НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп'ютерного практикуму

ДОСЛІДЖЕННЯ СИСТЕМ ЗАХИСТУ ЗАХИЩЕНИХ МЕСЕНЖЕРІВ

Виконали студенти
групи ФІ-32мн
Мельник Ілля,
Міснік Аліна

Перевірила:
Селюх П.В.

Київ — 2024

Мета роботи: Дослідження особливостей реалізації криптографічних механізмів протоколів захисту мультимедійної інформації типу SIP

Постановка задачі: Дослідити криптографічні методи, що можуть або вже використовуються у месенджерах, описати їх проблематику реалізації.

1 ХІД РОБОТИ

1.1 Key Agreement

Узгодження ключів є одним з основних протоколів ініціалізації сесій. Якщо ви хочете встановити спілкування по деякому зашифрованому каналу, то для побудови його, вам необхідно обмінятися ключами. Зазвичай ця процедура проходить за всім відомою схемою Діффі-Хелламана. Це вважається каноном для схем шифрування, де спілкуються лише двоє людей. У групах, цей алгоритм може використовуватися, але це вважається не надто гарним підходом, оскільки він вимагає тоді узгоджувати секрет з усіма. Наведемо декілька цікавих конструкцій, що базуються на цій схемі:

1) **X3DH.** Цей алгоритм використовується у Signal. Як можна зрозуміти з назви, він використовує 3 сторони – сервер та двох користувачів. Сервер необхідний для забезпечення роботи в асинхронному режимі, оскільки користувачі можуть бути не завжди у мережі. В такому разі, вони завантажують спеціальний сет одноразових ключів, які може взяти інший користувач за запитом у сервера і написати перше повідомлення тим самим створивши ініціалізацію сесії. Якби такої можливості не було, то користувачі могли б спілкуватися лише в онлайн форматі. Цей алгоритм також використовується як початковий алгоритм у протоколі Double Ratchet. Наведемо спрощений опис схеми:

а) Боб публікує свій ідентифікаційний ключ і попередні ключі на сервері.

б) Аліса отримує «набір попередніх ключів» із сервера та використовує його для надсилання початкового повідомлення Бобу.

в) Боб отримує та обробляє початкове повідомлення Аліси.

2) **PQXDH.** Це новий алгоритм узгодження ключів, який наразі впроваджується у Signal. Він використовує постквантову криптографію – інкапсуляцію ключів, що дозволяє ускладнити задачу для зловмисника, а

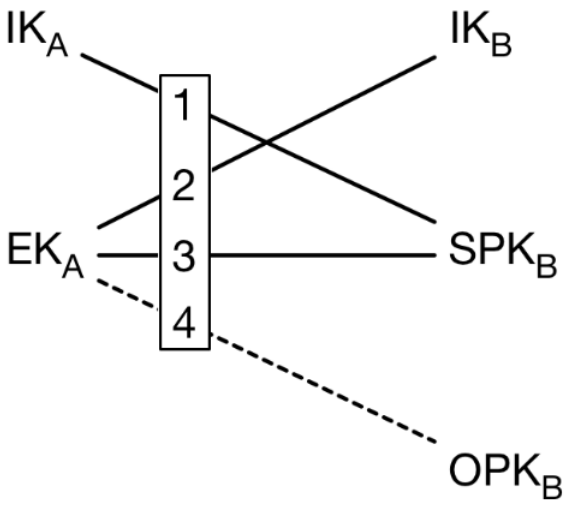


Рисунок 1.1 – Взаємодія між ключами користувачів у X3DH

саме вирішення по суті двох задач для взлому системи. Алгоритм використовує стандартизований алгоритм інкапсуляції ключів – Crystals-Kyber-1024 (IND-CCA secure algorithm from NIST).

