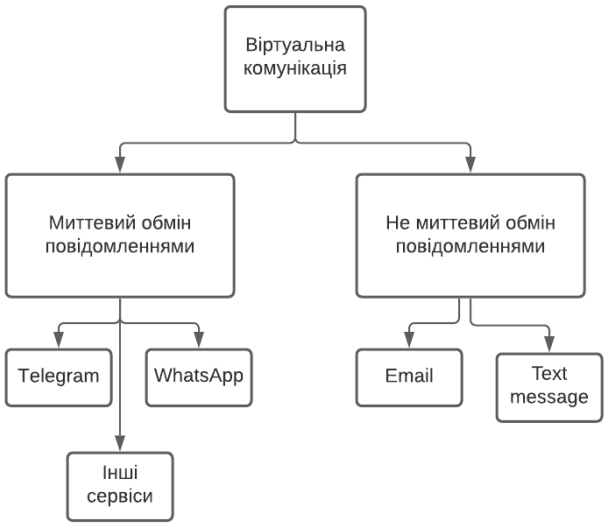

Рис.1.2. Концепція віртуальної комунікації

Як видно з рис.1.2, віртуальне спілкування можна розділити на два різні типи концепцій обміну повідомленнями. Концепція архітектури віртуального спілкування може бути більш складною і враховувати комунікацію, що базується на людині, як текстове спілкування з другом, та комп'ютерне спілкування, як текстове спілкування за допомогою чат-ботів.

Telegram — це платформа для обміну миттєвими повідомленнями, яка була розроблена в 2013 році після скандалу з АНБ. Telegram був розроблений для функціонування на смартфонах, планшетах та ПК [2]. Організовуючи комунікацію з однією людиною, груповий чат та можливість надсилати файли декільком людям із контактів. Різниця між Telegram та іншими сервісами та додатками для обміну миттєвими повідомленнями полягає в тому, що він пропонує лише безпечні повідомлення.

Коли справа доходить до безпечних чатів, вони реалізували власну версію криптографічного протоколу, який вони назвали протоколом MTProto. Цей же протокол також використовується для звичайних хмарних чатів для шифрування зв'язку між сервером і клієнтом.

Наскрізні зашифровані чати, які надає Telegram, не дозволяють користувачам робити знімки екрана всередині секретного чату. Тому зображення під час розмови знімаються зовнішньою камерою.

1.2. Особливості початкового налаштування сервісу Telegram

Початкове налаштування програми Telegram та реєстрація користувачів такі ж, як і в інших сервісах. На рис.1.3. показано, де користувач вводить свій номер телефону для реєстрації, а на рис.1.4. показано процес активації.

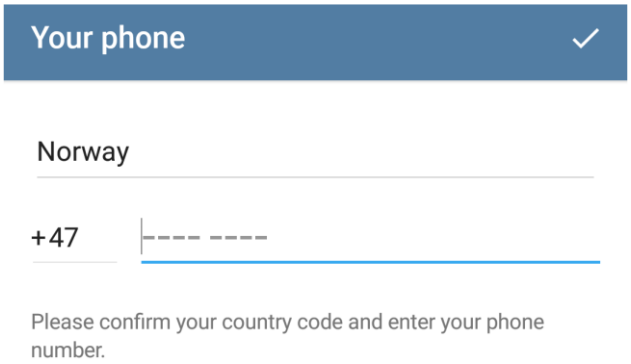

Рис.1.3. Реєстрація в Telegram за допомогою мобільного телефону

Telegram надсилає код активації за допомогою SMS, який можна ввести вручну або надати Telegram доступ, щоб зробити це автоматично, і якщо повідомлення не отримає користувач протягом наступних двох хвилин, система надсилає нове повідомлення. Якщо користувач не отримує код підтвердження по SMS, він може попросити Telegram зателефонувати користувачеві та активувати його за допомогою телефонного дзвінка.

У Telegram за замовчуванням не ввімкнено наскрізне шифрування, а це означає, що одному із користувачів (наприклад, Алісі) потрібно почати секретний

чат із іншим користувачем (наприклад, Бобом). Звичайні повідомлення, які в Telegram називають хмарними чатами, не мають жодного шифрування.

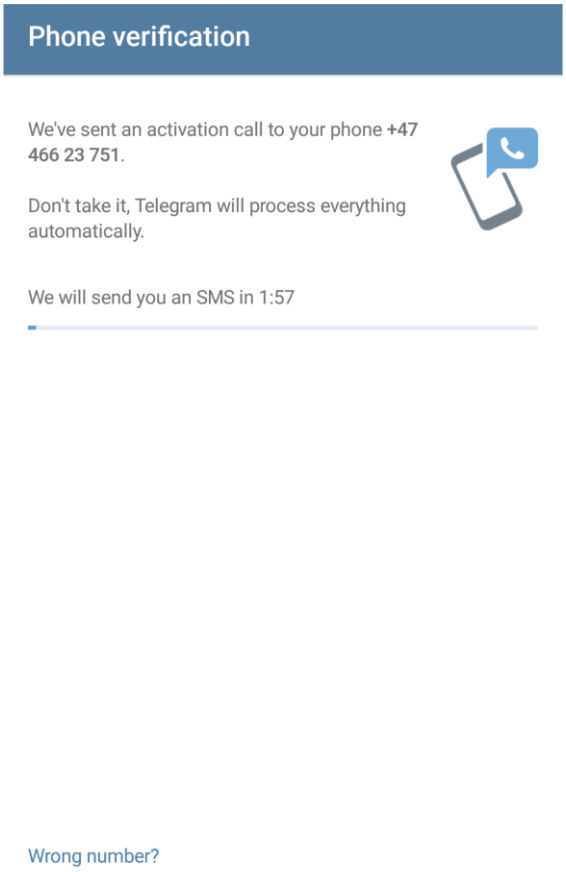

Рис.1.4. Верифікація в Telegram за допомогою мобільного телефону

На рис.1.5. показано перше повідомлення, яке отримує користувач Аліса, коли починає спілкування з користувачем Бобом. Telegram надає інформацію про секретний чат, що він зашифрований наскрізним шифруванням і що він не дозволяє пересилати повідомлення з міркувань безпеки.

На рис.1.6. показано, як користувач Аліса ініціює безпечну розмову в чаті з користувачем Бобом, і подвійні позначки показують, що Боб отримав і прочитав повідомлення, а одна позначка ілюструє, що повідомлення було надіслано [3].

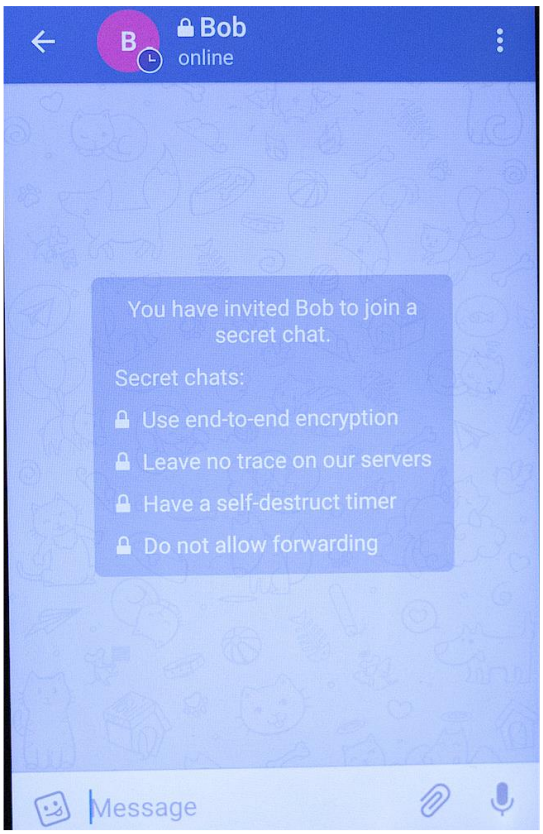

Рис.1.5. Ініціація контактів

Користувач Аліса надсилає повідомлення-рекомендацію користувачеві Бобу перевстановити програму, щоб перевірити, чи впливають ключові зміни на спілкування між ними. Аліса знає, що Боб перевстановив свою програму, і намагається надіслати нове повідомлення Бобу, як показано на рис.1.6.

Боб так і не отримав повідомлення, навіть після того, як він перевстановив програму. Це показує, що Telegram використовує лише ті самі ключі під час встановлення програми, а якщо її видалено, пристрій втрачає ключі. Аліса надсилає повідомлення зі старими ключами пристрою, і їй потрібно створити нові, ініціюючи новий секретний чат із Бобом, щоб це спрацювало, та обмінявшись новими ключами безпеки.

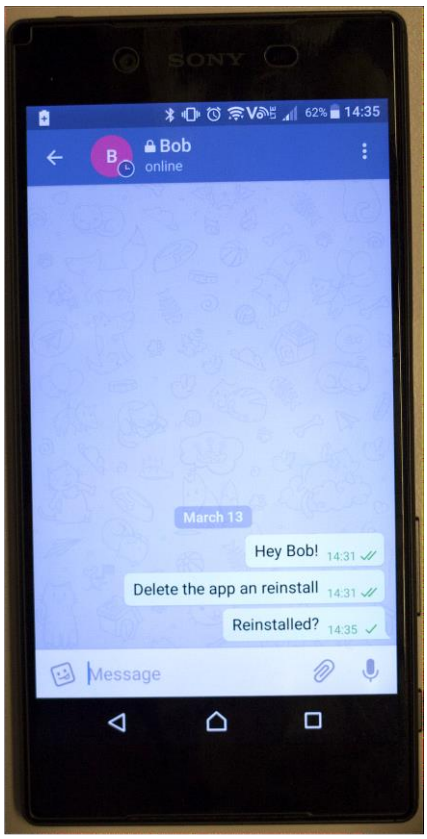

Рис.1.6. Повідомлення після перевстановлення

Telegram ніде не зберігає ключі або будь-яку іншу інформацію про те, що двоє користувачів мали секретний чат, щоб перевірити, чи хтось із користувачів перевстановили програму чи ні. Якщо ключі буде втрачено або безпечно видалено одним із учасників, все залежить від користувачів.

1.3. Зміна ключів під час передачі повідомлення в сервісі Telegram

Telegram не зберігає жодної інформації про те, що користувачі (Аліса або Боб) вели секретний чат, усе це робить клієнт, і на сервер нічого не надсилається, говорячи, що вони мали розмову між їх.

Процес перевірки між учасниками. Процес верифікації між користувачами Алісою і Бобом досить складний при використанні секретних чатів у Telegram. Якщо Аліса хоче підтвердити ключ шифрування Боба, їй потрібно перейти на сторінку налаштувань безпечних чатів і натиснути кнопку «Ключ шифрування», тоді з’явиться сторінка підтвердження. На рис.1.7. показана сторінка перевірки із

зображенням, отриманим із ключа шифрування, та ключем шифрування, наведеним нижче [4].

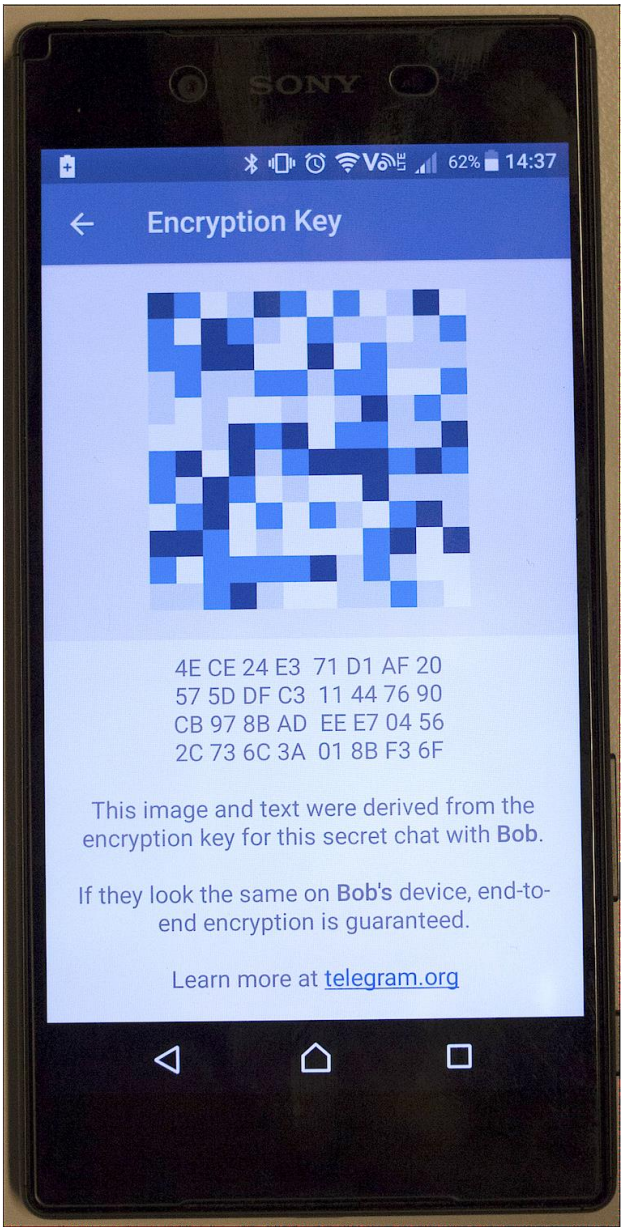

Рис.1.7. Процес перевірки (верифікації) в сервісі Telegram

Сторінка перевірки мала використовувати справжній QR-код, так само, як це роблять сервіси Signal і WhatsApp, і реалізувати сканер QR-коду, який може сканувати код і перевіряти, що це правильний ключ шифрування.

Telegram підтримує багато інших реалізацій безпеки. На сторінці налаштувань є можливість переглянути налаштування «Конфіденційність та безпека» для програми. На рис.1.8. показано налаштування, які можна змінити в програмі.

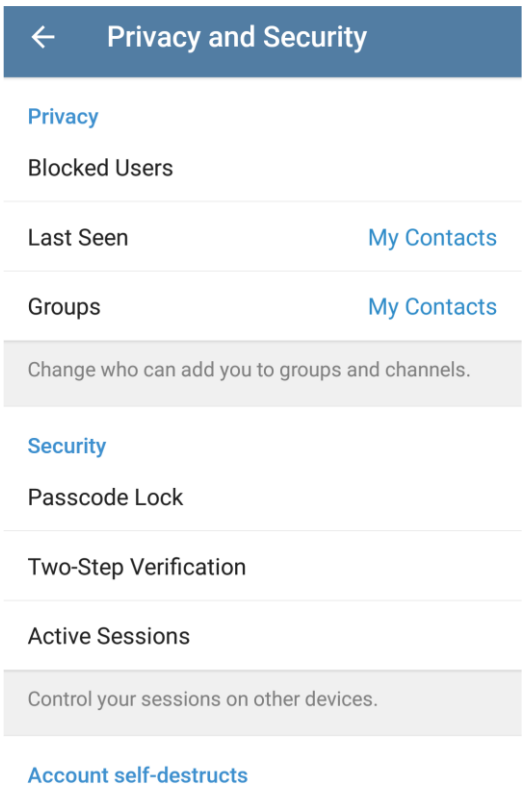

Рис.1.8. Налаштування безпеки в сервісі

Telegram підтримує двоетапну перевірку, тобто коли користувач хоче увійти на іншому пристрої або після перевстановлення, йому потрібно ввести другий, особисто обраний пароль після коду активації, отриманого в SMS. «Активні сеанси» – це список пристроїв, на які користувач увійшов. Останній варіант «Акаунт самознищується» — це вимірювання безпеки. Якщо користувач не використовував свій обліковий запис протягом останніх шести місяців-рік, обліковий запис видаляється Telegram. Тривалість цього кроку (для самознищення) можна змінити на один місяць, три місяці, півроку або один рік.

На рис.1.9. показані параметри, які користувач отримує, натиснувши «Блокування паролем» у списку налаштувань «Конфіденційність та безпека». Ця функція блокує всю програму за допомогою пароля, який вибирає користувач. Telegram реалізував можливість розблокування програми за відбитком пальця, якщо користувач додав відбиток пальця в операційну систему, у нашому випадку систему Android. Користувач має можливість змінити час автоматичного блокування програми, від однієї хвилини до п’яти годин. Останній показаний

варіант — «Дозволити знімок екрана», який, якщо ввімкнено, дозволяє користувачам робити знімки екрана будь-що всередині програми, а якщо його не ввімкнено, вони не мають доступу до цього, за винятком безпечних чатів, яким ніколи не дозволено робити знімки екрана.

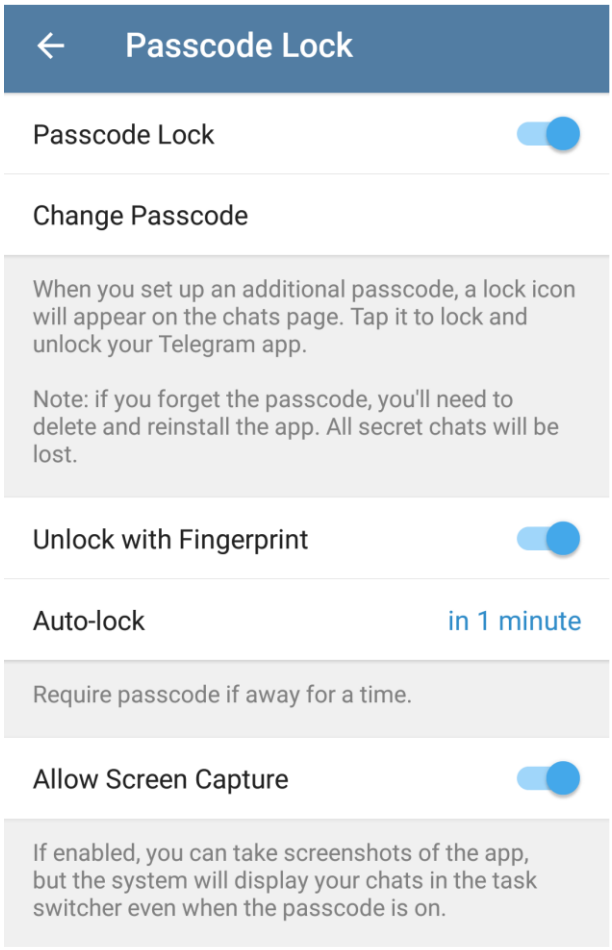

Рис.1.9. Додаткові налаштування безпеки в Telegram

1.4. Вимоги до виконання алгоритмів та протоколів безпеки в сервісі Telegram

Аутентифікація. Це ідентифікація осіб, яка необіхдна для підтвердження того, що особа з якою ведеться спілкування саме та, за кого себе видає. Якщо один користувач надсилає повідомлення іншому користувачеві, і повідомлення змінюється в процесі пересилання до приймача, то приймач зафіксує модифікацію. Аутентифікацію повідомлень також називають автентифікацією походження

даних. Аутентифікація повідомлень захищає цілісність повідомлення, щоб гарантувати, що кожне отримане повідомлення перебуває в тому ж стані, що й було надіслано, без жодних змін до повідомлення [6].

