НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 4

«ДОСЛІДЖЕННЯ СИСТЕМ ОБМІНУ ПОВІДОМЛЕННЯМИ ТА IP-ТЕЛЕФОНІЇ»

Виконали:

Студенти групи ФІ-22 мн

Ковальчук Ольга

Коломієць Андрій

Толмачов Євгеній

Київ – 2024

Мета та основні завдання роботи

Дослідження особливостей реалізації криптографічних механізмів систем обміну повідомленнями та IP-телефонії.

Вимоги

Проаналізувати існуючу інформацію про системи Viber, WhatsApp, Skype, Telegram та їх криптографічні механізми. Детально розібрати опис усіх механізмів протоколу, структуру пакетів та характеристики систем. Довести теоретично можливість існування в системі виявлених протоколів та зробити огляд відомих аналізів захищеності вказаних протоколів, включаючи вже виправлені помилки. Зробити порівняльний аналіз можливостей вказаних систем, їх криптографічних механізмів та рівня захищеності (обгрунтований). Дати рекомендації користувачам щодо безпечного використання таких систем. Всію зібрану інформацію офрмити у вигляді детального звіту з власним аналізом рівня захищеності та обраних криптографічних механізмів.

Виконання роботи

Viber

https://www.viber.com/app/uploads/viber-encryption-overview.pdf

Терміни

Обліковий запис Viber – набір пристроїв, зареєстрованих у Viber на один номер телефону. Обліковий запис складається з одного основного пристрою та (необов'язково) необмеженої кількості додаткових пристроїв. Повідомлення, надіслані або отримані будь-яким пристроєм, відображаються на всіх інших зареєстрованих пристроях того самого облікового запису, починаючи з моменту їх реєстрації й далі (попередня історія не видно).

Основний пристрій – мобільний телефон під керуванням iOS, Android або Windows Phone, зареєстрований у службі Viber за номером телефону, який керує мобільний оператор.

Додатковий пристрій – зазвичай iPad, планшет або настільний ПК, який має бути підключений до основного пристрою.

ID ключ- 256-бітна пара ключів побудованих на еліптичній кривій Curve25519, яка використовується для ідентифікації облікового запису Viber. Ідентифікаційний ключ генерується лише основним пристроєм.

PreKeys - набір середньострокових пар ключів Curve25519, які використовуються для встановлення захищених сеансів один до одного між пристроями. Попередні ключі генеруються окремо кожним пристроєм в обліковому записі.

Сеанс - двостороннє безпечне віртуальне з'єднання між двома певними пристроями. Захищені сеанси автоматично встановлюються між усіма пристроями одного облікового запису та між двома пристроями різних облікових записів, які обмінюються повідомленнями. Захищений сеанс з іншим обліковим записом позначається у Viber сірим замком на екрані розмови.

Довіра - стан між двома рахунками, який вказує що одноранговий обліковий запис автентифіковано, а що ні зловмисник "людина посередині" втрутився в між будь-якими двома пристроями в аккаунт. Користувачі можуть встановити довіру, слідуючи процедуру автентифікації, описану нижче. А довірений сеанс позначається у Viber зеленим замком.

Зловживання довірою - стан між двома обліковими записами це відбувається, коли ідентифікаційний ключ довіреного вузла має змінено після встановлення довіри. Це може станеться законно, якщо одноранговий користувач замінив його основний пристрій, перевстановив Viber, або якщо ключі були втрачено через несправність зберігання. Однак це також може трапиться, якщо людина посередині підслуховує розмову. Порушений сеанс визначається в Viber червоним замоком.

Підготовка до налаштування сесії

Під час встановлення кожен основний пристрій Viber генерує одну 256-бітну пару ключів Curve-25519 під назвою ID Key. Приватна частина ключа зберігається на пристрої, а публічна частина ключа завантажується на сервери Viber. Додаткові пристрої (ПК і планшети) отримують закритий ключ від основного пристрою за допомогою безпечного методу передачі даних.