3) **ZRTP.** Це досить старий, але надійний протокол узгодження ключів, що зазвичай використовується для передачі медіапотоків по VoIP або IP телефонія. Він також використовує алгоритм Діффі-Хеллмана для узгодження ключів між двома користувачами, але окрім цього, він дозволяє генерувати деяку важливу структуру для автентифікації як SAS (Short Authenticated String). Цей рядок є короткою інтерпретацією секрету і дозволяє точно звіритися користувачам навіть просто продиктувавши їх. SAS формується простим застосуванням геш-функції від секрету і по суті дає зловмиснику лише один варіант, щоб вгадати секретний рядок. За такої ситуації навіть 16 бітний рядок буде мати незначну ймовірність вгадування (приблизно 0.00002). ZRTP може підтримувати декілька режимів роботи, дозволяючи не генерувати кожен раз новий секрет (припустимо ви маєте обмежені потужності або мали технічні збої), але варто розуміти, що в такому разі стійкість до атак різко падатиме, тому даним режимом не варто користуватися досить часто і частіше використовувати звичайний режим, що генерує новий секрет між користувачами.

Оскільки узгодження ключів за алгоритмом Діффі-Хеллмана є досить

популярним і по суті не існує концептуальних суперників йому, то можна підсумувати, що відомі месенджери можуть ініціалізувати обмін ключами у текстових чатах схожим до X3DH алгоритмом, а при спілкуванні голосовим дзвінком чи по відеозв'язку ZRTP-подібним протоколом.

1.2 Encryption

Месенджер можна назвати приватним і безпечним, якщо виконується наскрізне шифрування. Тобто мати доступ до розшифрованого повідомлення можуть мати лише кінцеві користувачі – відправник і отримувач. Усі інші піри чи ноди, які брали участь у ланцюгу передачі не можуть розшифрувати чи отримати якусь інформацію з повідомлення. Звичайно було б ідеально ще приховувати й метадані, щоб месенджер мав ще і властивість анонімності, але ця реалізація є досить складною. По-перше, якщо ми можемо приховати відправника, то отримувача ми не можемо приховати з причини того, система повинна знати куди надсилати повідомлення. Звичайно можна використовувати різні протоколи міксування (TOR або маловідомий NYM на основі Sphinx), але навіть ці протоколи не можуть вам завжди гарантувати безпеку. Тому поки опустимо проблему анонімності й перейдемо до опису досягнення приватності.

Для правильного шифрування повідомлень, протокол повинен мати наступні властивості:

– Forward secrecy. Отримавши поточний ключ, зловмисник не може отримати майбутні ключі для розшифрування наступних повідомлень.

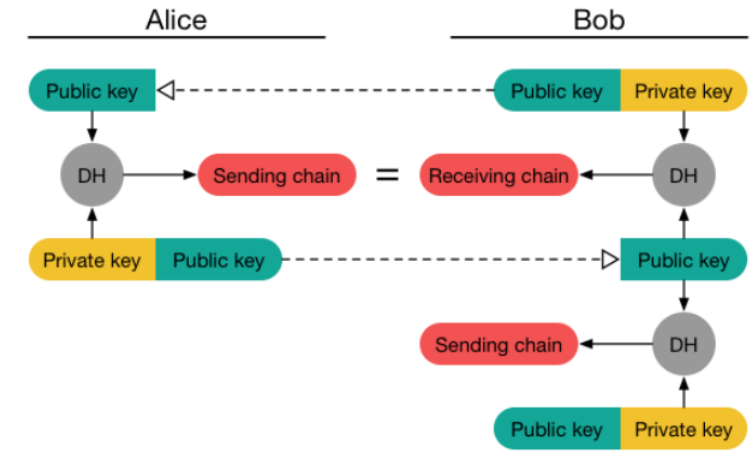
– Backward secrecy. Отримавши поточний ключ, зловмисник не може розшифрувати попередні повідомлення/отримати старі ключі.

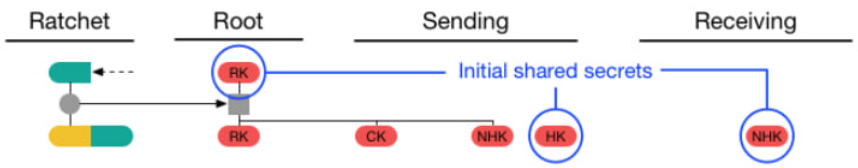
Якщо алгоритм відповідає заданим властивостям, то він є досить безпечним для використання і має достатню секретність. І якщо говорити про можливі варіації алгоритмів, то існує лише один відомий підхід шифрування у багатьох месенджерах, що забезпечують приватність – алгоритм Double Ratchet.