Коди аутентифікації повідомлень (MAC) забезпечують гарантію щодо джерела та цілісності повідомлення. Код аутентифікації повідомлення обчислюється за допомогою повідомлення та спільного секретного ключа між двома сторонами.

Це означає, що для аутентифікації повідомлення одержувач повинен поділитися з відправником секретним ключем, який використовується для обчислення коду аутентифікації повідомлення. Зловмисник не може підтвердити повідомлення, оскільки тільки відправник і одержувач мають загальний секрет, і якщо зловмисник змінює повідомлення, то змінюється також і обчислений ключ MAC.

Конфіденційність. Конфіденційність забезпечує дотримання необхідного рівня секретності на кожному етапі обробки даних і запобігає несанкціонованому розголошенню. Зазвичай це досягається шляхом шифрування даних від відправника до одержувача, і тільки ті, у кого є правильний ключ дешифрування, можуть прочитати, що містять зашифровані дані. У криптографічних протоколах конфіденційність є важливою для забезпечення того, щоб ключі та інші дані були доступні лише за призначенням.

Однак зловмисники мають можливість зруйнувати механізми конфіденційності, викравши файли паролів, зламавши схеми шифрування або за допомогою соціальної інженерії. З іншого боку, користувачі можуть навмисно або випадково розкрити конфіденційну інформацію, не зашифрувавши її перед відправкою іншій особі, або стати жертвою атаки соціальної інженерії. Конфіденційність може бути забезпечена шляхом шифрування даних під час їх зберігання та передавання, забезпечуючи суворий контроль доступу та класифікацію даних, а також навчання персоналу належним процедурам захисту даних [5].

Цілісність. Це гарантія того, що будь-хто протягом передачі не змінює повідомлення та його зміст. Будь-яка сторона ні за яких обставин не повинна приймати змінене повідомлення. Апаратне забезпечення, програмне забезпечення та механізми зв'язку повинні працювати узгоджено, щоб правильно підтримувати та обробляти дані та переміщувати дані в цільові місця без несподіваних змін. Система і мережа повинні бути захищені від зовнішнього втручання та компроментації.

Середовища, які забезпечують цей атрибут безпеки, гарантують, що зловмисники або помилки користувачів не порушують цілісність систем або даних. Цього можна досягти шляхом використання хеш-функції у поєднанні з шифруванням або за допомогою коду аутентифікації повідомлення (MAC) для створення окремого поля перевірки. Цілісність даних є формою цілісності, яка є важливою для більшості криптографічних протоколів для захисту таких елементів, як поле ідентифікації або одноразові номери.

Підтвердження секретності. Ідея секретності пересилання полягає в тому, що коли довгостроковий ключ скомпрометований, ключі сеансів, які раніше були встановлені за допомогою цього довгострокового ключа, не повинні бути порушені. Типовим прикладом протоколів, які забезпечують пряму секретність, є протоколи угоди про ключ, де довгостроковий ключ є лише використовується для аутентифікації обміну. Протоколи транспортування ключів, у яких довгостроковий ключ використовується для шифрування ключа сеансу, не можуть забезпечити пряму секретність [6].

Протокол встановлення ключів забезпечує пряму секретність, якщо компрометація довгострокових ключів набору принципалів не ставить під загрозу ключі сеансу, встановлені під час попередніх запусків протоколу за участю цих принципалів.

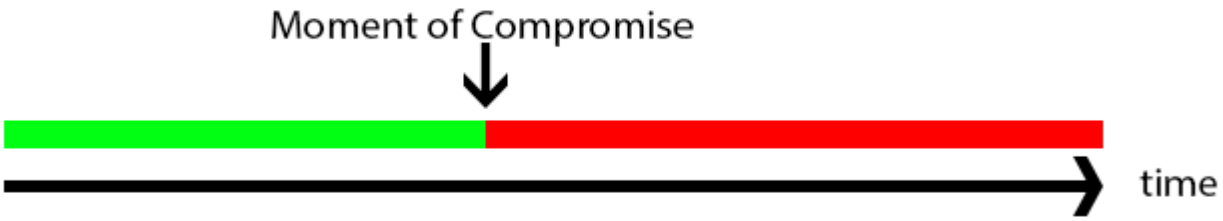

Рис.1.10. Підтвердження секретності

Зворотна секретність. Зворотна секретність або майбутня секретність – це покроковий етап, коли протокол може гарантувати, що компроміс щодо довгострокових ключів не дозволить розшифрувати наступні шифротексти пасивним зловмисником. Протокол також підтримує майбутню секретність, коли він може забезпечити «самовідновлювальний» аспект кодом Діффі-Хеллмана, оскільки якщо будь-який ефемерний ключ скомпрометований або виявляється слабким у будь-який момент мессенджер зафіксує це і обчислить нові ключі для решти повідомлень, надісланих під час розмови.

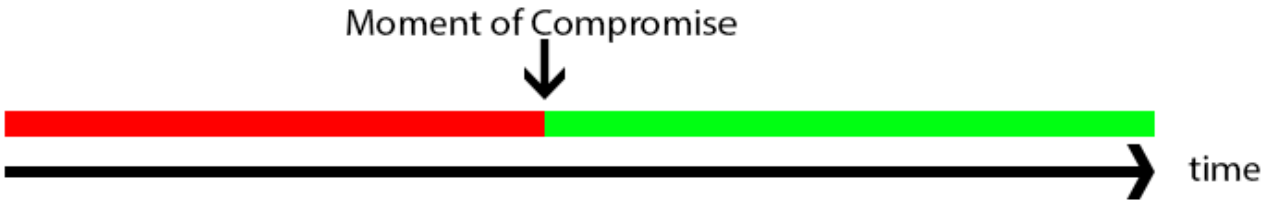

Рис.1.11. Зворотна секретність

Відмовність. Це властивість, спільна для нових протоколів безпечного обміну повідомленнями, коли інші не можуть підтвердити, що дані були надіслані однією конкретною особою. Якщо користувач Боб отримує повідомлення від користувача Аліси, він може бути впевнений, що його надіслала саме Аліса, але не може довести нікому, що це написала Аліса. Захищені протоколи обміну повідомленнями, які пропонують відмовність, можуть запевнити користувача, що

будь-хто може підробити повідомлення після розмови, щоб вони виглядали так, ніби вони надійшли від нього, але якщо це відбувається під час розмови, учасники впевнені, що слова, які вони бачать, є справжніми і не є такими. змінені кимось іншим, крім них самих.

Синхронність. Існує два типи зв'язку - синхронний та асинхронний, і протокол чату може бути організований одним із них. Синхронні протоколи мають вимогу, щоб усі учасники були онлайн, щоб вони могли отримувати або надсилати повідомлення. Якщо протокол чату асинхронний, це означає, що учасникам не потрібно бути в мережі для отримання повідомлень, таких як текстові SMS-повідомлення або електронні листи, оскільки є третій комронент мережі, який зберігає інформацію, доки одержувач знову не під'єднається до мережі.

Коли справа доходить до сучасних протоколів, не рекомендується використовувати асинхронний протокол. Причина в тому, що існують соціальні та технічні обмеження, такі як акумулятор пристрою, обмежений прийом або інші соціальні проблеми події, які означають, що у людей завжди будуть проблеми з підтримкою онлайн для отримання повідомлень. Ось чому більшість рішень для обміну миттєвими повідомленнями забезпечує асинхронне середовище, маючи сторонній сервер, який зберігає повідомлення, поки інший учасник не під'єднається до мережі Інтернет, щоб отримати їх. Нові безпечні протоколи обміну повідомленнями повинні бути асинхронною платформою для спілкування, щоб вони стали популярними серед кінцевих користувачів.

Додаткові властивості безпеки. Ці властивості безпеки менші та стисліші, ніж інші властивості, але все ж таки важливі для впровадження безпеки протоколів обміну повідомленнями.

- Послідовність учасників. У будь-який момент, коли повідомлення приймається достовірною стороною, усі достовірні сторони гарантовано матимуть однаковий список учасників.
- Перевірка місця призначення. Коли повідомлення приймається достовірною стороною, вона може перевірити, чи було воно включено до набору призначених одержувачів повідомлення.

- Збереження анонімності. Будь-які функції анонімності, що надаються базовою архітектурою конфіденційності трафіку, не підриваються (наприклад, якщо система конфіденційності трафіку забезпечує анонімність, рівень безпеки розмови не деанонімізує користувачів шляхом зв’язування ідентифікаторів ключів).
- Послідовність спікера. Всі учасники погоджуються щодо послідовності повідомлень, надісланих кожним учасником. Протокол може виконувати перевірку узгодженості блоків повідомлень під час роботи протоколу або після надсилання кожного повідомлення.
- Збереження причинно-наслідкових зв'язків. Реалізації можуть уникнути відображення повідомлення перед повідомленнями, які причинно передують йому.
- Глобальна розшифровка. Всі учасники бачать всі повідомлення в однаковому порядку. Коли ця функція безпеки гарантована, це означає, що гарантовано як узгодженість динаміки, так і збереження причинно-наслідкового зв’язку.

1.5. Особливості організації групових чатів в сервісі Telegram

Останніми роками стало легше вести групову бесіду з друзями та колегами за допомогою Facebook Messenger, Slack або інших популярних програм для обміну повідомленнями.

Проблема з цими програмами обміну повідомленнями полягає в тому, що вони не шифруються наскрізьним шифруванням, але, наприклад, Open Whisper Systems наполегливо попрацювали, щоб запровадити швидкі та надійні наскрізні зашифровані групові чати. Тому важливо, щоб нові протоколи та програми мали додаткові властивості безпеки та функції для підтримки наскрізного шифрування.

- Обчислювальна рівність: чи поділяють учасники однакове обчислювальне навантаження під час розмови один з одним.

- Рівність довіри: немає жодного учасника, який мав би більше довіри або відповідальності всередині групи, ніж будь-який інший.
- Обмін повідомленнями підгрупи: учасники можуть надсилати повідомлення лише іншій підгрупі без створення нової розмови.
- Договірне членство: групі не потрібно перезапускати протокол безпеки, коли учасник залишає розмову.
- Розширюване членство: немає необхідності перезапускати протокол безпеки під час додавання нового члена після створення групи.

Важливо, щоб протокол мав можливість змінювати криптографічні ключі, коли новий користувач приєднується до безпечної групової розмови, оскільки тоді нові користувачі не мають можливості розшифрувати раніше надіслані повідомлення.

Новими криптографічними ключами також слід обмінюватися, коли користувач залишає розмову. Зміна ключів не є такою великою проблемою, це просто перезапуск протоколу, але це часто обчислювально дорого. Протоколи, які пропонують членство, яке можна стягнути та розширити, досягають цих функцій без перезапуску протоколу.

Інші аспекти зручності використання слід брати до уваги, розглядаючи зручність використання та впровадження. Йдеться про інші фактори зручності використання, такі як стійкість, підтримка кількох пристроїв і якщо протокол потребує додаткових послуг.

- Стійкість до невпорядкованості: якщо повідомлення затримується в маршруті, але в кінцевому підсумку прибуває, його вміст стає доступним після прибуття.
- Стійкість до скинутих повідомлень: повідомлення можна розшифрувати без отримання всіх попередніх повідомлень. Це бажано для асинхронних і ненадійних мережевих служб.
- Асинхронний: повідомлення можна безпечно надсилати на пристрої, які не підключені до мережі Інтернет на момент надсилання.

- Підтримка кількох пристроїв: користувач може підключатися до розмови з кількох пристроїв одночасно і мати такий самий погляд на розмову, як і інші.
- Без додаткових послуг: протокол не вимагає іншої інфраструктури, крім учасників протоколу. Зокрема, протокол не повинен вимагати додаткових серверів для передачі повідомлень або зберігання будь-який ключовий матеріал.

Висновки до першого розділу

Описано особливості налаштувань безпеки в сервісі для миттєвого обміну повідомленнями Telegram. Проілюстровано особливості функціонування сервісу та зазначено головні принципи безпеки, які повинна мати безпечна система обміну повідомленнями, щоб її можна було назвати безпечною.

Розглянуто аутентифікацію, конфіденційність, цілісність, а також синхроність, стійкість, відмовність та асинхронність. Дано огляд груповим чатам, які важливі для кінцевого користувача, та підкреслено, що різні протоколи також повинні мати додаткові функції для підтримки функцій безпеки та конфіденційності.

2 ДОСЛІДЖЕННЯ ПРОТОКОЛІВ БЕЗПЕЧНОГО ОБМІНУ ПОВІДОМЛЕННЯМИ В СЕРВІСІ TELEGRAM

Оскільки смартфони почали широко використовувати наприкінці 2000-х років, ряд сервісів для миттєвого обміну повідомленнями (IM), таких як WhatsApp, KakaoTalk, LINE, Facebook Messenger і Telegram, увірвалися в магазини мобільних додатків. Різні клієнти миттєвих повідомлень можна класифікувати на три типи відповідно до наданих ними протоколів шифрування: без шифрування, шифрування клієнт-сервер і шифрування між клієнтом або наскрізне шифрування (E2EE). Оскільки відсутність захисту конфіденційності видається постійно, зараз більшість послуг IM надають E2EE на основі перевірених криптографічних протоколів. Telegram особливо вважається одним з найбільш безпечних публічних сервісів і має понад 500 мільйонів активних користувачів [6].

2.1. Виокремлення моделі загроз для сервіса Telegram

Оцінюючи властивості безпеки та конфіденційності безпечних протоколів обміну повідомленнями, можна припустити, що під час оцінки можуть виникнути численні загрози. В сервісі Telegram модель загроз може виникнути з боку таких зловмисників:

- Активні зловмисники. Атаки «Людина в середині» можливі як в локальних мережах, так і в глобальних мережах, додавши проксі між програмами та серверами, які обробляють повідомлення.
- Пасивні зловмисники. Ці зловмисники реєструють все, що надсилається користувачеві та від нього, і потенційно можуть використовувати цю інформацію, щоб відстежувати, з ким вони спілкуються і коли. Пасивні зловмисники також можуть реєструвати інформацію, таку як повідомлення та ключі, навіть якщо вміст повідомлень зашифровано.

- Постачальники послуг. Системи обміну повідомленнями, які потребують централізованої інфраструктури (наприклад, Signal і Matrix), повинні зберігати інформацію про користувачів у безпеці. Оператори сервісу в будь-який момент можуть стати потенційним супротивником.

Припускається, що кінцеві точки додатків, у які впроваджено ці протоколи, є безпечними і що пристрої користувача не мають жодного шкідливого програмного забезпечення чи експлойтів.

Проблемні зони в протоколах безпечного обміну повідомленнями поділяються на три окремі зони - безпека та конфіденційність, зручність використання та налаштування, а також функції групового чату. Функції зручності використання протоколів безпечного обміну повідомленнями були виключені, оскільки розглядаються різні функції зручності, які надають протоколи.

Розглядаються основні протоколи щодо того, як вони налаштовані, і чи обробляють вони шифрування, як це мається на увазі. Проходячи через оцінку протоколів, припускається, що протоколи реалізовані так, як вони розроблені, без будь-яких відомих експлойтів. Вибрані властивості безпеки та конфіденційності є основними властивостями, які досягаються за допомогою різних типів криптографічних примітивів [7].

2.2. Дослідження протоколу Off-the-Record

Даний протокол ззаснований на аутентифікованому обміні ключами Діффі-Хеллмана. Однак вони запровадили інший підхід до кроку ходу, додаючи нові включення до повідомлень. Коли користувачі генерують нові ефемерні ключі сеансу, а далі аутентифікують обмін за допомогою їх довгострокових ключів, спільний секрет із цього обміну ключами використовується для отримання ключа шифру відправника та одержувача для кожної сторони, а також набору ключів MAC для кожної сторони. Після цього протокол використовує їх для захисту повідомлень за допомогою підходу шифрування, а потім MAC, який в кінцевому

підсумку забезпечує конфіденційність, цілісність та аутентифікацію. Протокол SIGMA також додає функцію узгодженості участі для обміну ключами [8].

Протокол OTR підписує повідомлення спільними MAC-ключами, а не довгостроковими ключами, але для посилення можливості роз'єднання повідомлень і відмовлення від повідомлень вони також публікують ключі MAC і водночас використовують піддатливе шифрування. OTR підписує лише ефемерні ключі, а не кожен параметр під час обміну ключами; тоді він забезпечує часткову відмову від участі, оскільки партнери по розмові можуть використовувати підписані ефемерні ключі для підробки стенограм.