Крім того, кожен клієнт Viber генерує серію PreKeys. Кожен PreKey має два 256-бітних ключа Curve-25519-пари, що називаються ключом рукостискання та ключем храпового механізму. Приватні ключі зберігаються на пристрої, тоді як

відкриті ключі завантажуються на сервер. Сервер містить кілька ключів на пристрій і запитує більше, коли вони падають нижче настроюваний поріг. Кожен запит змушує клієнт генерувати додаткові ключі та завантажувати загальнодоступні частини на сервер.

Налаштування безпечного сеансу

Сеанс потрібно встановлювати лише один раз між кожними двома пристроями Viber, які бажають спілкуватися надійно. Коли сеанс існує, його можна використовувати для надсилання необмеженої кількість повідомлень в будь-якому напрямку.

Щоб надіслати захищене повідомлення, потрібні захищені сесії, що існують між пристроєм-відправником і всіма пристроями одержувачами, а також між надсилаючим пристроєм і всіма іншими пристроями відправника. Тому наприклад, якщо користувач А має мобільний телефон і ПК зареєстрований у Viber під тим самим обліковим записом щоб спілкуватися з користувачем В, який має мобільний телефон і ПК, необхідно встановити захищені сесії між кожною парою пристроїв.

Сеанси між пристроями одного облікового запису є встановлюється під час реєстрації пристроїв. Тільки потрібен один сеанс між будь-якими двома пристроями, і цей сеанс можна використовувати для синхронізації будь-якого кількість розмов з іншими обліковими записами Viber.

Щоб встановити сеанс з іншою обліковий запис, пристрій («Аліса»), який бажає встановити сеанс із одноранговим користувачем («Боб») надсилає запит до серверу Viber з номером телефону одержувача. Сервер відповідає відкритим ID Key ключем однорангового пристрою та рядом одноранговий відкритих попередніх ключів, по одному на кожен пристрій зареєстрований в обліковому записі акаунта «Боба». Пристрої «Боба» коли це станеться, не потрібно бути онлайн.

Потім пристрій «Аліса» генерує два 256-бітних Curve-25519 поєднує ключі як власне рукостискання та храпові ключі та отримує кореневий ключ наступним чином:

RootKey = SHA256 (DH(IDAlice, HSBob) || DH(HSAlice, IDBob) || DH(HSAlice, HSBob))

Потім RootKey використовується для отримання ключа сеансу за допомогою:

TempKey = HMAC_SHA256(RootKey, DH (RatchetAlice, RatchetBob))

New RootKey = HMAC_SHA256(TempKey, "root")

SessionKey = HMAC_SHA256(TempKey, "mesg")

DH вказує на використання еліптичної кривої Діффі-Хеллмана алгоритм обміну ключами. HS вказує алгоритм рукостискання.

Різні рядки, передані до функцій HMAC дозволяє впевнитися, що навіть якщо ключ сеансу зламано, кореневий ключ не може бути отриманий назад з нього.

Потім «Аліса» надсилає «Бобу» повідомлення про початок сеансу містить власний публічний ідентифікатор ID , ідентифікатор «Боба» попередній ключ (pre-key), який використовувався для цього сеансу, і його власний публічне рукостискання та храпові ключі. Коли "Боб" виходить в Інтернет і отримує це повідомлення зі своєї вхідної скриньки, він може реконструювати той самий корінь і сеансові ключі за допомогою тієї самої процедури DH.

Слід зазначити, що якщо сесія встановлена з основним пристроєм однорангового пристрою, але не може бути встановлений із вторинним пристроєм (наприклад, тому що він перебуває в автономному режимі та поза PreKeys на сервері). розмова все одно буде захищена наскрізним захистом шифрування, але цей вторинний пристрій не буде в змозі розшифрувати ці повідомлення, тому не буде вміє їх відображати.