1.2.1 Double Ratchet

Double Ratchet або подвійний храповик – це досить унікальна конструкція шифрування, що підтримує обидві властивості секретності – forward і backward. Це успішно досягається з допомогою зміною ключів на кожному кроці, що означає що кожне наступне повідомлення шифрується новим ключем, а генерація цих ключів відбувається з допомогою KDF функції. KDF функція дозволяє отримувати ключ або ключі, подавши на вхід деякий ключовий вміст. Якщо коротко, то дана функція дозволяє або згенерувати декілька ключів з одного, або отримати новий ключ. Double Ratchet можна поділити насправді на дві конструкції: symmetric ratchet і DH ratchet або asymmetric ratchet.

Схема цього алгоритму виглядає наступним чином:





Кожен користувач має три ланцюги: кореневий, відправника та отримувача. Кореневий ланцюг містить ключі, що використовуються для входу у функцію KDF, що генерує значення наступного кореневого ключа і ключа для шифрування повідомлення. Окрім нього, на вхід у KDF подається спільний секрет. Ключа для шифрування/дешифрування повідомлення записується у відповідно до ролі ланцюг відправника/отримувача, а також саме зашифроване повідомлення. Окрім цього варто пам'ятати, що старі ключі видаляються.

Деякі месенджери використовують спрощену версію даного протоколу, об'єднавши ланцюги відправника й отримувача в один. Розрізнення повідомлення у такому випадку відбувається за підписом користувачів (алгоритм Olm для Matrix).

На жаль для використання у багатокористувацьких чатах (групах), цей алгоритм не підійде для реалізації через те, що кожному користувачеві прийдеться запускати Double Ratchet з усіма іншими. Це обчислювально-складно та має свої недоліки при забезпеченні загальної синхронізації. Тому для шифрування у групах пропонується використовувати звичайний symmetric ratchet, що просто генеруватиме новий ключ для кожного повідомлення у групі, а користувачі лише підписуватимуть своїми власними ключами.

У підсумку можна сказати, що усі відомі відкриті реалізації, такі як Signal, Matrix, Element використовують Double Ratchet для спілкування між учасниками та Symmetric Ratchet для спілкування у групах.

1.3 Broadcast encryption

Broadcast encryption або технологія широкомовного шифрування – це дуже недооцінений пласт, що не має оптимальних підходів. У разі успішності, ці алгоритми дали б змогу шифрувати канали, тобто групи де один блогер публікує якусь інформацію своїм підписникам (умовно за якусь плату). Наразі усі відомі алгоритми можна розділити за наступними

критеріями:

– Побудовані на асиметричній чи симетричній криптографії

– Шифрування за ID-based чи ні

– Статичні чи динамічні

– Шифрування на невелику кількість користувачів з великої кількості та на велику кількість за умови невеликої кількості відкликаних користувачів.

Найбільше проблем виникає з останнім критерієм, оскільки ми не можемо забезпечити оптимальний усереднений випадок.

Тому наразі можна підсумувати, що або канали не мають наскрізного шифрування або використовують якийсь один з відомих підходів.

1.4 Call and videocall Encryption

Оскільки дзвінки відбуваються лише у випадку, коли обидва користувачі онлайн, то в такому випадку ми можемо використовувати протоколи які можуть працювати лише в синхронному режимі. Це полегшує з однієї сторони нам роботу в організації такого спілкування, але водночас мережа починає стикатися з іншою проблемою – безперервне спілкування. Коли в нас відбувається дзвінок, то відбувається обмін зашифрованими пакетами даних. Це може бути лише один потік даних або декілька (аудіо- та відеопотоки). Окрім цього ми повинні розуміти, що організація транспортування потребує регуляризації кількості та розмірів пакетів відносно швидкості доставлення та можливої затримки. Оскільки існує не так багато реалізацій, ми наведемо як це працює у Signal.