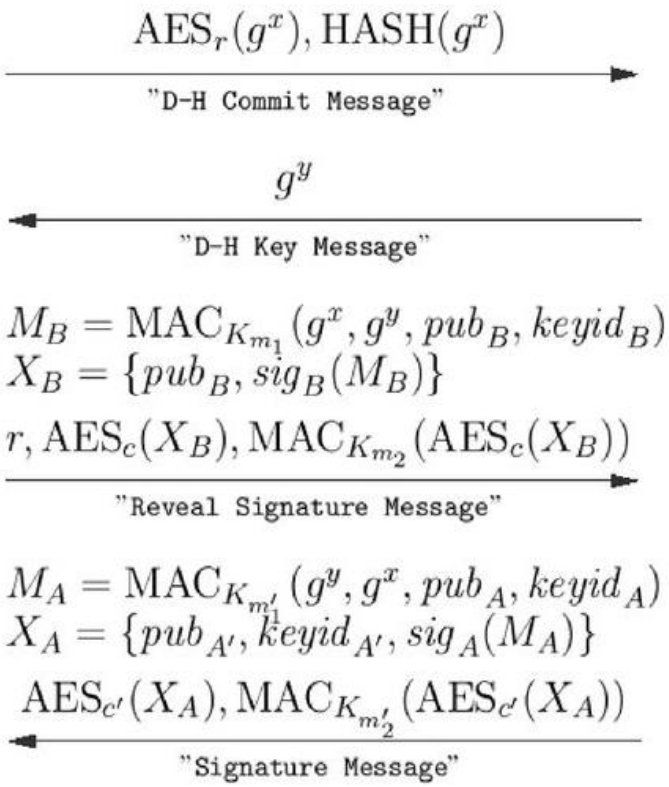

Рис.2.1. Приклад протоколу Off-The-Record (OTR)

Зворотна секретність забезпечується, коли ключі повідомлень обчислюються за новими значеннями DH, які оголошуються відправником з кожним надісланим повідомленням. Властивість безпеки надається через зламаний ключ регулярно замінюється новим ключовим матеріалом під час сеансу. Збереження анонімності забезпечується OTR, оскільки довгострокові відкриті ключі ніколи не спостерігаються ні під час обміну ключами, ні під час сеансу. Збереження

причинності частково досягається, оскільки повідомлення неявно посилаються на своїх причинно-наслідкових попередників на основі того, які ключі вони використовують.

Узгодженість частково досягається, оскільки зловмисник не може скинути повідомлення, не видаляючи також усі майбутні повідомлення, оскільки тоді одержувачі не зможуть розшифрувати наступні повідомлення. Наслідком послідовності є те, що одержувач повинен триматися за повідомленнями, що не в порядку, тому що, якщо вони не надходять у порядку, повідомлення буде зашифровано несподіваним ключем, і в той же час вікно компромісів збільшується, і OTR в кінцевому підсумку лише частково забезпечує пряму секретність. Якщо повідомлення виходить з ладу або відкидається під час передачі, протокол може зберігати ключ дешифрування, доки учасник не отримає його. Проблема зберігання ключа дешифрування полягає в тому, що він підвищує ймовірність успішних атак з боку зловмисників.

Протокол Off-The-Record створений для обміну миттєвими повідомленнями, який не забезпечує асинхронний обмін повідомленнями між учасниками, але через можливість синхронності протокол OTR не покладається на додаткові послуги для встановлення з’єднання між двома учасниками.

2.3. Дослідження протоколу Signal

Порушення даних є дуже реальною загрозою як для клієнтів, які хочуть захистити свою інформацію від крадіжки особистих даних, так і для компаній, які хочуть, щоб їх вважали надійними. Втрата інформації про клієнта означає не просто втрату продажу, а й потенційну постійну шкоду репутації вашої компанії, яку важко та дорого відновити.

Використання месенджерів на робочих місцях до втрати інформації, до якої користувачі отримали доступ: конфіденційна інформація щодо клієнтів (дані кредитних карток, адреси, паспортні дані, і т д).

Signal Protocol, який є відкритим вихідним кодом і не є федеративним, є криптографічним протоколом, який використовується при розробці зашифрованого чату. Він забезпечує E2EE для обміну миттєвими повідомленнями, відеодзвінків і голосових дзвінків. E2EE означає, що повідомлення шифруються перед відправленням і можуть бути розшифровані лише на пристрої передбачуваного одержувача. Протокол Signal складається з алгоритму Double Ratchet, 3-DH Handshake і попередніх ключів для асинхронної можливості.

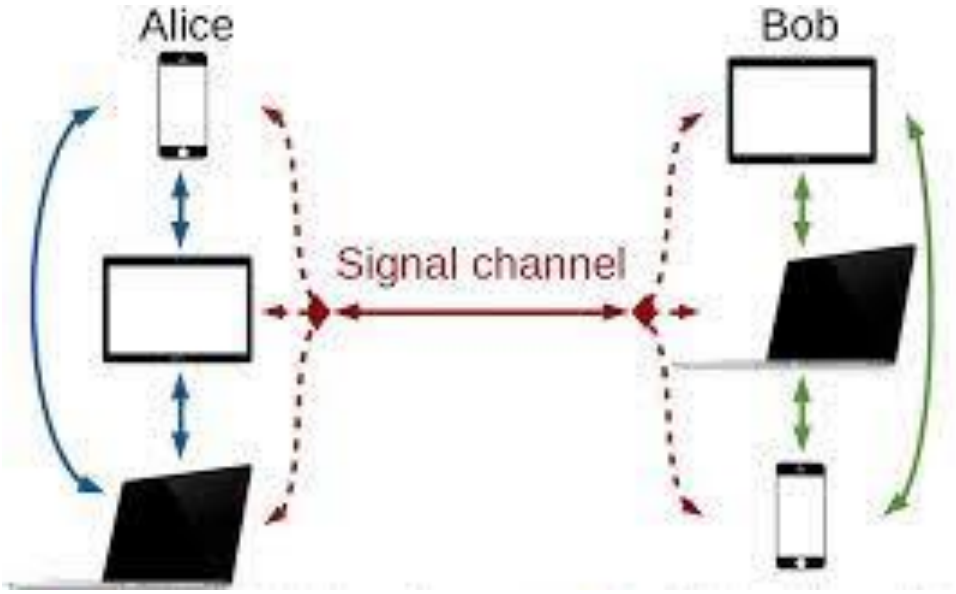

Рис.2.2. Приклад протоколу Signal

Попередня таємниця забезпечується, оскільки KDF фіксується, а зворотна секретність забезпечується, оскільки, навіть якщо ключі KDF містяться, вони в кінцевому підсумку будуть замінені новими ключами.

3-DH Handshake забезпечує такий же рівень аутентифікації, що і AKE від Sigma, але 3-DH вдається досягти повної відмови від участі, оскільки будь-хто може підробити стенограму між двома сторонами. Йому не вдається продовжити збереження анонімності, оскільки 3-DH використовує довгострокові відкриті ключі під час початкової угоди про ключ.

Ключі використовуються для досягнення асинхронної системи обміну повідомленнями шляхом відправки набору попередніх ключів (ефемерних загальнодоступних DH включень) на центральний сервер, а потім відправник може запитати наступний ключ для одержувача для обчислення ключів шифрування.

Використовуючи центральний сервер для збереження ключів, протокол Signal втрачає властивість відсутності додаткових послуг.

Повідомлення, що не в порядку, і пропущені повідомлення повністю підтримуються в розмовах один-на-один асинхронно за допомогою попередніх ключів, оскільки це реалізовано в алгоритмі Double Ratchet разом з узгодженням ключів X3DH. Набір Кількість попередніх ключів завантажується на центральний сервер, який зберігає їх безпечно та надсилає по одному користувачеві, який запитує ключ для шифрування повідомлення.

Груповий чат досягається за допомогою багатоадресного шифрування, яке при відправці одного зашифрованого повідомлення групі, воно надсилається на сервер, а потім передає його іншим учасникам під час розшифрування. Ключ надсилається як окреме повідомлення кожному учаснику групового чату. Груповий чат забезпечує асинхронний обмін повідомленнями, узгодженість та збереження причинно-наслідкових зв’язків, що досягається шляхом додавання ідентифікаторів попереднього повідомлення до нових повідомлень, але не може гарантувати узгодженість учасників. Signal частково надає багато пристроїв, у тому сенсі, що лише додатковий комп’ютер може приєднатися до розмови за допомогою програми Signal Desktop, яка є лише розширенням Chrome, а не власний додаток [9].

Протокол Signal забезпечує рівність обчислень і довіри, обмін повідомленнями підгруп, властивості, що стискаються та розширюються. Використовуючи попарний груповий обмін повідомленнями та багатоадресне шифрування, Signal отримує можливість передавати керування групами самим клієнтам, що полегшує користувачам зміну групи, розширення або зменшення її розміру, без необхідності перезапускати всю групову розмову та протокол.

Коли користувачі хочуть надіслати групове повідомлення, вони надсилають повідомлення кожному з користувачів, які беруть участь, і додають параметр до заголовка, який пояснює, що він призначений для конкретного групового чату. Сервер Signal не знає про групову бесіду, оскільки повідомлення шифруються за допомогою звичайного відкритого ключа. Попарний груповий обмін

повідомленнями також робить обчислення нових криптографічних ключів і довіри настільки ж вимогливим, як якщо б була лише розмова один на один.

2.4. Дослідження протоколу Matrix

Протокол Matrix був розроблений для реалізації мережі, побудованої на основі лінійної історії подій (events) всередині ациклічного графа (DAG). Найпоширенішим способом використання протоколу є реалізація серверів повідомлень (наприклад, сервер Synapse, клієнт Riot) та «з'єднання» інших протоколів один з одним за допомогою мостів (наприклад, реалізація на libpurple за допомогою XMPP, Telegram, Discord та IRC).

Протокол Matrix складається з двох різних алгоритмів: olm для розмов один на один і megolm для групових каналів/чатів між кількома пристроями. Алгоритм olm заснований на протоколі Signal Protocol, що означає, що вони досягають тих же властивостей безпеки, що й Signal, тоді як алгоритм megolm — це новий криптографічний варіант на основі AES, розроблений для групових чатів [11].

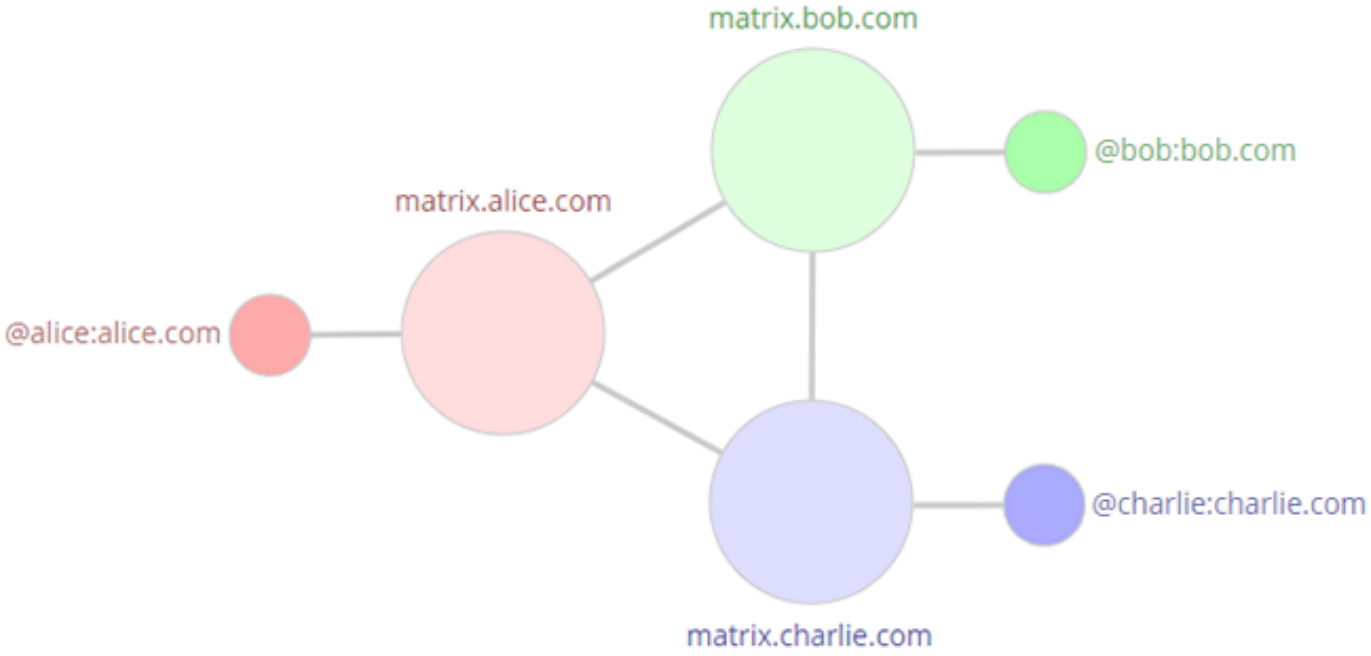

Рис.2.3. Архітектура протоколу Matrix

Хоча протокол Matrix надає користувачам ті ж властивості безпеки, що й Signal, він має деякі додаткові функції, які надає алгоритм megolm. З Matrix можна

використовувати декілька пристроїв, оскільки megolm реалізує окремий механізм для кожного пристрою-відправника, який бере участь у груповому чаті. Кожен груповий чат відслідковує додавання нових пристроїв.

NCC Group перевірила обидва алгоритми і виявила, що у megolm є деякі недоліки безпеки щодо подальшої та майбутньої секретності. Якщо зловмиснику вдається зламати ключ до сеансів Megolm, він може розшифрувати будь-які майбутні повідомлення, надіслані учасникам групової розмови. Matrix SDK, який використовується в програмах, які реалізували протокол Matrix, наприклад Riot; ключі Megolm оновлюються після певної кількості повідомлень, надісланих між учасниками.

Також частково забезпечується пряма секретність, оскільки megolm зберігає запис значення храповика, що дозволяє їм дешифрувати будь-які повідомлення, надіслані під час сеансу після відповідного пункту розмови [12].

2.5. Дослідження спеціалізованого протоколу MTProto

Telegram відомий як один з найпопулярніших сервісів обміну миттєвими повідомленнями (IM) для безпечного зв’язку. Він має наскрізне шифрування (E2EE) у секретних чатах на основі їх налаштованого протоколу під назвою MTProto. Цей абсолютно новий протокол вважається безпечною альтернативою. На основі налаштованого протоколу Telegram, який називається MTProto, він забезпечує шифрування клієнт-сервер у хмарних чатах для синхронізації всіх підключених пристроїв і E2EE у секретних чатах лише для двох пристроїв, які використовуються для ініціювання або прийняття секретних чатів [15].

2.5.1. Режими роботи IGE

Блокові шифри, такі як DES і AES, є важливими криптографічними активами і широко використовуються для масового шифрування даних. Оскільки їхні операції працюють лише з групою бітів фіксованої довжини, яка називається

блоком, може бути багато підходів до комбінування повторюваних операцій для кількох блоків (тобто режимів роботи). Як простий приклад, режим електронної кодової книги (ECB) ділить повідомлення на блоки і шифрує кожен з них незалежно, отже, ті самі блоки зашифрованого тексту генеруються з тих самих блоків відкритого тексту. Ця властивість робить систему дуже вразливою та незахищеною від аналізу трафіку. З кількох режимів роботи, більш безпечних, ніж режим ECB, режим Infinite Garble Extension (IGE) використовується в MTProto Telegram.

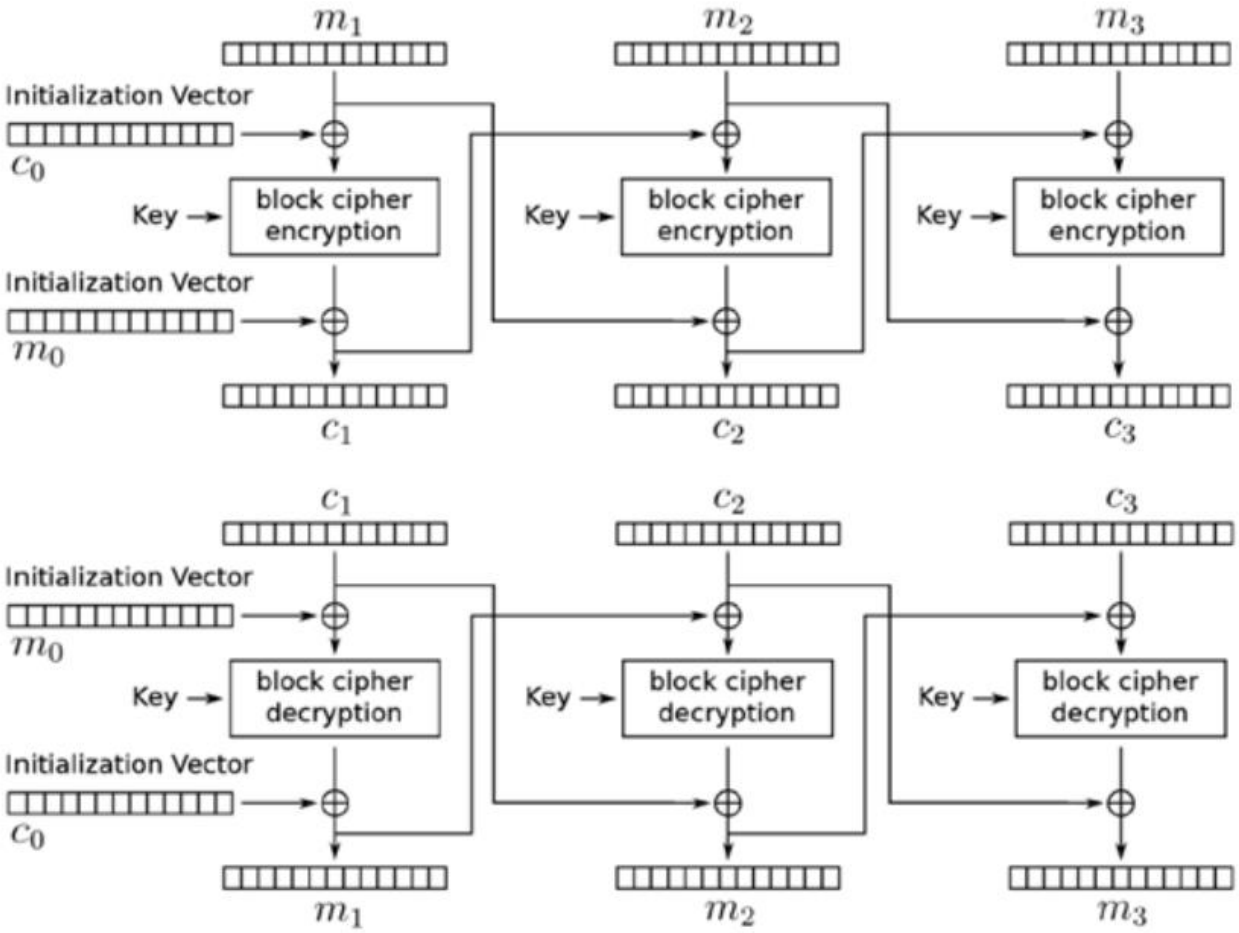

Рис.2.4. Шифрування та дешифрування в режимі IGE

Режим IGE має властивість що помилки поширюються вперед невизначено, тобто будь-яка різниця в зашифрованому тексті змінює (тобто спотворює) розшифровка всього наступного зашифрованого тексту.