Обмін повідомленнями

Пристрій, який надсилає повідомлення цільовому користувачеві, йому слід зашифрувати це повідомлення для кожного сеансу з кожним пристрій, який має цільовий користувач. Щоб досягти цього, використовується ефемерний одноразовий 128-бітний симетричний ключ, генерується та використовується для шифрування тіла повідомлення з використанням алгоритму шифрування Salsa20. Цей ефемерний потім ключ повідомлення шифрується за допомогою ключа кожного одержувача сесії. Пристрій-відправник надсилає уніфіковані повідомлення на сервер, що містить один зашифрований шифротекст і набір зашифрованих ефемерних ключів. А розгортання на стороні сервера розділяє це повідомлення та доставляє відповідні частини до кожного цільового пристрою.

Два пристрої по черзі просувають через сеанс ключі, які називається храповим механізмом. Кожного разу напрямок розмови змінюється, апарат випадковим

чином генерує нову храпову пару ключів і знову виконує наступну послідовність:

TempKey = HMAC_SHA256(RootKey, DH (RatchetAlice, RatchetBob))

New RootKey = HMAC_SHA256(TempKey, "root")

SessionKey = HMAC_SHA256(TempKey, "mesg")

Ratchetthis_device є приватною частиною нещодавно отримана пара ключів. Поруч із кожним повідомленням публічна частина Ratchetthis_device також надсилається. Одержувач запускає DH за допомогою останнього приватного храповика разом із загальнодоступною парою храповиків відправника.

Ця подвійний храповик виконує дві речі: по-перше, безперервний храповий рух забезпечує вперед і зворотню секретність так навіть якщо ключі є скомпрометовані, минулі та майбутні повідомлення не можуть бути розшифрованим. По-друге, алгоритм підтримує аутентифікація однорангового вузла, оскільки ланцюг DH кореневих ключів, починаючи з ідентифікаційних ключів обох пристроїв. Якщо ідентифікаційний ключ однорангового вузла є надійним у будь-який момент, то всьому ланцюжку можна довіряти.

Зашифровані дзвінки

Кожна сторона виклику генерує ефемерний 256-біт пару ключів Curve25519. Публічна частина підписана за допомогою приватного ідентифікаційного ключа пристрою, яким обмінюються два пристрої під час фази встановлення виклику. Інші сторона автентифікує запит, використовуючи публічність однорангового ID ключ.

Кожен пристрій перевіряє підпис і виконує розрахунок DH для отримання одноразового ключа сеансу.

Сеансовий ключ дійсний лише протягом цього певний часу виклику, і не зберігається в енергонезалежній пам’яті. Потік RTP аудіо або аудіо/відеодзвінка перетворюється на SRTP і шифрується через Salsa20 алгоритм з використанням сеансового ключа.

Обмін фото та відео файлами

Клієнт-відправник генерує ефемерний симетричний ключ Salsa20 і шифрує файл. Зашифрований файл, разом із підписом HMAC завантажується на сервери Viber. Обчислюється додатковий підпис MD5 зашифровані дані та надіслані разом із файлом, щоб надати серверу зберігання простий спосіб перевірки цілісність передачі незалежно від наскрізного шифрування. Потім відправник надсилає одержувачу повідомлення Viber з ідентифікатором завантаженого файлу та ключем шифрування. Це повідомлення зашифровано наскрізним шляхом за допомогою шифрованого сеансу між відправником і одержувачем. Одержувач створює посилання для завантаження з ідентифікатора файлу, завантажує зашифрований файл і розшифровує його за допомогою ключа.

Безпечні групи

Усі члени захищеної групи мають спільний доступ до секретного ключа (симетричний ключ шифрування Salsa20) який не відомий ні Viber, ні третім особам.

Для нових груп цей спільний секрет генерується творця групи та надіслано всім учасникам за допомогою безпечного сеансу один на один, описані вище. Для незахищених груп, які були створені з минулим версії Viber, цей секрет генерується першим учасником, який надсилає повідомлення до групового чату. У додатку Viber будь-яка група може додати додаткових учасників до групи. Ці учасники отримають секрет від членів групи, який їх додав.