Signal реалізував свою власну структуру SFU (Selective Forwarding Units) для реалізації шифрованих дзвінків (саме підтримка групових). Наразі дана структура дозволяє підтримувати дзвінки з 40 користувачами. Перед тим, як описати як працює SFU, сформуємо більш чітко, що повинна вирішити дана система:

1) Пропускна здатність інтернет-з'єднання кожного учасника постійно

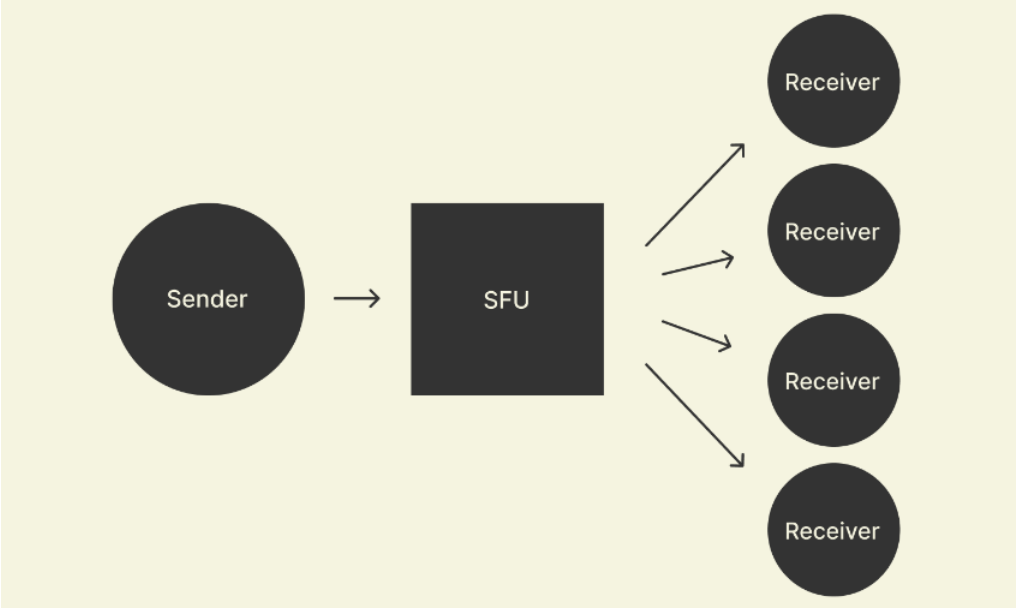


Рисунок 1.2 – Загальна схема взаємодії з SFU

змінюється. SFU повинен постійно і ретельно коригувати кількість даних, які він надсилає кожному учаснику, щоб все було «як треба».

2) SFU не може змінювати медіа, які він пересилає; кожен учасник повинен надсилати SFU відео з різною роздільною здатністю, а SFU повинен постійно і ретельно перемикатися між ними.

А тепер опишемо техніки, які можуть ці задачі вирішити:

– **Simulcast and Packet Rewriting** – Одночасне надсилання шарів кожним учасником до SFU. Це дозволяє SFU перемикатися між різними роздільними здатностями. Деякі відеокодеки, такі як VP9 або AV1, роблять це легко: перемикання шарів вбудовано у відеокодек у спосіб, який називається SVC. Оскільки Signal все ще використовує VP8 для підтримки широкого спектра пристроїв, і оскільки VP8 не підтримує SVC, SFU повинен зробити щось, щоб перетворити 3 шари в 1. Подібно до сервера потокового відео, SFU надсилає вам відео з різною роздільною здатністю. Але на відміну від сервера потокового відео, тут нічого не зберігається, і він повинен робити це повністю на льоту. Це відбувається за допомогою процесу, який називається перезаписуванням пакетів. Перезапис пакетів – це процес зміни міток часу, порядкових номерів та подібних ідентифікаторів, які містяться в медіапакеті й вказують, де на часовій

шкалі медіапакету знаходиться його місце. Він перетворює пакети з багатьох незалежних часових шкал медіа (по одному для кожного шару) в одну уніфіковану часову шкалу медіа (один шар).