Для дешифрування (рис.2.4) блоки з'єднані ланцюжком наступним чином:

$$c_i \leftarrow f_K(m_i \oplus c_{i-1}) \oplus m_{i-1}. \quad (2.1)$$

$$m_i \leftarrow {f_K}^{-1}(c_i \oplus m_{i-1}) \oplus c_{i-1}. \quad (2.2)$$

де f_K і ${f_K}^{-1}$ є функцією шифрування та дешифрування блочного шифру з ключем K відповідно. З цього рівняння перший вихідний блок c_1 потребує двох неіснуючих входів, c_0 і m_0. Вектор ініціалізації (IV) визначається за допомогою другого випадкового ключа K_0: $c_0 = f_{K_0}(m_0)$ або довільно заданих параметрів.

2.5.2. Аутентифіковане шифрування

Хоча попередні режими роботи забезпечували конфіденційність для блочних шифрів, були розроблені набагато кращі режими, які одночасно забезпечують конфіденційність, цілісність та автентичність, відомі як AE. Це поняття AE ввели, щоб гарантувати як конфіденційність повідомлення, так і цілісність відправника під час передачі по незахищеному каналу, як мобільна мережа.

Була запропонована загальна парадигма щодо безпечного шифрування та використання безпечного протокола MAC (AES і HMAC) - це стійкість до загрози розрізнення шифртексту відносно атаки на основі підібраного відкритого тексту (IND-CPA), стійкість до загрози відносно відкритого тексту (NM-CPA) або стійкість до загрози розрізнення шифртексту відносно атаки на основі підібраного шифрованого тексту (IND-CCA) для конфіденційності, а також цілісність відкритих текстів (INT-PTXT) і цілісність шифротекстів (INT-CTXT).

Припускається, що противнику A дозволено атаку з вибраним повідомленням, як показано нижче:

1. Визначення 1 (INT-PTXT). AE задовольняє умови безпеки INT-PTXT, якщо перевага будь-якого імовірнісного поліноміального зловмисника A для створення зашифрованого тексту $c = \varepsilon(m)$, де m раніше не було створено відправником, є незначним.

2. Визначення 2 (INT-CTXT). AE задовольняє умови безпеки INT-CTXT, якщо перевага будь-якого імовірнісного поліноміального зловмисника A для

створення зашифрованого тексту $c = \varepsilon(m)$, який раніше не був створений відправником, є незначною, незалежно від того, чи є основний відкритий текст m новим чи ні.

Виходячи з наведених вище вимог безпеки, було розроблено та проаналізовано три методи компонування для шифрування та протоколів MAC, а саме: Encrypt-and-MAC (E&M), MAC-then-Encrypt (MtE) і Encrypt-then-MAC (EtM).

1. Encrypt-and-MAC (E&M) - для шифрування з аутентифікацією шифрується відкритий текст m як $Enc(m)$, де Enc є шифруванням. Алгоритм безпечного протоколу шифрування та додавання тегу t із m за допомогою MAC, тобто зашифрованого тексту $\varepsilon(m) = Enc(m)||t$. Для дешифрування за допомогою верифікації перевіряється дійсність тегу, а також розшифровка шифротексту.

2. MAC-then-Encrypt (MtE) - для шифрування з аутентифікацією шифрується відкритий текст m як $Enc(m)$ і додається t тег $Enc(m)$ замість m, тобто шифрований текст $\varepsilon(m) = Enc(m)||t_{enc}$, де t_{enc}— тег $Enc(m)$. Для дешифрування за допомогою верифікації спочатку перевіряється тег, а потім розшифровується зашифрований текст.

3. Encrypt-then-MAC (EtM) - для шифрування з аутентифікацією додається тег t із m спочатку, потім шифрується доданий простий-текст $m||t$, тобто зашифрований текст $\varepsilon(m) = Enc(m||t)$. Для дешифрування за допомогою верифікації спочатку розшифрується зашифрований текст, щоб отримати відкритий текст і тег.

Результати безпеки для композитних схем AE показують, що EtM може досягти лише найвищого визначення безпеки в AE (табл.2.1).

Таблиця 2.1.

Результати безпеки для композитних схем AE

	IND-CPA	IND -CCA	NM-CPA	INT-PTXT	INT-CTXT
E&M	небезпечно	небезпечно	небезпечно	безпечно	небезпечно
MtE	безпечно	небезпечно	небезпечно	безпечно	небезпечно
EtM	безпечно	безпечно	безпечно	безпечно	безпечно

2.5.3. Використання протоколу MTProto в таємних (секретних) чатах

MTProto має два різних протоколи шифрування для хмарних і таємних (секретних) чатів. Хмарні чати використовують шифрування клієнт-сервер для синхронізації всіх підключених пристроїв, а таємні (секретні) чати використовують E2EE лише для двох пристроїв, які використовуються щоб почати або прийняти секретний чат [14].

MTProto використовує обмін ключами Diffie-Hellman (DH), алгоритм безпечного хешування 1 (SHA-1), функцію отримання ключа (KDF) і AES-256 в режимі IGE як криптографічні примітиви, а загальний процес описаний на рис.2.5.

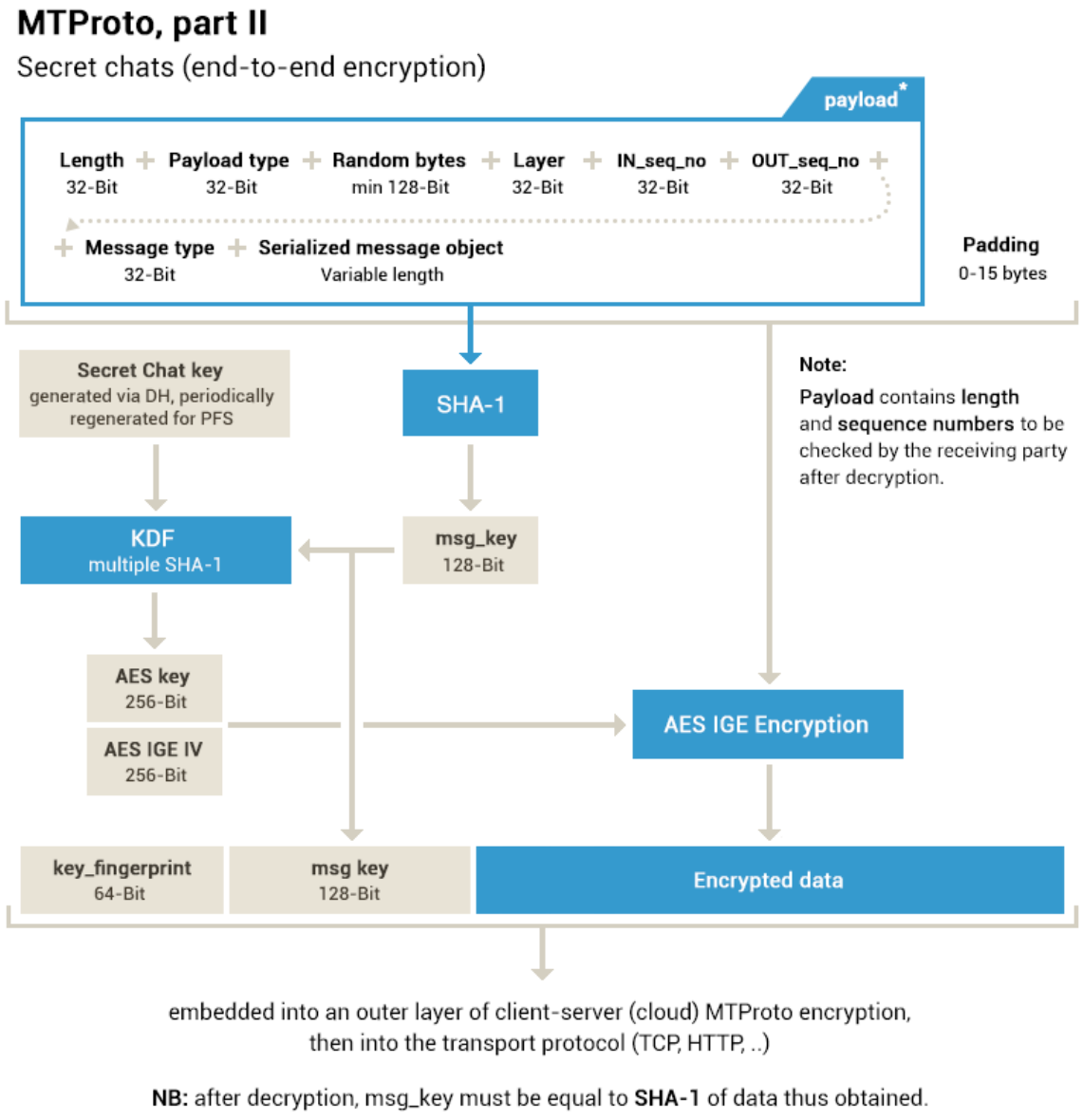

Рис.2.5. MTProto в E2EE

Генерація ключів. Обмін ключами DH використовується для створення коротривалого ключа. Після обміну ключами відправник і одержувач користуються одним і тим же 2048-бітовим симетричним ключем K. Щоб захистити попередні сеанси, секретний ключ генерується повторно, коли ключ використовувався для понад 100 повідомлень або більше.

Корисне навантаження x генерується шляхом конкатенації деякої допоміжної інформації, випадкових байтів, повідомлення та заповнення таким чином, що $|x|\ mod\ B\ =\ 0$, де B — довжина блоку. Потім корисне навантаження, крім заповнення, обчислюється за допомогою хеш-функції SHA-1, вихід якої називається тегом. Цей tag знову хешується KDF для створення ключа AES та IV. Вхід KDF є $(K; tag)$, а вихідний – (k, c_0, m_0) довжини (k, B, B)де $k = 256$ біт і $B\ =$ 128 біт.

Шифрування. Для шифрування використовується AES-256 в режимі IGE.

Нехай $x_1 \ldots . x_l$ — це l блоків корисного навантаження, кожен із довжиною B, тоді зашифрований текст обчислюється, як показано нижче:

$$c_i \leftarrow F_k(m_i \oplus c_{i-1}) \oplus m_{i-1}, \tag{2.3}$$

де F — псевдовипадкова перестановка, наприклад, AES.

Кінцевий результат шифрування c, включаючи іншу інформацію.

$$c = (tag, c_1, \ldots c_l). \tag{2.4}$$

Розшифрування. Враховуючи зашифрований текст c, tag знову використовується в KDF. Використовуючи (tag, K), вихід KDF (k, c_0, m_0) такий самий, як і шифрування. Крім того, режим IGE знову використовується для дешифрування, і корисне навантаження x відновлюється.

$$m_i \leftarrow {F_k}^{-1}(c_i \oplus m_{i-1}) \oplus c_{i-1}. \tag{2.5}$$

Корисне навантаження, за винятком заповнення, обчислюється за допомогою хеш-функції і перевіряє, чи є результат таким самим, як і тег у зашифрованому тексті c. Якщо так, можна перевірити, що повідомлення в корисному навантаженні є оригінальним відкритим текстом.

Відомі атаки на MTProto. Вчені теоретично продемонстрував, що MT-Proto не відповідає IND-CCA та INT-CTXT у 2017 році. На основі вразливості випадкового заповнення, що MTProto не перевіряє ні довжину, ні вміст заповнення під час розшифрування AES-256 в режимі IGE, були спробовані дві атаки: розширення довжини заповнення та заміна останнього блоку.

Атака 1. Розширення довжини заповнення. З визначення 2.3 створили новий зашифрований текст, щоб зламати безпеку INT-CTXT MTProto. Щоб зрозуміти безпеку INT-CTXT, необхідно узагальнити чому MTProto не захищений IND-CCA, використовуючи наступне: для імовірнісного зловмисника з поліноміальним часом A завжди виграє наступну гру, тобто MT-Proto не є безпечним. IND-CCA безпечний при:

1. A виводить різні повідомлення M_0 і M_1 однакової довжини.

2. Претендент C вибирає $b \in \{0; 1\}$ випадковим чином і виводить зашифрований текст $C_b \leftarrow \varepsilon(M_b)$.

3. A додає 128-бітовий випадковий блок c_r до C_b і просить C розшифрувати $C' = c_r || C_b$

4. C повертає M', де $M' = M_b$ для будь-якого b.

5. A вгадує b як 0, якщо $M' = M_0$, 1 інакше.

Ця атака можлива, оскільки додаткове заповнення зашифрованого тексту дає лише розширення заповнення відкритого тексту без зміни відкритого тексту. Очевидно, A отримує зашифрований текст $C' = \varepsilon(M_b)$ для M_b і цей зашифрований текст раніше не був створений відправником через випадкове заповнення.

Як наслідок можна стверджувати, що протокол MTProto не є захищеним INT-CTXT. Щоб захиститися від цієї атаки, необхідно перевірити довжину заповнення в M', змінивши параметр процесу розшифровки. Алгоритм дешифрування відкине повідомлення, якщо довжина заповнення перевищує розмір блоку [15].

Атака 2. Заміна останнього блоку. Оскільки заповнення не перевіряється в процесі MTProto, можна здійснити колізію шляхом зміни останніх 128-бітових (16-байтових) блоків.

Для імовірнісного зловмисника A з поліноміальним часом, A виграє наступний шаблон з ймовірністю не більше 2^{-8}, тобто MTProto не є безпечним INT-CTXT при:

1. A виводить повідомлення M, довжина якого в байтах дорівнює $b \ mod \ 16$.

2. Заявник C хешує M у ключ повідомлення $msg\_key \leftarrow SHA-1(M)$ для забезпечення цілісності відкритого тексту.

3. Перед шифруванням до M додається $16 - b$ випадкових байтів заповнення, а потім надсилається $C = \varepsilon(M||r)$.

4. А змінює останні 16-байтні блоки C, щоб отримати $C' \neq C$.

5. А виводить C'.

З наведеного вище шаблону C розшифровує C'як $M'||r'$ Тоді лише останній байт M' відрізняється від M через непіддатливість режиму IGE. Таким чином, вони стверджують, що можна мати $M' = M$ з імовірністю не більше ніж 2^{-8}, коли A вибирає повідомлення M з довжиною в байтах, що дорівнює $1 \ mod \ 16$, тобто вони можуть генерувати дійсний зашифрований текст $C' \neq C$ з імовірністю не більше 2^{-8}.

Щоб зробити протокол захищеним від цієї атаки, достатньо додати доповнення до обчислення тегу аутентифікації. Але оскільки це вимагає зміни всього шифрування в процесі аутентифікації, стає неможливим зв’язок зі старішими версіями протоколу через сумісність версій.

2.5.4. Реалізація протоколу MTProto в таємних (секретних) чатах

Для експериментальної демонстрації вразливостей випадкового заповнення, спрощується MT-Proto, зберігаючи основні компоненти, обмін ключами DH, SHA-1 корисного навантаження, налаштований KDF та зашифрований зв’язок через

мережу з AES- 256 в режимі IGE. У порівнянні з оригінальним протоколом є дві незначущі відмінності:

1) використовуються прості числа та генератори RFC-3526 в обміні ключами DH;

2) необхідно розглядати корисне навантаження як лише довжину та зміст повідомлення. Вони тривіальні та незначні під час тестування вразливостей випадкового заповнення.

MTProto було розроблено в macOS Sierra та Ubuntu 16.10 з використанням Python 3.5.2. Він розроблений для демонстрації комунікації типових користувачів (наприклад, Аліси та Боба) через мережу Інтернет. При проведені експерименту, комунікація була змодельована в приватній мережі між хостом і віртуальною машиною.

1. *Генерація ключів.* На рис. 2.6. описано процедуру генерації та обміну ключами. Боб діє як сервер і прослуховує вхідні з'єднання з мережі.