Груповий секрет передається вперед за допомогою a HMAC-SHA256 з кожним надісланим повідомленням. Кожна група повідомлення містить порядковий номер, який вказує скільки разів була викликана хеш-функція. Прямий секрет підтримується одностороннім алгоритм хешування; навіть якщо ключ зламано, минулі розмови неможливо розшифрувати. Минулі ключі відкидаються клієнтом і не зберігаються в непостійне зберігання, за винятком короткого вікна минулого хеші, які використовуються в умовах гонки, коли два або більше учасників пишуть до групи одночасно.

Реєстрація вторинного пристрою

Ключовою особливістю екосистеми Viber є концепція вторинних пристроїв. Додатковим пристроєм є ПК, iPad або планшет, який підключено до мобільного пристрою користувача телефону і бачить однакову історію повідомлень обох вхідні та вихідні повідомлення. Наскрізне шифрування Viber на додаткових пристроях працює наступним чином: шифрування здійснюється окремо для кожного пристрою. Якщо якийсь користувач A надсилає користувачеві B повідомлення, яке має два пристрої, то користувачеві A потрібен окремий наскрізний зв’язок сеанси з двома пристроями та шифрує дані двічі, кожен з

яких використовує інший набір ключів. Автентифікація виконується лише один раз для всього рахунок. Якщо користувач А довіряє користувачеві В, довіра є автоматично застосовується до всіх пристроїв користувача В, а не тільки один.

Автентифікація здійснюється шляхом спільного використання приватного ідентифікаційного ключа між усіма пристроями одного і того ж акаунта.Ідентифікаційний ключ генерується лише первинним пристроєм і передається на вторинні пристрої під час реєстрації безпечним способом, наступним чином: вторинний пристрій генерує ефемерний 256- розрядну пару ключів Curve-25519. Потім пристрій генерує QR-код, що містить «Viber UDID» пристрою (загальнодоступний унікальний ідентифікатор пристрою, згенерований Viber), а також публічна частина пара ефемерних ключів. Користувач використовує свій основний пристрій для сканування QR-коду. Основний пристрій також генерує 256-біт Curve-25519 пару ключів і виконує обчислення DH з відкритим ключем із QR. Результат хешується використання SHA256 для створення спільного секрету. Потім основний шифрує свій приватний ідентифікаційний ключ з цим секретом і надсилає його та публічну частину ефемерного ключа через сервери Viber до цільового пристрою, ідентифікований за його UDID як прочитаний QR-код. Зашифрований текст підписується за допомогою HMAC- SHA256. Додатковий пристрій отримує повідомлення, виконує той самий DH і хеш для отримання того самого та використовує його для розшифровки основного приватного ідентифікатора ключа. Ідентифікаційний ключ є частиною ланцюга DH, який створює спільні секрети для сесій один на один. Тому без правильного ідентифікатора вториний пристрій не може брати участь у жодних безпечних розмовах частиною якого є основний пристрій.

Автентифікація

Автентифікація надає засіб ідентифікації зловмисника, що не є посередником, що порушує наскрізну безпеку. У додатку Viber аутентифікація виконується в контексті аудіо (або аудіо/відео) виклику Viber. Під час розмови кожен користувач може натиснути на екран блокування, щоб побачити рядок цифр, який обчислюється таким чином: обидва пристрої виконують обчислення DH за допомогою власного приватного ідентифікаційного ключа та однорангового пристрою відкритий ідентифікаційний ключ, опублікований на етапі встановлення виклику. Результат DH хешується за допомогою SHA256 і скорочується до 160 біт. Потім ці 160 біт перетворюються на рядок із 48 десяткових символів (0-9). Обидві сторони мають бачити однаковий рядок чисел і порівнювати їх зачитування їх учаснику виклику. Якщо обидві сторони чують іншу, прочитайте вголос той самий рядок цифр, який вони бачать на власному екрані, це дає дуже високу оцінку ступінь впевненості в тому, що ідентифікаційні ключі не були підроблені і ніхто не in-the-middle існує для цієї розмови. Перевірка ідентифікаційного ключа захищає як безпечні дзвінки, так і безпечні чати 1-1. У дзвінках, ідентифікаційний ключ використовується для

підпису повідомлення DH обміну ключами. У чатах 1-1 ідентифікаційний ключ функціонує як корінь ланцюга DH, що веде до створення спільного секрету. В У свою чергу, 1-1 сеанс захищає групові сеанси, оскільки групові ключі обмінюються протягом 1-1 сеансів.