– **Congestion Control** – це механізм, який визначає, скільки даних потрібно надсилати мережею: не надто багато і не надто мало. Він має довгу історію, здебільшого у вигляді контролю перевантажень TCP. У Signal Calling Service реалізовано googcc і transport-cc у вигляді потокової обробки. На вхід потокового конвеєра надходять дані про час відправлення та отримання пакетів, які називаються квитанціями (acks). Виходами конвеєра є зміни в тому, скільки потрібно відправити через мережу, які називають цільовою швидкістю відправлення. Якщо коротко описати схему, то одержувач надсилає періодичні оновлення про те, коли він отримав посилку. А відправник поєднує цю інформацію з власною, обчислюючи час затримки. Перші кілька кроків потоку наносять acks на графік залежності затримки від часу, а потім обчислюють нахил, щоб визначити, чи затримка збільшується, зменшується чи залишається незмінною. Останній крок вирішує, що робити на основі поточного нахилу.

– **Rate Allocation** – допомагає визначити, які рівні потрібно пересилати. Якщо бюджет достатньо великий, ми можемо надіслати все, що хочемо. Але якщо ні, ми повинні розставити пріоритети. Щоб допомогти у визначенні пріоритетів, кожен учасник повідомляє серверу, яка роздільна здатність йому потрібна, запитуючи максимальну роздільну здатність. Використовуючи цю інформацію, протокол працює за наступними правилами:

1) Шари, більші за запитувану максимальну роздільну здатність, виключаються;

2) Менші шари мають пріоритет над більшими;

3) Більші запитувані роздільні здатності мають пріоритет перед меншими запитуваними роздільними здатностями.

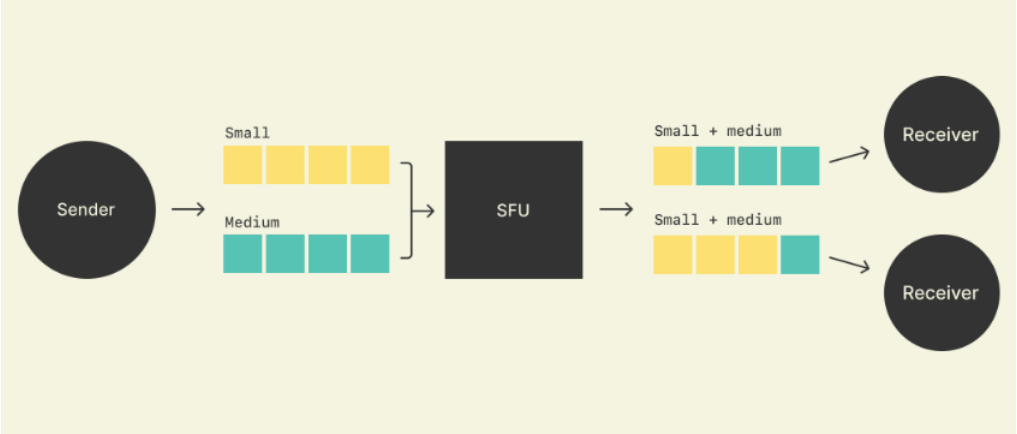


Рисунок 1.3 – Розподіл пакетів

1.4.1 WebRTC

Виділимо окремо бібліотеку WebRTC на основі якої будуються багато реалізацій дзвінків. В якомусь сенсі можна сказати, що реалізація Signal це покращена версія даної бібліотеки, тому що ця бібліотека розв'язує проблеми не лише улаштовування пакетів, а й транспортування їх. Припустимо поборення для встановлення зв'язку перешкод таких як NAT чи фаерволли. Для цього використовують спеціальний протокол ICE (Interactive Connectivity Establishment).

ВИСНОВКИ

В даній роботі ми розглянули основні алгоритми шифрування у месенджерах, розглянули їх переваги й проблематики. З нашого дослідження для чатів один на один більшість анонімних месенджерів використовують алгоритм узгодження ключів X3DH. Для відеодзвінків та аудіодзвінків можна використовувати ZRTP. Безпечна перевірка з'єднання може бути здійснена тільки в реальному часі за допомогою QR-коду або посилання. Інші способи є небезпечними. Для каналів, реалізації broadcast encryption, на жаль, не мають ефективного підходу у зв'язку з не вирішуваністю деяких проблем.