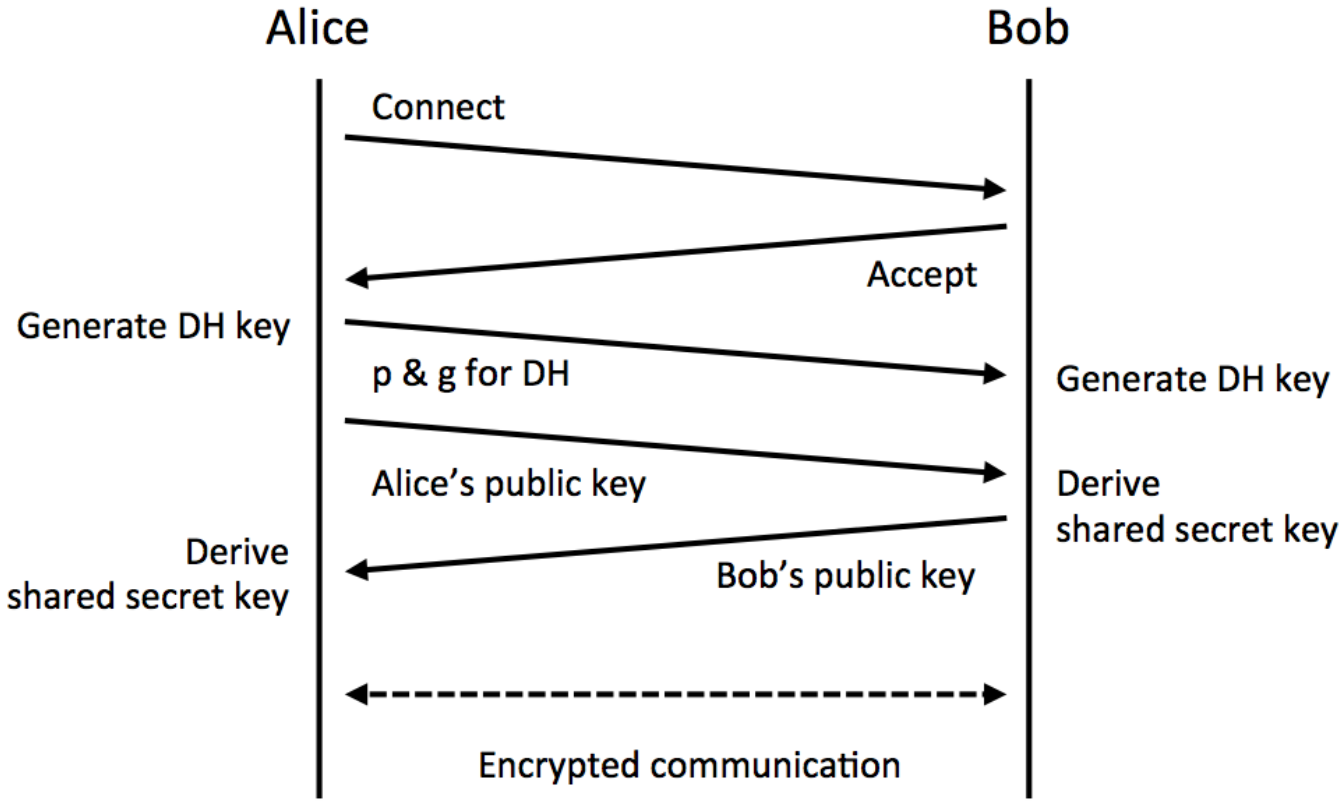

Рис.2.6. Процедура генерації та обміну ключів

Коли Аліса хоче поговорити з Бобом, вона ініціює з’єднання протоколу керування передачею (TCP) до Боба і надсилає свій відкритий ключ DH. Тоді Боб приймає підключення Аліси і отримує її відкритий ключ. Боб також створює свій власний ключ DH і надсилає відкритий ключ назад Алісі [16].

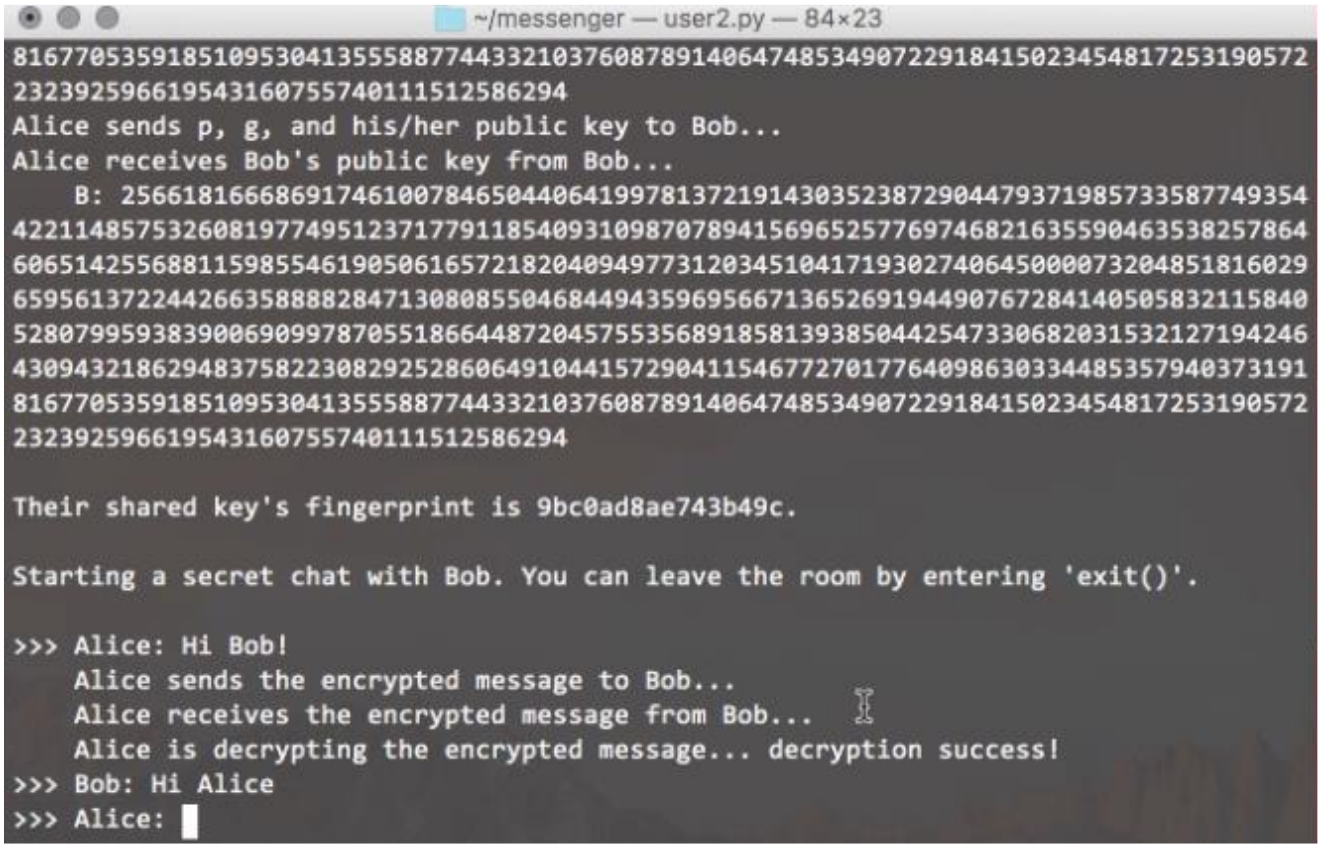

Рис.2.7. Скріншот спрощеного виконання MTProto (1 частина)

Оскільки Аліса і Боб знають відкритий ключ один одного, тепер вони можуть отримати спільний секретний ключ і почати зашифроване спілкування. Після того, як вони почали секретний чат, один надсилає повідомлення та генерує ключ повідомлення з SHA-1 корисного навантаження. Ключ AES і IV для AES-256 в режимі IGE отримані KDF за допомогою ключа DH і ключа повідомлення, а псевдокод KDF показаний в Алгоритмі 1, де символ + позначає конкатенацію рядків, msg_key є останніми 16 байтами SHA. -1 корисного навантаження і dh_key є загальним секретним ключем між Алісою та Бобом. Цей ключ шифрування змінює повідомлення за повідомленням у MTProto.

Алгоритм 1: Псевдокод KDF

def kdf (dh_key, msg_key):

a = sha1 (msg_key + dh_key [0:32])

b = sha1 (dh_key [32:48] + msg_key + dh_key [48:64])

c = sha1 (dh_key [64:96] + msg_key)

d = sha1 (msg_key + dh_key [96:128])

aes_key = a [0:8] + b [8:20] + c [4:16]

aes_iv = a [8:20] + b [0:8] + c [16:20] + d [0:8]

return (aes_key, aes_iv)

2. Шифрування та дешифрування повідомлень. Коли Аліса або Боб пишуть повідомлення, MTProto спочатку додає деяку інформацію про повідомлення і викликає його як корисне навантаження. Для простоти додається до корисного навантаження лише 32-бітове поле довжини повідомлення. Потім доповнене корисне навантаження шифрується за допомогою AES-256 в режимі IGE.

Отримавши ключ і IV, додаються кілька байтів випадкового заповнення до кінця корисного навантаження, щоб вирівняти його з 16-байтовими блоками. Для шифрування та дешифрування використовується *aes_ige_enc()* і *aes_ige_dec()*, як показано в Алгоритмі 2.

Алгоритм 2: Псевдокод AES-256 в режимі IGE

def aes_ige_enc (key, iv, M):

aes = AES. new (key) # ECB mode

c_prev = iv [0:16]

m_prev = iv [16:32]

C = bytes ()

for i in range (0, len (M), 16):

m = M[i:i +16]

aes_ecb_enc_in = xor (m, c_prev)

aes_ecb_enc_out = aes . encrypt (aes_ecb_enc_in)

c = xor (aes_ecb_enc_out, m_prev)

m_prev = m

```
c_prev = c
C += c
return C

def aes_ige_dec (key, iv, C):
aes = AES . new (key) # ECB mode
c_prev = iv [0:16]
m_prev = iv [16:32]
M = bytes ()
for i in range (0, len(C), 16):
c = C[i:i +16]
aes_ecb_dec_in = xor (c, m_prev)
aes_ecb_dec_out = aes . decrypt (aes_ecb_dec_in)
m = xor (aes_ecb_dec_out, c_prev)
m_prev = m
c_prev = c
M += m
return M
```

Оскільки AES в режимі IGE на практиці використовується мало, він навіть не включений до популярних криптографічних бібліотек. Режим IGE реалізовується із маніпулюванням введенням і виводом AES в режимі ECB за допомогою бібліотеки PyCrypto 2.6.1 на Python. key і iv — це значення, отримані від $kdf()$, а M — вирівняне корисне навантаження з заповненням [18].

Після того як *aes_ige_enc()* повертає зашифровані байти доповненого корисного навантаження, дані, надіслані через мережу, включають відбитки для ключа DH, *msg_key* та зашифроване корисне навантаження. Коли Боб або Аліса отримують дані, повідомлення розшифровується в зворотному порядку. Спочатку перевіряється відбиток пальця, а ключ і IV для AES отримують KDF за допомогою ключа *msg* і спільного секретного ключа. Потім, нарешті, повідомлення розшифровується *aes_ige_dec()*.

Знімок екрана спрощеного виконання MTProto показано на рис.2.8. Дану реалізацію можна також розширити, щоб мати виділений сервер для передачі повідомлень між користувачами, як це роблять хмарні сервери Telegram. У цьому випадку можна продемонструвати зловмисника на сервері, який може бачити всі обміни між Алісою та Бобом. Поки спілкування зашифровано, можна перевірити відомі атаки або знайти новий експлойт, використовуючи реалізацію.

Рис.2.8. Скріншот спрощеного виконання MTProto (2 частина)

Як приклад побудови будівельних блоків для E2EE в системі обміну повідомленнями, пропонується типова конструкція, наведена в таблиці 2.2.

Цими чотирма базовими складовими є математичний рівень, криптографічний примітивний рівень, рівень керування ключами та рівень служби безпеки.

Математичний містить базову арифметику для криптографічних примітивів, криптографічний примітивний рівень виконує популярні криптографічні примітиви для безпечного E2EE. Рівень керування ключами виконує генерацію, розповсюдження та узгодження ключів, включаючи будь-які операції, пов’язані з ключами. Нарешті, рівень служби безпеки надає необхідні функції безпеки для певних миттєвих повідомлень і буде використовуватися для боротьби з

додатковими атаками або легкого вимірювання криптографічної міцності при збереженні основного дизайну криптографічних протоколів.

Таблиця 2.2.

Базові складові для E2EE в IM

Рівень служби безпеки	Аутентифіковане шифрування Вперед таємність Зворотна сумісність MAC, HMAC Багаторазове шифрування Режими роботи та ін.
Рівень керування ключами	Генерація ключів Розповсюдження ключів Ключова угода KDF та ін.
Криптографічний примітивний рівень	Симетричний: AES Асиметричні: RSA, ElGamal, ECC Хеш: SHA-1, SHA-3 тощо.
Математичний рівень	Арифметика з множинною точністю Арифметика кінцевого поля Модульне піднесення до степеня Арифметика еліптичної кривої тощо.

Висновки до другого розділу

Досліджено три протоколи для безпечного обміну повідомленнями, які забезпечують наскрізне шифрування повідомлень. Зазначено, що протоколи безпечного обміну повідомленнями не забезпечують кожну властивість безпеки, а це означає, що є можливість покращення для всіх протоколів. Дослідники безпеки повинні працювати разом над протоколами, щоб придумати способи реалізації решти властивостей, які не досягнуті.

Підкреслено, що незважаючи на те, що Telegram вважається одним із найбільш безпечних сервісів обміну повідомленнями, їхній власний протокол MTProto не був повністю розглянутий експертами з криптоаналітики. У дизайні криптографічного протоколу є багато сумнівних варіантів, наприклад, колізії SHA-1, налаштований KDF, нестандартний алгоритм заповнення та режим IGE, який не забезпечує автентичності.

Проаналізовано вразливості безпеки E2EE для Telegram і запропоновано типові базові складовідля E2EE в загальній системі обміну повідомленнями.

3 ТЕХНОЛОГІЇ ЗАХИСТУ ІНФОРМАЦІЇ В TELEGRAM ВІД СПАМУ

Telegram – це сервіс, за допомогою якого здійснюється обмін миттєвими повідомленнями, однак він дозволяє користувачам приймати участь в різних групах, надсилати повідомлення, фотографії, відео та інші. Однак і в даному сервісі наявні проблеми, що створюють передумови для зменшення безпеки для користувачів.

3.1. Спам та фішинг в сервісі Telegram

Одним із доступних та ефективних, згідно зі статистикою, вважається спам у Telegram. Спам — масове розсилання інформації рекламного чи будь-якого іншого характеру користувачам. Поширення рекламної інформації – необхідна умова для популяризації та просування товарів та послуг у маси. Коштів та способів існує безліч [17].

Поряд з іншими сервісами Telegram має свою аудиторію. Кількість користувачів даної програми безперервно поповнюється новими учасниками. Маркетологи визнали сервіс, як хороший засіб для збільшення обсягів продажів. Не дивлячись на невдоволення користувачів, згідно зі статистикою, з 10 тис. відправлених листів 50 призводять до укладання угоди.

Ще одним критичним різновидом загроз в сучасному сервісі Telegram є фішинг. Це вид інтернет-шахрайства, метою якого є отримання доступу до конфіденційних даних користувачів - логінів і паролів. Досягається шляхом проведення масових розсилок електронних листів від імені популярних брендів, а також особистих повідомлень всередині сервісу, наприклад, від імені банків або соціальних мереж. Приклад фішингового сайту представлено на рис.3.1.

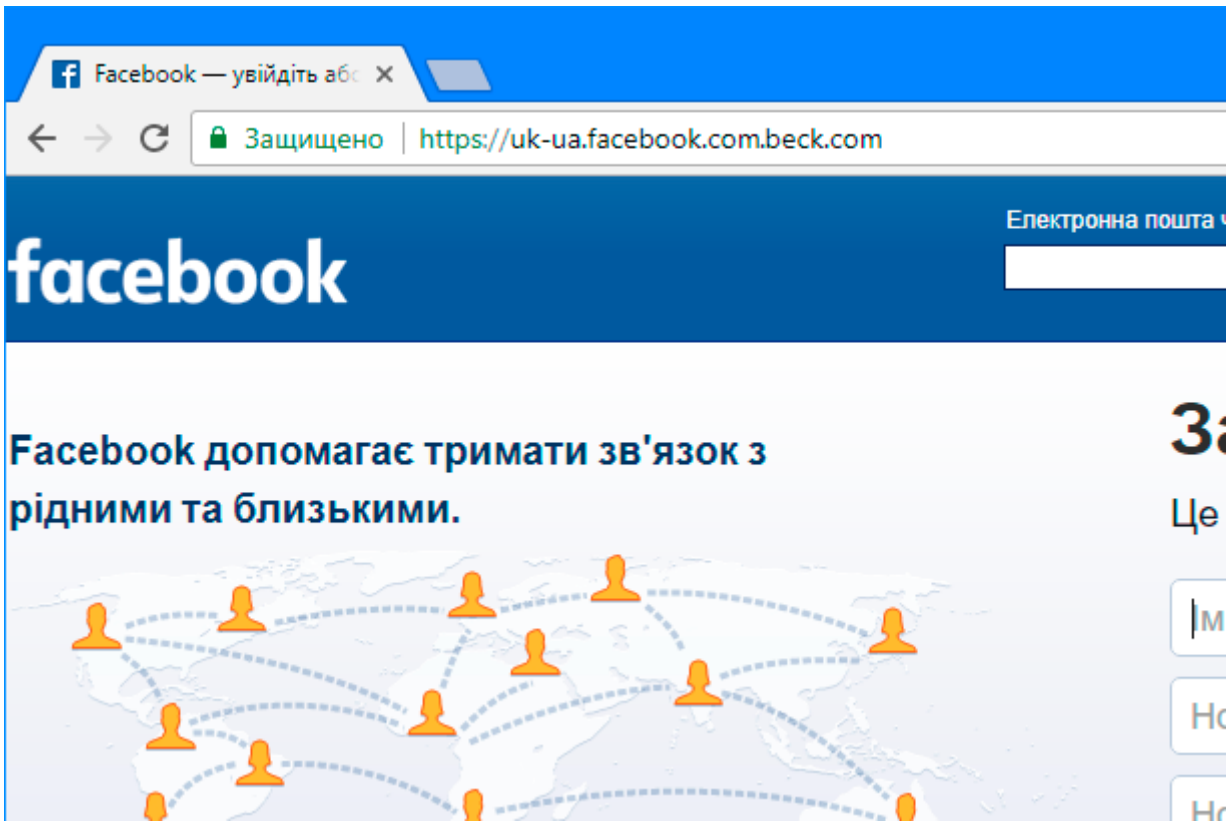

Рис.3.1. Фішінгове посилання

Вкрадені в результаті фішингових атак дані користувачів все частіше вивантажуються не тільки за допомогою електронної пошти, але й таких легітимних сервісів, як Google-форми та месенджер Telegram.