Skype

https://en.wikipedia.org/wiki/Skype_security#:~:text=All%20traffic%20in%20a%20session,XORed%20with%20the%20message%20content.

Skype — це система голосового зв'язку через Інтернет-протокол (VoIP), розроблена компанією Skype Technologies SA. Це однорангова мережа, у якій голосові виклики проходять через Інтернет, а не через мережу спеціального призначення. Користувачі Skype можуть шукати інших користувачів і надсилати їм повідомлення.

Skype повідомляє, що для зв'язку між клієнтами Skype використовується 256-бітне шифрування Advanced Encryption Standard (AES)/Rijnadel; хоча під час дзвінка на телефон або мобільний телефон частина дзвінка через комутовану телефонну мережу загального користування (PSTN) не шифрується. Відкриті ключі користувача сертифікуються сервером Skype під час входу за допомогою 1536-бітних або 2048-бітних сертифікатів RSA . Шифрування Skype є невід'ємною частиною протоколу Skype і є прозорим для абонентів. Деякі приватні розмови через Skype, наприклад аудіодзвінки, текстові повідомлення та надсилання файлів (зображень, аудіо чи відео), можуть використовувати наскрізне шифрування , але його, можливо, доведеться ввімкнути вручну.

Політика безпеки

У політиці безпеки компанії зазначено, що:

1. Імена користувачів унікальні.
2. Абоненти повинні надати ім'я користувача та пароль або інші облікові дані автентифікації.

3. Кожен абонент надає іншому підтвердження особи та привілеїв кожного разу, коли встановлюється сеанс. Кожен перевіряє докази іншого, перш ніж сеанс зможе передавати повідомлення.
4. Повідомлення, що передаються між користувачами Skype (без користувачів PSTN), шифруються від абонента до абонента. Жоден проміжний вузол (маршрутизатор) не має доступу до значення цих повідомлень. Це твердження було підірване в травні 2013 року доказами того, що Microsoft (власник Skype) пінгувала унікальні URL-адреси, вбудовані в розмову Skype; це може статися, лише якщо Microsoft має доступ до незашифрованої форми цих повідомлень.

Реєстрація

Skype зберігає реєстраційну інформацію як на комп'ютері абонента, так і на сервері Skype. Skype використовує цю інформацію для автентифікації одержувачів дзвінків і гарантує, що абоненти, які бажають автентифікації, мають доступ до сервера Skype, а не до самозванця. Skype каже, що для цього використовує шифрування з відкритим ключем , як визначено RSA .

Сервер Skype має закритий ключ і розповсюджує відкритий аналог цього ключа з кожною копією програмного забезпечення. Під час реєстрації користувача користувач вибирає бажане ім'я користувача та пароль. Skype локально генерує відкритий і закритий ключі. Закритий ключ і хеш пароля зберігаються на комп'ютері користувача.

Потім із сервером Skype встановлюється сеанс із 256-бітним шифруванням AES . Клієнт створює ключ сеансу за допомогою свого генератора випадкових чисел .

Сервер Skype перевіряє, чи вибране ім'я користувача є унікальним і відповідає правилам іменування Skype. Сервер зберігає ім'я користувача та хеш пароля користувача [H (H (P))] у своїй базі даних.

Тепер сервер формує та підписує сертифікат ідентифікації для імені користувача, який пов'язує ім'я користувача, ключ перевірки та ідентифікатор ключа.

Однорангова угода про ключ

Для кожного виклику Skype створює сеанс із 256-бітним ключем сеансу. Цей сеанс існує доти, доки триває спілкування, і протягом фіксованого часу після цього. Skype безпечно передає сеансовий ключ одержувачу дзвінка під час підключення дзвінка. Потім цей сеансовий ключ використовується для шифрування повідомлень в обох напрямках.