Telegram-боти також застосовуються кіберзлочинцями в готових платформах для автоматизації фішингу, доступних у даркнеті: в них на основі ботів реалізована адміністративна частина, за допомогою якої контролюється весь процес атаки фішинга і ведеться облік викрадених грошей. Такі платформи поширюються за форматом cybercrime-as-a-service, за рахунок чого зростає кількість атакуючих угруповань та масштаб злочинного бізнесу.

Команда CERT-GIB (Центр моніторингу та реагування на інциденти інформаційної безпеки Group-IB) проаналізувала інструменти для створення фішингових сторінок, так звані, фішинг-кити, та з'ясувала, що у 2020-му році за їх допомогою фішингові сайти найчастіше створювалися під різні онлайн-сервіси (онлайн-шопінг, онлайн-кінотеатри та ін.), електронну пошту, а також фінансові

організації. Сумарно Group-IB виявила фішинг-кити, націлені понад 260 унікальних брендів по всьому світі [19].

Фішинг-кит - це набір готових інструментів для створення та запуску фішингових веб-сторінок, підроблених під сайт конкретної компанії або кількох відразу. Як правило, фішинг-кити продаються в даркнеті на спеціалізованих форумах. З їх допомогою злочинці, які не мають глибоких навичок програмування, можуть легко розгортати інфраструктуру для масштабних фішингових атак і швидко відновлювати її роботу у разі блокування. Дослідникам з кібербезпеки фішинг-кити цікаві насамперед тим, що аналіз одного такого «набору» дозволяє розібратися в механізмі реалізації фішингової атаки та встановити, куди вирушають викрадені дані. Крім цього, дослідження фішинг-китів часто допомагає виявити цифрові сліди, які ведуть розробників такого «товару».

Дані, які користувач вводить на фішинговому сайті, зловмисник отримує не відразу: спочатку вони записуються в локальний файл, після чого головним завданням стає вилучення викраденого. Найчастіше для надсилання таких даних використовуються поштові адреси, зареєстровані на безкоштовних сервісах електронної пошти. Вони становлять 66% загальної кількості адрес, знайдених у фішингових наборах. Найбільш поширені акаунти на Gmail та Yandex.

Функціональність фішинг-китів не обмежується створенням сторінок для викрадення даних користувачів: деякі можуть підвантажувати шкідливі файли на пристрій жертви. Іноді продавці наборів для фішингу обманюють своїх покупців, намагаючись заробити на них двічі. Крім продажу створеного ними шкідливого інструменту вони можуть зацікавитися і викраденими з його допомогою даними. Використовуючи спеціальний скрипт, вбудований у тіло фіш-кита, вони направляють потік вкрадених даних користувача або отримують прихований доступ до хостингу свого покупця.

Фішинг-кити змінили правила гри в сегменті боротьби з кіберзлочинами: раніше зловмисники припиняли свої кампанії після блокування шахрайських ресурсів і швидко переключалися на інші бренди, сьогодні вони автоматизують атаку, моментально виводячи нові фішингові сторінки на зміну заблокованим.

Автоматизація таких атак, у свою чергу, призводить до поширення складнішої соціальної інженерії, яка починає застосовуватися в масштабних атаках, а не в точкових, як було раніше. Це дозволяє представникам однієї з найдавніших кіберзлочинних професій залишатися на плаву.

Існує багато різних технічних способів захисту від фішингу, але «слабка ланка» в цьому ланцюзі - сам користувач, його цікавість і неуважність дозволяють шахраям ефективно застосовувати методи соціальної інженерії.

Зазвичай портали роблять розсилку про спробу злому акаунтів і про те, що користувачі можуть отримати лист з шахрайською посиланням. З цього випливає що джерелами спаму у меседжерах є приватні, публічні групові чати, а також особисті чати.

Спам з'являється за наступнии сценарієм: учасник групи додає спам-бота в групу, учасник додає спамера в групу, користувач власноруч запускає спам-бота, спамер пише особисто.

Майже всі користувачі меседжерів, колись в своєму житті зіштовхувались зі спамом.

3.2. Метод протидії спаму в сервісі Telegram

Телеграм-спам – це термін, що відноситься до небажаної інформації, що містить посилання, рекламу або текст, що забиває обліковий запис, канал або групу. Поширюється через клієнти миттєвого обміну повідомленнями.

Завдяки спаму користувачі отримують прибуток від продажу товару за посиланням, інформують передплатників про новий контент і т. д. У особистих або робочих цілях повідомляється вузьке коло людей[21].

Принцип дії. За час існування месенджера був розроблений та вдосконалений алгоритм розповсюдження реклами (ручний, автоматизований). Формуються правила гарного тону для спамерів і способи обходу блокувань.

Рис.3.2. Метод блокування спаму в сервісі Telegram

Для досягнення мети вивчається функціональність месенджера, що дозволяє використовувати його недоліки та переваги на свою користь. Формується певна послідовність дій.

Відправка в приватні повідомлення. Знайомий контакт також може отримати статус спамера. Часта, незначна інформація, повідомлення про різноманітні розпродажі тощо – все це викликає негативну реакцію користувачів.

Чат і групова розсилка. Такі повідомлення відправляються за допомогою програм. До поширених можна віднести Твігі. Використання цього продукту можливо з базою даних номерів. Дозволяється збирати самостійно. Для цих цілей знадобиться вручну шукати групи в соціальних мережах і зберігати номери, що залишилися на сторінках, або використовувати парсери. Розсилка новин

здійснюється за допомогою смартфона з дійсною SIM-картою. Реалізувати це можна через SMS-активацію, де встановлено Telegram. У мережі також поширені всілякі емулятори, наприклад, для Android – це NOX.

Спам-боти. Вивчаючи представлені сервіси, завжди можна знайти спосіб обійти обмеження і досягти бажаного результату: учасник ненавмисно додає бота до групи, антиспам-бот визначає його появу. Метод Telegram API покликаний накладати обмеження на права писати повідомлення в групу або канал. До того, як заборона набуде чинності, спам-бот встигає розповсюдити чат [22].

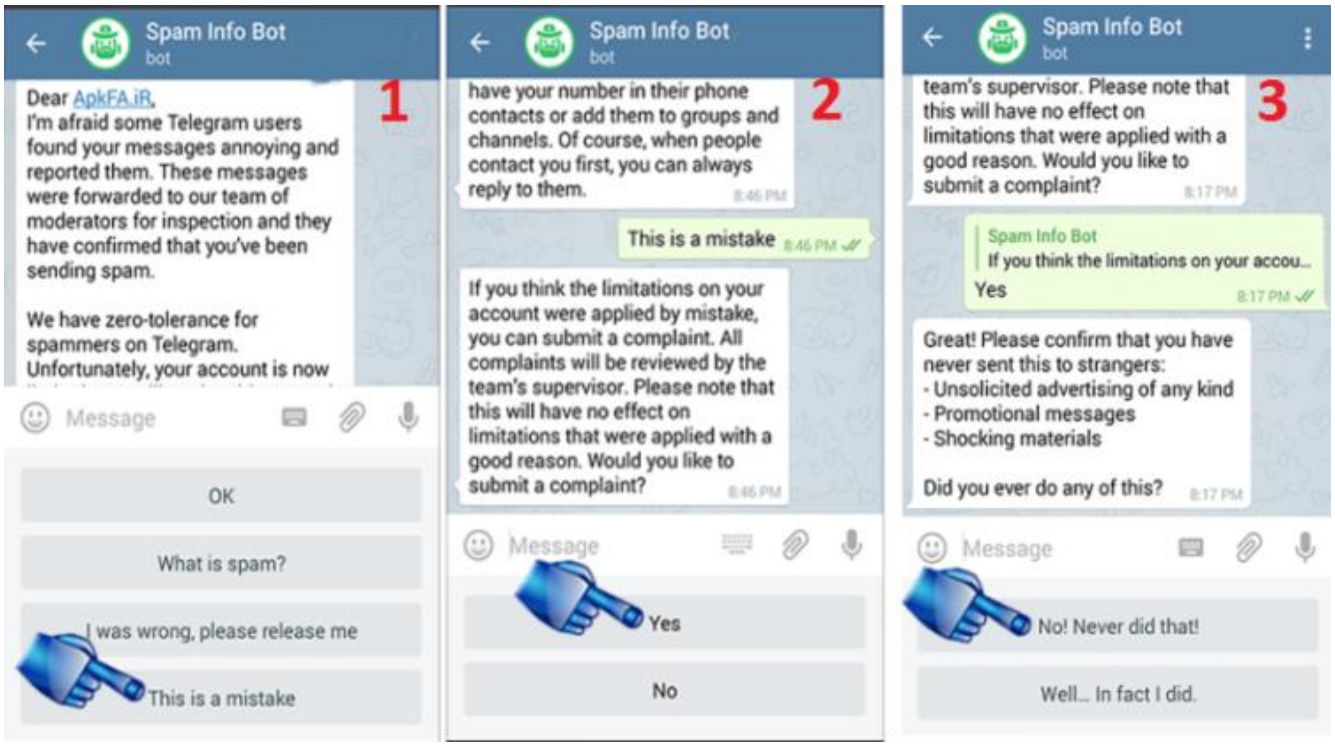

Рис.3.3. Приклад блокування спаму в сервісі Telegram за допомогою боту

Розглянутий приклад показав, що спамерські атаки легко реалізуються через користувачів, а завдяки функціоналу Telegram можна досягти поставлених цілей за короткий проміжок часу. Але слід бути обережним. Нерідко трапляється, що спам-бот Telegram надсилає віруси та шкідливе програмне забезпечення.

Кожен користувач прагне використовувати месенджер тільки для спілкування з близькими людьми. Тому розробники пропонують використовувати

@SpamKillerRobot, який відповідає за видалення небажаної інформації з групових чатів.

Процес очищення включає в себе наступні кроки:

1. Додається бот в чат Telegram, де з’явився спамер;
2. В групі, необхідно перейти до налаштування;
3. Обрати адміністратора;
4. Ввести Spam killer Robot в рядок пошуку, та додати його до групи.

Видалення небажаних повідомлень у групі здійснюється на основі відповідей користувачів на такі повідомлення зі словом «спам». Очистка чату відбувається після надходження скарг як мінімум від 3 осіб.

Робот-бот Spam_killer – помічник в адмініструванні великих груп з активним листуванням.

Існує ряд рекомендацій, які також допомагають зменшити отримання непотрібних повідомлень:

1. Оскільки завданням спамера є формування бази контактів, підозрілі повідомлення слід залишати без відповіді. В іншому випадку велика ймовірність потрапити в базу розсилки;
2. Не переходити за посиланнями, надісланими незнайомцями. Ці дії не тільки свідчать про те, що користувач являється активним контактом, але й дають можливість заразити пристрій шкідливим програмним забезпеченням;
3. Потрібно надсилати скарги на дії користувачів адміністраторам групи.

Виконання цих кроків дозволить користувачеві приховати контакт від ботів і уникнути отримання рекламних розсилок.

Скарга на підтримку. У випадку, якщо обліковий запис знаходиться в базі спамерів, або користувач став їх жертвою, розробники месенджера пропонують скористатися FAQ або допомогою волонтерів[14].

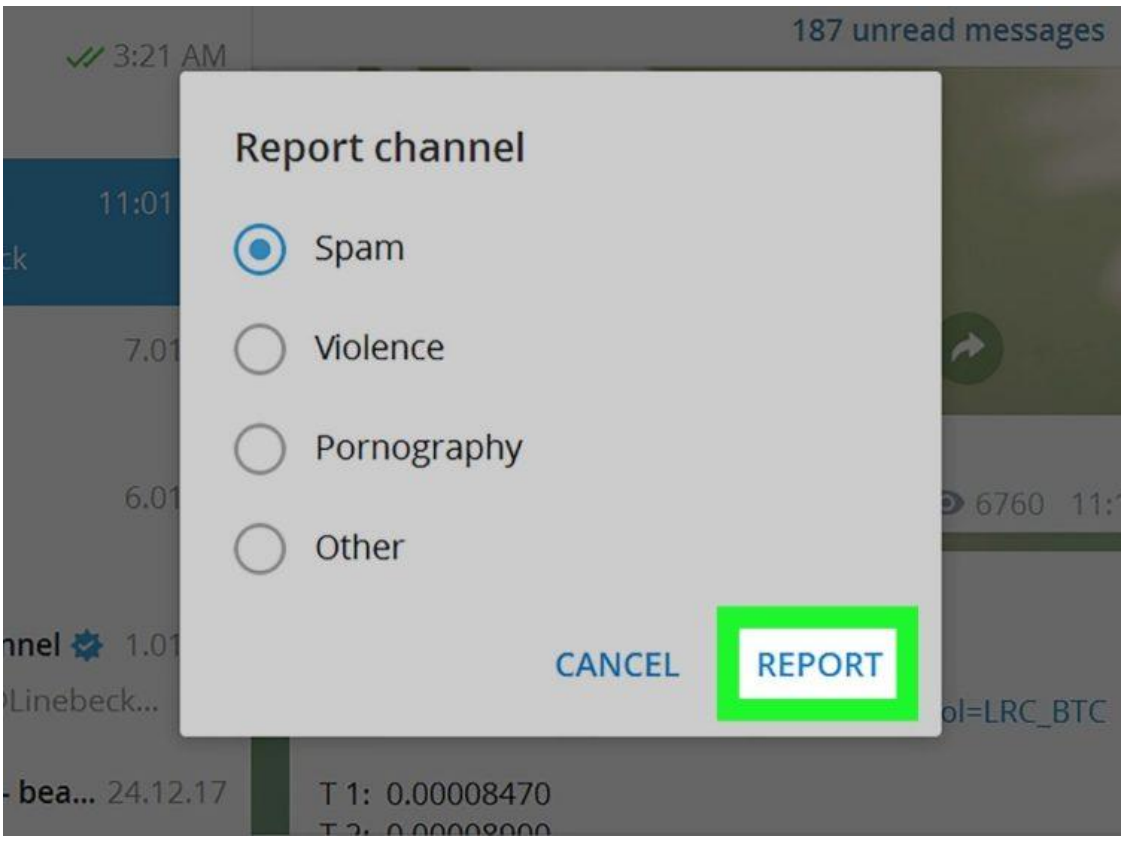

Рис.3.4. Приклад звернення до волонтерів сервісу Telegram для блокування спамерів

Існує кілька способів обмежити права користувачів і заборонити їм надсилати повідомлення групі чи конкретній особі:

- відкрити заблокований контакт;
- перейти до налаштувань;
- вибрати «заблокувати» і натиснути «ОК».

Якщо мова йде про групу (канал) Telegram, то адміністратори та учасники мають можливість блокування.

Практика показала, що причиною отримання спаму не завжди є добре спланована атака. Іноді самі учасники групи або передплатники каналу ненавмисно додають бота, відкриваючи таким чином простір для активності.

Скрипт програми, як захистити бота Telegram від спаму, представлений далі. Якщо користувач надсилає багато запитів за короткий час, скрипт виходить з ладу

і видає помилку - запит до Telegram API був невдалим. Сервер повернув HTTP 429 Забагато запитів[26]. Тіло відповіді:

```
import
asyncio


        from aiogram import Bot, Dispatcher, executor, types
        from aiogram.contrib.fsm_storage.redis import RedisStorage2
        from aiogram.dispatcher import DEFAULT_RATE_LIMIT
        from aiogram.dispatcher.handler import CancelHandler, current_handler
        from aiogram.dispatcher.middlewares import BaseMiddleware
        from aiogram.utils.exceptions import Throttled

        TOKEN = 'BOT_TOKEN_HERE'

        # In this example Redis storage is used
        storage = RedisStorage2(db=5)

        bot = Bot(token=TOKEN)
        dp = Dispatcher(bot, storage=storage)

        def rate_limit(limit: int, key=None):
            """
            Decorator for configuring rate limit and key in different functions.

            :param limit:
            :param key:
            :return:
            """

            def decorator(func):
                setattr(func, 'throttling_rate_limit', limit)
                if key:
                    setattr(func, 'throttling_key', key)
                return func

            return decorator

        class ThrottlingMiddleware(BaseMiddleware):
            """
            Simple middleware

            def __init__(self, limit=DEFAULT_RATE_LIMIT,
        key_prefix='antiflood_'):
```

```
        self.rate_limit = limit
        self.prefix = key_prefix
        super(ThrottlingMiddleware, self).__init__()

    async def on_process_message(self, message: types.Message, data:
dict):
        """
        This handler is called when dispatcher receives a message

        :param message:
        """
        # Get current handler
        handler = current_handler.get()

        # Get dispatcher from context
        dispatcher = Dispatcher.get_current()
        # If handler was configured, get rate limit and key from handler
        if handler:
            limit = getattr(handler, 'throttling_rate_limit', self.rate_limit)
            key = getattr(handler, 'throttling_key',
f"{self.prefix}_{handler.__name__}")
        else:
            limit = self.rate_limit
            key = f"{self.prefix}_message"

        # Use Dispatcher.throttle method.
        try:
            await dispatcher.throttle(key, rate=limit)
        except Throttled as t:
            # Execute action
            await self.message_throttled(message, t)

            # Cancel current handler
            raise CancelHandler()

    async def message_throttled(self, message: types.Message, throttled:
Throttled):
        """
        Notify user only on first exceed and notify about unlocking only on
last exceed

        :param message:
        :param throttled:
        """
        handler = current_handler.get()
```

```
        dispatcher = Dispatcher.get_current()
        if handler:
            key = getattr(handler, 'throttling_key',
f"{self.prefix}_{handler.__name__}")
        else:
            key = f"{self.prefix}_message"

        # Calculate how many time is left till the block ends
        delta = throttled.rate - throttled.delta

        # Prevent flooding
        if throttled.exceeded_count <= 2:
            await message.reply('Too many requests! ')

        # Sleep.
        await asyncio.sleep(delta)

        # Check lock status
        thr = await dispatcher.check_key(key)

        # If current message is not last with current key - do not send message
        if thr.exceeded_count == throttled.exceeded_count:
            await message.reply('Unlocked.')

@dp.message_handler(commands=['start'])
@rate_limit(5, 'start')  # this is not required but you can configure
throttling manager for current handler using it
async def cmd_test(message: types.Message):
    # You can use this command every 5 seconds
    await message.reply('Test passed! You can use this command every 5
seconds.')

if __name__ == '__main__':
    # Setup middleware
    dp.middleware.setup(ThrottlingMiddleware())

    # Start long-polling
    executor.start_polling(dp)
```

Приклад реалізації проти затоплення через декоратор можна побачити на аіограмі. Також така помилка трапляється, коли швидко видаляють-встановлюють webhook.

3.3. Дослідження актуальних конструкторів ботів у сервісі Telegram

Ще одним методом протидії спаму та фішингу, є розробка власного бота, який буде не лише протидіяти рекламній розсилці, але і виокремлювати аномальну поведінку серед підписників та користувачів у групових чатах. В свою чергу, це допоможе уникнути ситуцій, коли користувачі власноруч передаватимуть зловмисникам власні персональні та конфіденційні дані, переходячи за недостовріними посиланнями.

Конструктор ботів Telegram — це сервіс, який дозволяє створювати роботів, які можуть відповідати на запитання, виконувати безліч корисних функцій замість людей. Завдяки всім видам ботів для месенджера власники каналів можуть підтримувати зв’язок зі своїми передплатниками без перерв, заощаджуючи час і сили. Помічники можуть виконувати корисні завдання без написання окремого коду.

Основні команди для бота. Бот в основному використовується для підтримки зв'язку з користувачами в Telegram. До основних функцій помічника можна віднести: надавання повної інформації відвідувачам чату, демонстрація оголошень про події, адреси та інші корисні дані, відповідь на стандартні запитання (закладені під час розробки), поширення посилань на соціальні мережі, сайти, ознайомлення з новинами, оглядами товарів тощо, допомога при налаштувані підтримки користувачам, оформлення замовлення або реєстрація нового ділового партнера (користувача), проведення опитувань, для отримання достовірних даних, і т д [19].

Кожен користувач може створити будь-якого бота. Їх умовно поділяють на:

• Чат-боти. Вони потрібні для підтримки активності на каналі, розмови з передплатниками. Якщо його правильно налаштувати, то користувачі можуть навіть не зрозуміти, що спілкуються з роботом, а не з живою людиною.

• Інформатори. З їх допомогою адміністратор групи розміщує нові матеріали, новини тощо.

• Ігри. Їх головна функція - розвага. Деякі боти можуть грати навіть на гроші.

• Помічники. Вони допомагають власнику каналу адмініструвати, виконувати різноманітні завдання та виконувати за нього нудну монотонну роботу.

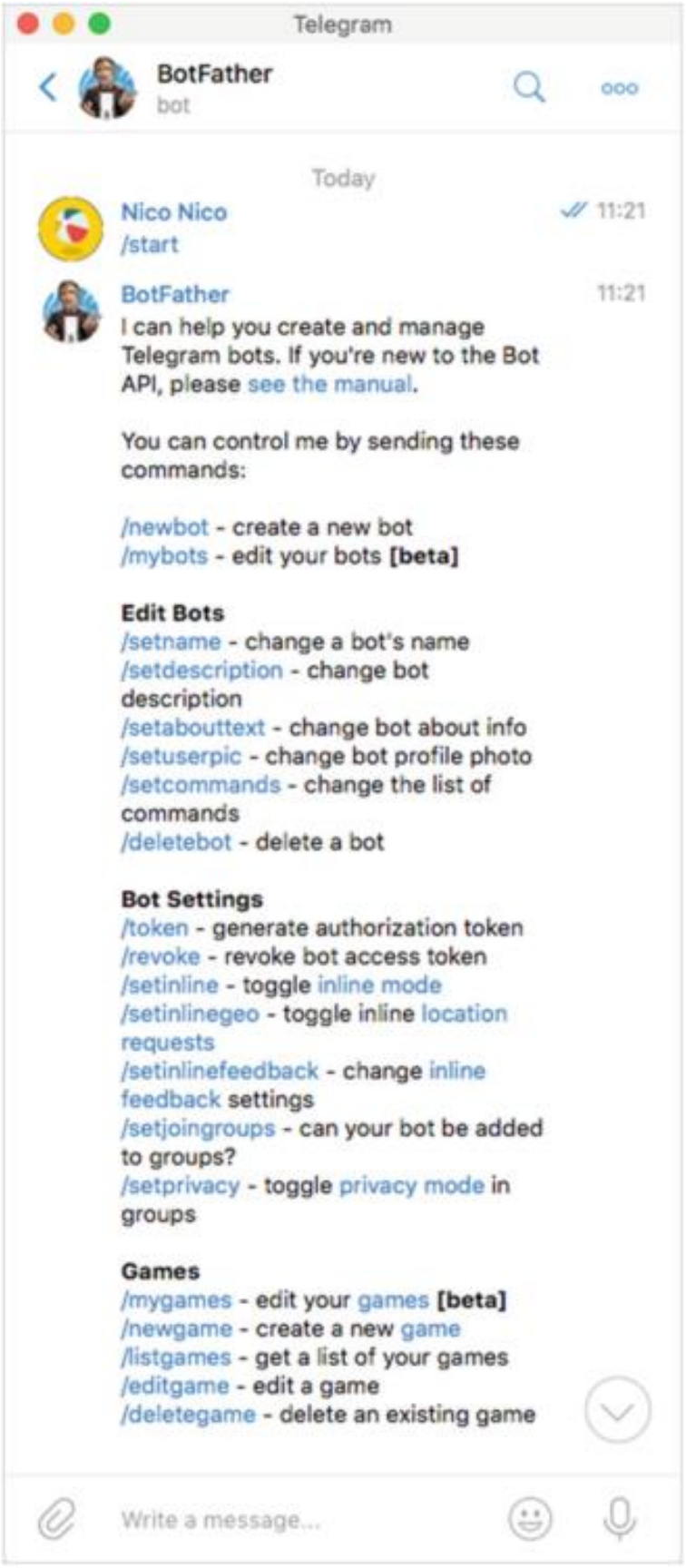

Рис.3.5. Конструктори телеграм-ботів

Спектр послуг зі створення помічника в Telegram досить широкий. Кожен з них відрізняється деякими функціями і функціональністю. Розглянемо найпопулярніші.

Manybot. Платформа інтегрована в Telegram та пропонує безкоштовно створити персональних помічників. Під час роботи з конструктором не потрібні навички програмування

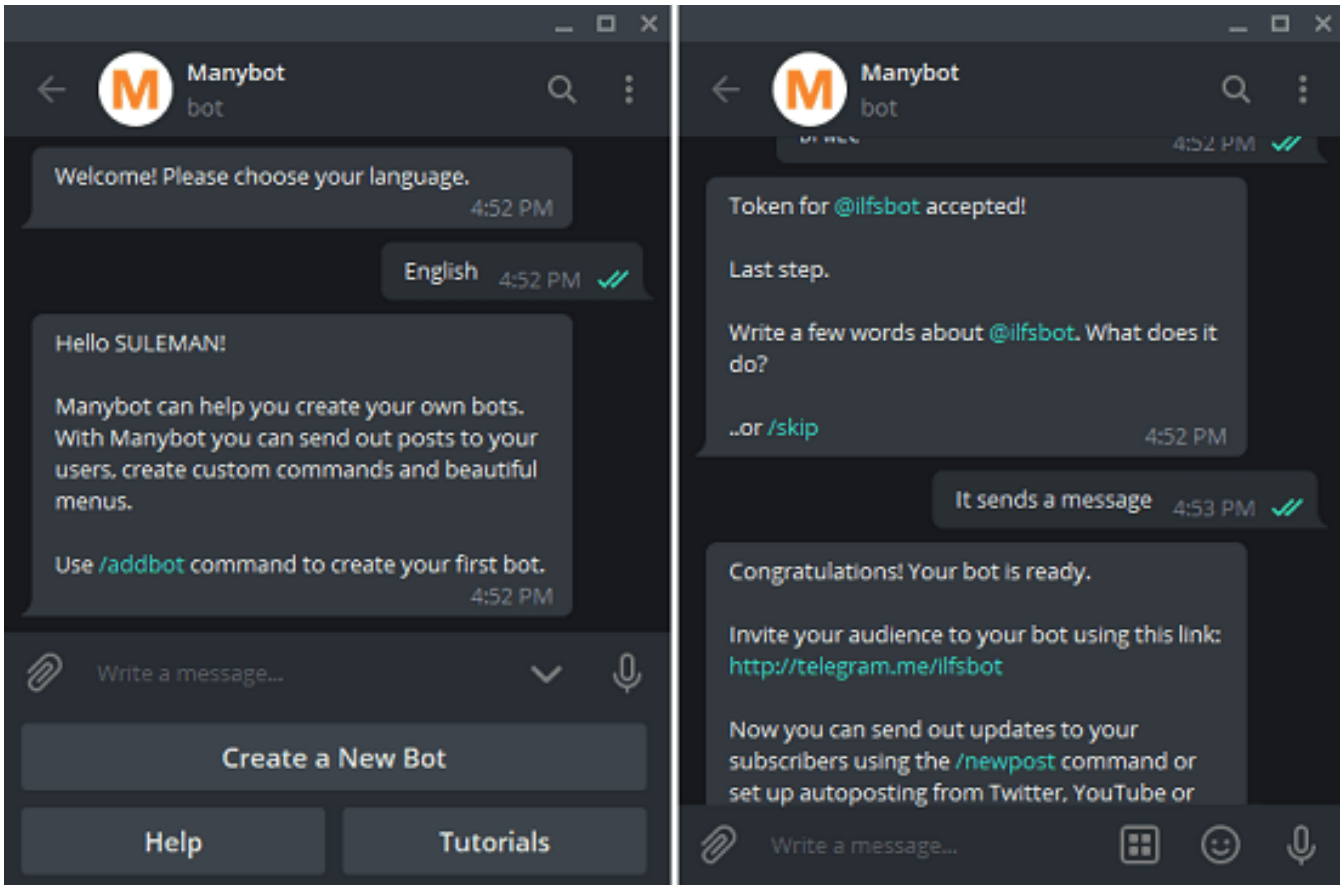

Рис.3.6. Приклад налаштування ManyBot

Даний бот може створити меню, надсилати повідомлення, публікувати автоматично з YouTube, Twitter.

Telebot. Конструктор бота для telegram швидко формує помічника з доступних шаблонів і підключає його. Для клієнта надається детальна інструкція. Використовуючи сервіс, можна створювати дії на різноманітні команди за допомогою форми для введення тексту, а також звичайної або онлайн-клавіатури.

Flow XO Service. Основною метою сервісу є створення, редагування та підтримка функціонування ботів для вирішення різного роду бізнес-завдань.

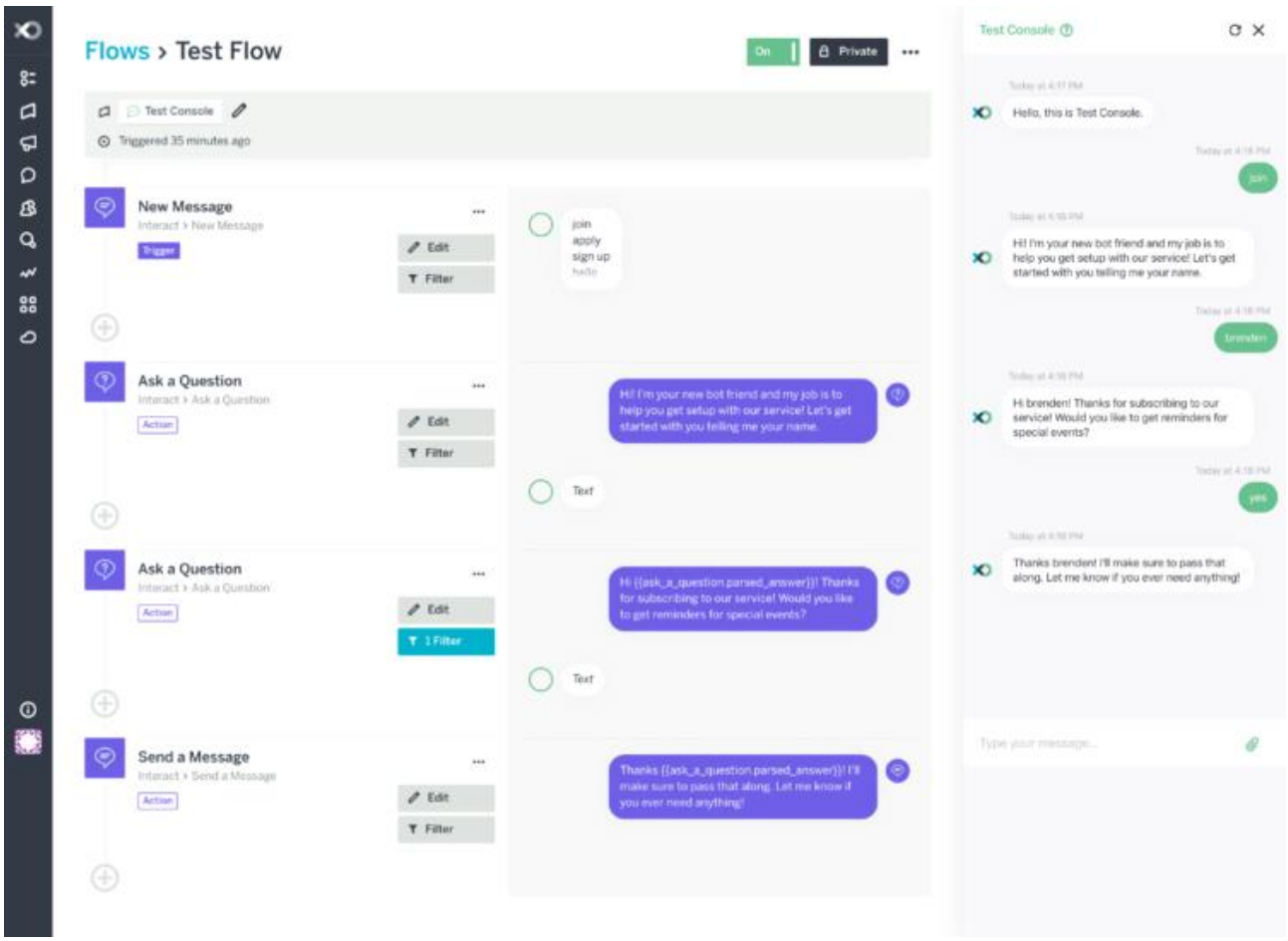

Рис.3.7. Приклад налаштування Flow XO Service

Flow XO Service має такі функції:

- використання HTTP/JSON для створення випусків;
- набір фільтрів;
- редактор брехні;
- вбудований повноцінний web-месенджер;
- тригери;
- сортування контенту;
- робота з javascript.

Деякі сервіси можна використовувати безкоштовно, є й платні платформи. Що вибрати користувачеві, залежить від того, на який результат він прагне і скільки він готовий заплатити.

3.4. Розробка бота для протидії спаму в сервісі Telegram

1. Створення бази даних. Використовуючи із таблиці конфігурацій бота, через журнали повідомлень, які використовує ШІ для вивчення та ідентифікації поведінки спамерів, налаштування груп, навіть профілі користувачів — все це в одному великому безладі UUID. До таблиці можна додавати нові функції, не думаючи двічі, іноді додаткові стовпці [22].

Рис.3.5. Приклад реалізації БД для таблиці конфігурацій бота

В цьому випадку можна зіштовхнутися з проблемою. Адже таблиця з журналами має вагу, та злегкістю може перевищити і 30, і 50 ГБ, половина з яких буде індексами. При цьому запити стають досить повільними, і доводиться декілька раз оновлювати SQL-сервер, щоб обробляти зростаючий трафік. Міграція на MySQL 8 призвела до 8 годин простою протягом ночі. Це було далеко не прийнятно і водночас надзвичайно напружено, оскільки головною метою цього проекту було надання користувачам найшвидшого телеграм-бота, який може легко керувати своїми групами. Тому результатом стало планування нового макета, відображення зв’язків, а потім кодування всіх міграцій для нового макета.

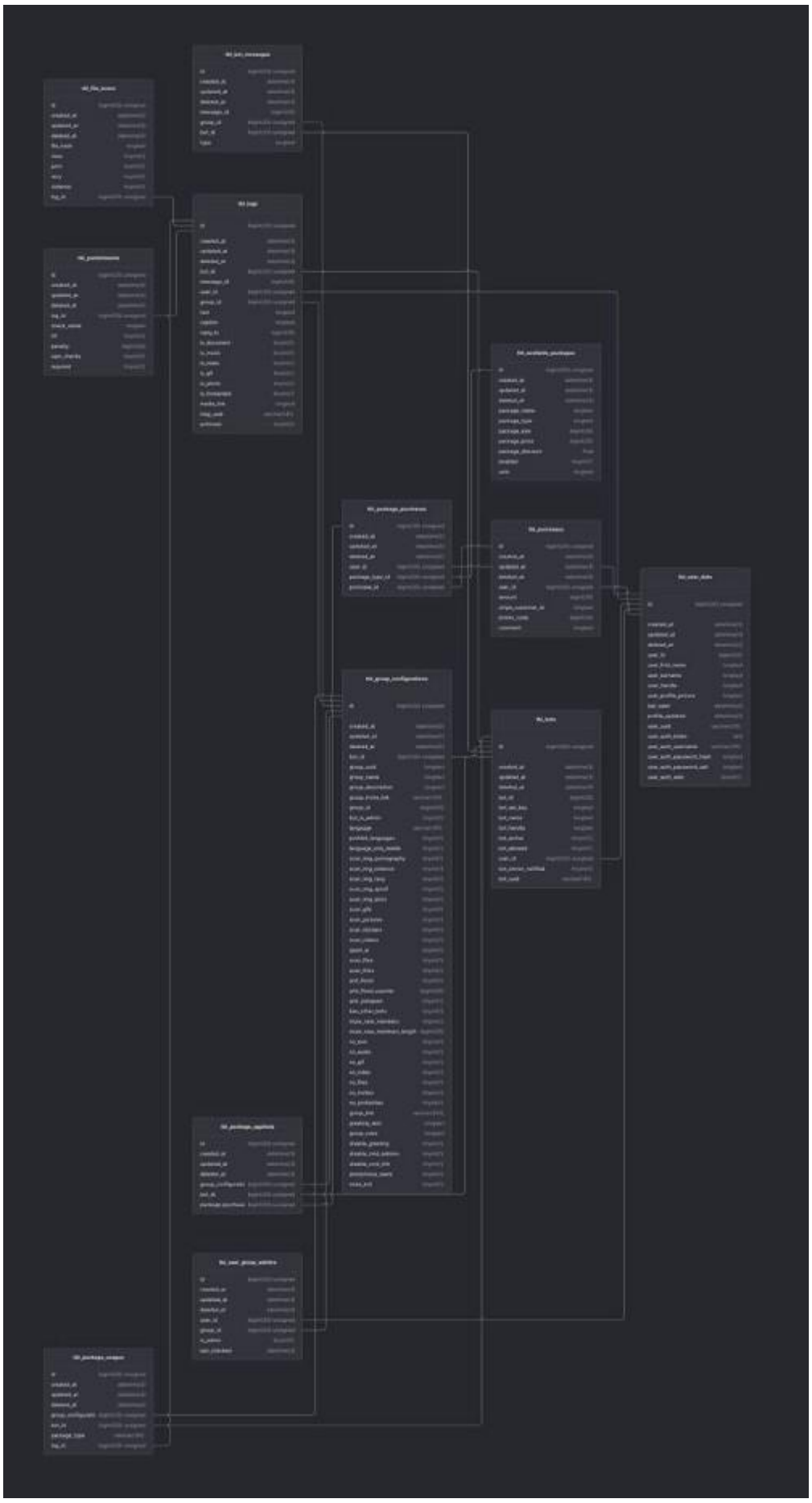

Рис.3.6. Приклад упорядкованої БД для налаштування бота

Рис.3.5 та рис.3.6 були розташовані в одній базі даних, служби протягом трьох днів тестування використовували обидві версії, щоб переконатися, що нічого не бракує, і всі запити працювали та отримували інформацію, яку вони повинні були отримати.

2. Спостереження за поведінкою мікросервісів (моніторинг усіх запитів). Традиційно це аналіз та спостереження за журналом запитів і запитів без використання індексів, а також - відповідна реакція. Вся ця робота дозволила знову зменшити масштаб екземплярів SQL і залишитися нижче 40% використання із 600/100 запитів в секунду.

3. Фінальне рішення. Для виконання цього кроку необхідно додати бота до планувальника завдань на власній машині (Windows, наприклад), а для цього потрібно виконити наступні дії:

- Створити завдання;
- Додати назву завдання (це потрібно для умови, коли ПК вимкнено/заблоковано);

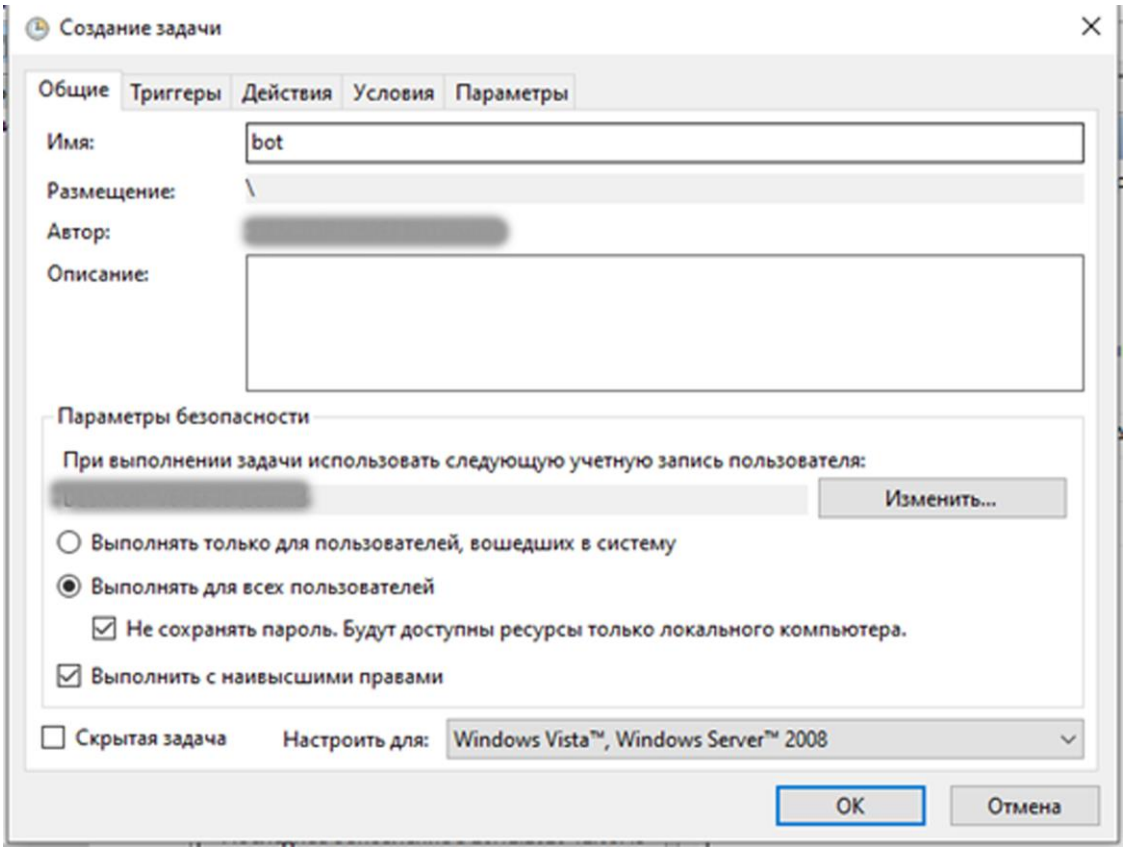

Рис.3.7. Створення завдання

- В закладинці «Дії» необхідно додати файл боту;
- В закладинці «Умови» додати вимоги, як показано на рис.заповнити так як на рис. 3.8.

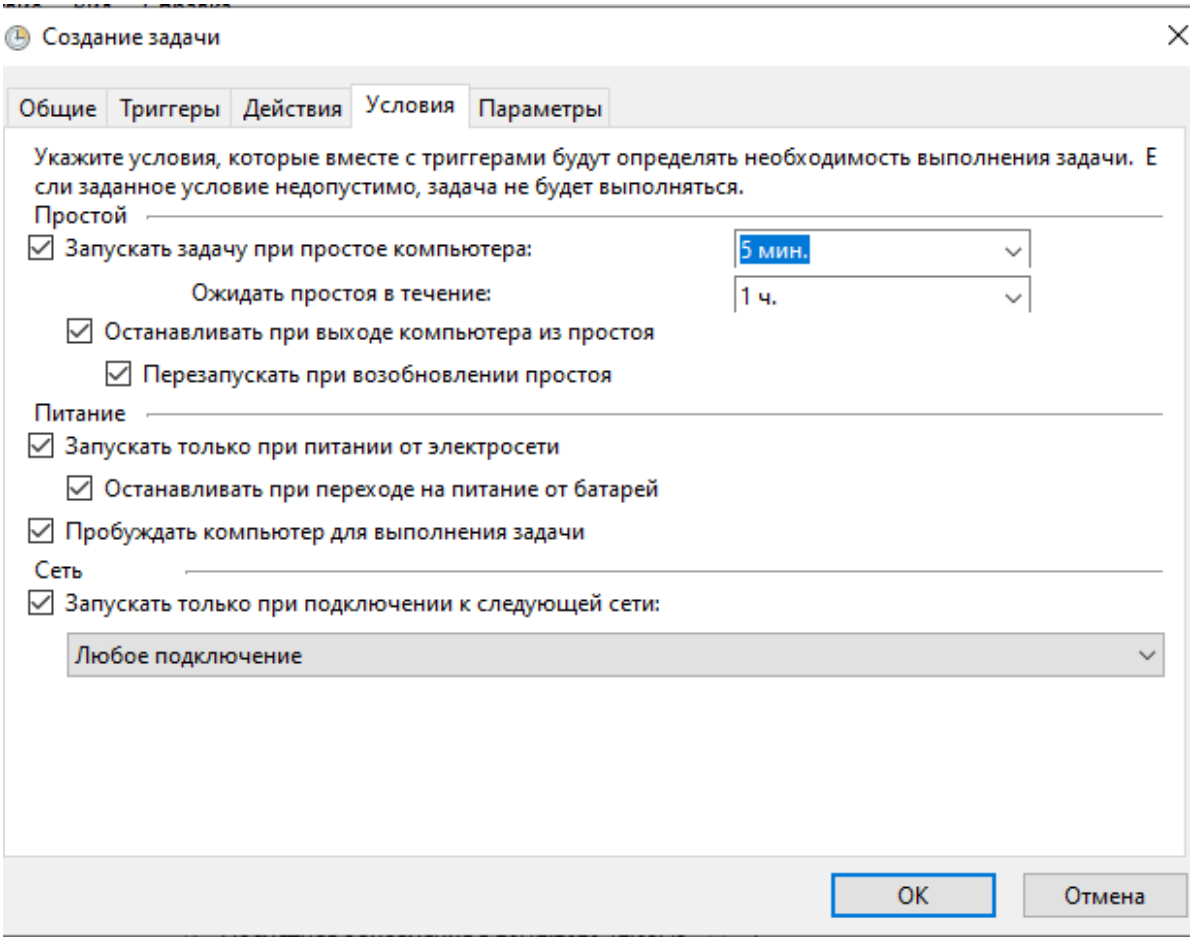

Рис.3.8. Додавання тригерів

- Фінальний крок – завершення завдання.

Перевірити виконання завдання можна шляхом звернення до «Диспетчер завдань» (рис.3.9.).

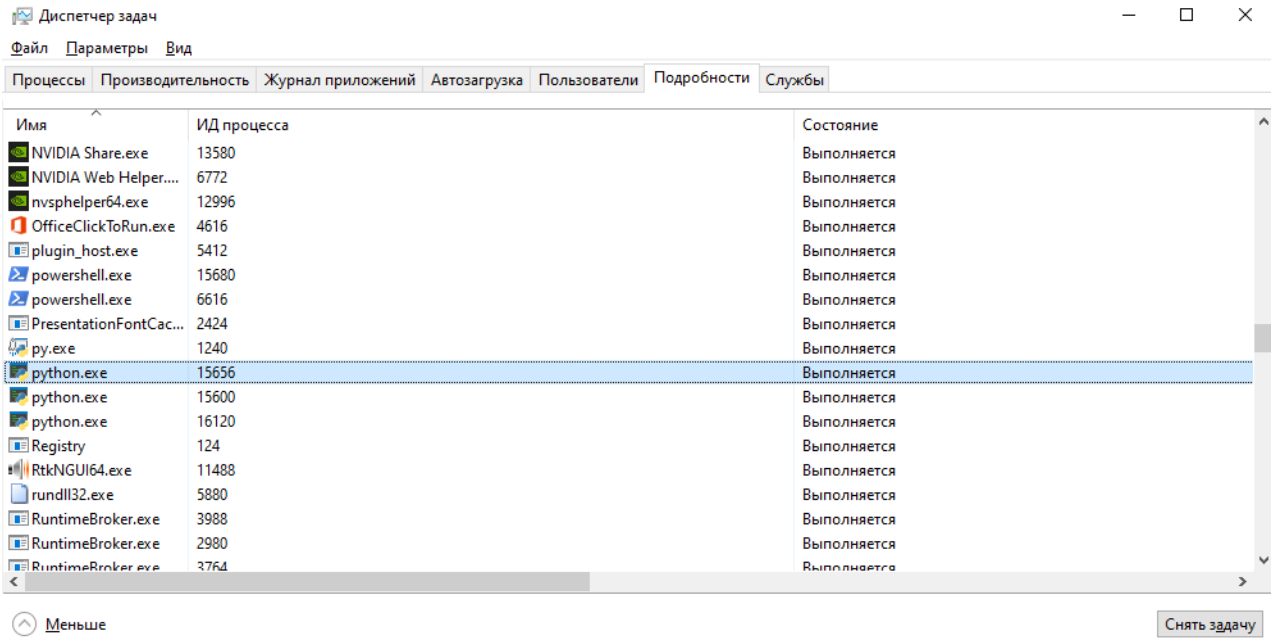

Рис.3.9. Екземляр боту (процес Python.exe)

Ще одним із варіантів перевірки можна вважати виконання в командному рядку команду «netstat -n -p». Маючи інформацію щодо ідентифікатора процесу можна також і отримати інформацію за яким портом працює бот[18].

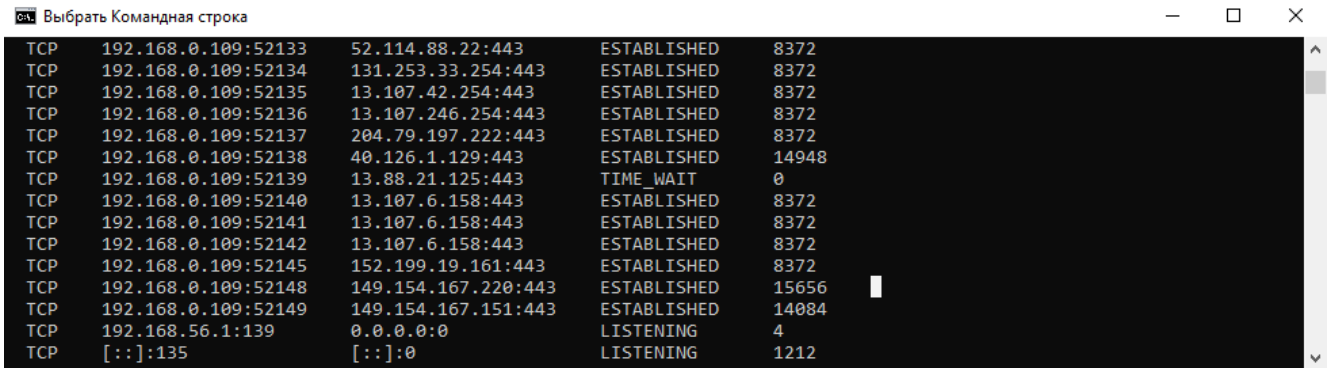

Рис.3.10. Результат netstat

Висновки до третього розділу

Розглянуто приклади і методи захисту від небажаного трафіку в сервісі Telegram. Визначено, що причиною отримання спаму не завжди є добре спланована атака. Іноді самі учасники групи або користувачі каналу ненавмисно додають бота, відкриваючи таким чином простір для активності.

Продемонстровано варіанти створення власного бота за допомогою наявних прикладів, які вже існують в середовищі сервісу Telegram. При цьому зазначено, що ці боти можуть бути вразливі до спаму та фішингу через тих розробників, які над ними працювали.

Приведено алгоритм розробки власного боту, а також особливості реєстрації його на власному ПК. В цій реалізації не виникає сумнівів щодо виконання розробленим ботом всіх тих функцій, які потрібні користувачам. Адже достовірність кожної із команд може бути гарантована саме розробником.

ВИСНОВКИ

В магістерській роботі отримано наступні результати:

1. Описано особливості налаштувань безпеки в сервісі для миттєвого обміну повідомленнями Telegram.

2. Досліджено протоколи для безпечного обміну повідомленнями, які забезпечують наскрізне шифрування повідомлень. Зазначено, що протоколи безпечного обміну повідомленнями не забезпечують кожну властивість безпеки, а це означає, що є можливість покращення для всіх протоколів. Підкреслено, що незважаючи на те, що Telegram вважається одним із найбільш безпечних сервісів обміну повідомленнями, їхній власний протокол MTProto не був повністю розглянутий експертами з криптоаналітики.

3. Проаналізовано вразливості безпеки E2EE для Telegram і запропоновано типові базові складовідля E2EE в загальній системі обміну повідомленнями.

4. Розглянуто приклади і методи захисту від небажаного трафіку в сервісі Telegram. Визначено, що причиною отримання спаму не завжди є добре спланована атака. Іноді самі учасники групи або користувачі каналу ненавмисно додають бота, відкриваючи таким чином простір для активності.

5. Продемонстровано варіанти створення власного бота за допомогою наявних прикладів, які вже існують в середовищі сервісу Telegram. При цьому зазначено, що ці боти можуть бути вразливі до спаму та фішингу через тих розробників, які над ними працювали. Приведено алгоритм розробки власного боту, а також особливості реєстрації його на власному ПК. В цій реалізації не виникає сумнівів щодо виконання розробленим ботом всіх тих функцій, які потрібні користувачам. Адже достовірність кожної із команд може бути гарантована саме розробником.