Криптографія сеансу

Весь трафік у сеансі шифрується за допомогою алгоритму AES, що працює в режимі цілочисельного лічильника (ICM). Skype шифрує поточний лічильник і сіль за допомогою ключа сеансу за допомогою 256-бітного алгоритму AES . Цей алгоритм повертає потік ключів, а потім обробляє XOR із вмістом повідомлення. Сесії Skype містять кілька потоків. Лічильник ICM залежить від потоку та розташування в потоці.

Генерація випадкових чисел

Skype використовує випадкові числа для кількох криптографічних цілей. Цілі включають захист від атак на відтворення, створення пар ключів RSA та створення половин ключів AES для шифрування вмісту. Безпека однорангового сеансу Skype значною мірою залежить від якості випадкових чисел, згенерованих обома кінцями сеансу Skype. Генерація випадкових чисел залежить від операційної системи.

Криптографічні примітиви

Skype використовує стандартні криптографічні примітиви для досягнення своїх цілей безпеки. Криптографічні примітиви, які використовуються в Skype, це блоковий шифр AES, криптосистема з відкритим ключем RSA, схема заповнення підпису ISO 9796-2, хеш-функція SHA-1 і потоковий шифр RC4 .

Протокол узгодження ключів

Ключове узгодження досягається за допомогою власного симетричного протоколу. Для захисту від атаки відтворення однорангові комп’ютери кидають виклик один одному випадковими 64-розрядними одноразовими номерами . Відповідь на виклик полягає в тому, щоб налаштувати виклик у приватний спосіб і повернути його, підписаного особистим ключем відповідача.

Однорангові вузли обмінюються сертифікатами ідентифікації та підтверджують, що ці сертифікати є законними. Оскільки сертифікат ідентифікації містить відкритий ключ, кожна сторона може підтверджувати підписи, створені іншим вузлом. Кожен вузол додає 128 випадкових бітів до 256-бітового ключа сеансу.

Автоматичне оновлення

Ще одним ризиком для безпеки є автоматичні оновлення, які не можна вимкнути , починаючи з версії 5.6, як у гілках Mac OS, так і в Windows, хоча в останній, і лише з версії 5.9, автоматичне оновлення можна вимкнути в певних випадків.

Прослуховування за схемою

Правоохоронні органи Китаю, Росії та США мають можливість підслуховувати розмови Skype і мати доступ до географічного розташування користувачів Skype. У багатьох випадках достатньо простого запиту на інформацію, без дозволу суду. Цю можливість навмисно додала корпорація Майкрософт для правоохоронних органів у всьому світі після того, як вони придбали Skype у 2011 році. Це реалізовано шляхом перемикання клієнта Skype для певного облікового запису користувача з шифрування на стороні клієнта на шифрування на стороні сервера, дозволяючи розповсюджувати незашифрований потік даних.

Фактичні та потенційні недоліки

У той час як Skype шифрує сеанси користувачів, інший трафік, включно з ініціюванням дзвінків, може контролюватися неавторизованими сторонами.

Інша сторона безпеки полягає в тому, чи створює Skype ризик для комп’ютерів і мереж користувачів. У жовтні 2005 року було виявлено та виправлено пару недоліків безпеки. Ці недоліки дозволили хакерам запустити ворожий код на комп’ютерах із уразливими версіями Skype. Перша помилка безпеки торкнулася лише комп’ютерів Microsoft Windows . Це дозволяло зловмиснику використовувати переповнення буфера для збою системи або змусити її виконати довільний код. Зловмисник може надати неправильну URL-адресу за допомогою формату URI Skype і спонукати користувача попросити його виконати атаку. Друга помилка безпеки вплинула на всі платформи; він використовував переповнення буфера на основі купи , щоб зробити систему вразливою.

Проблеми, зокрема кілька потенційно впливаючих на безпеку, включають:

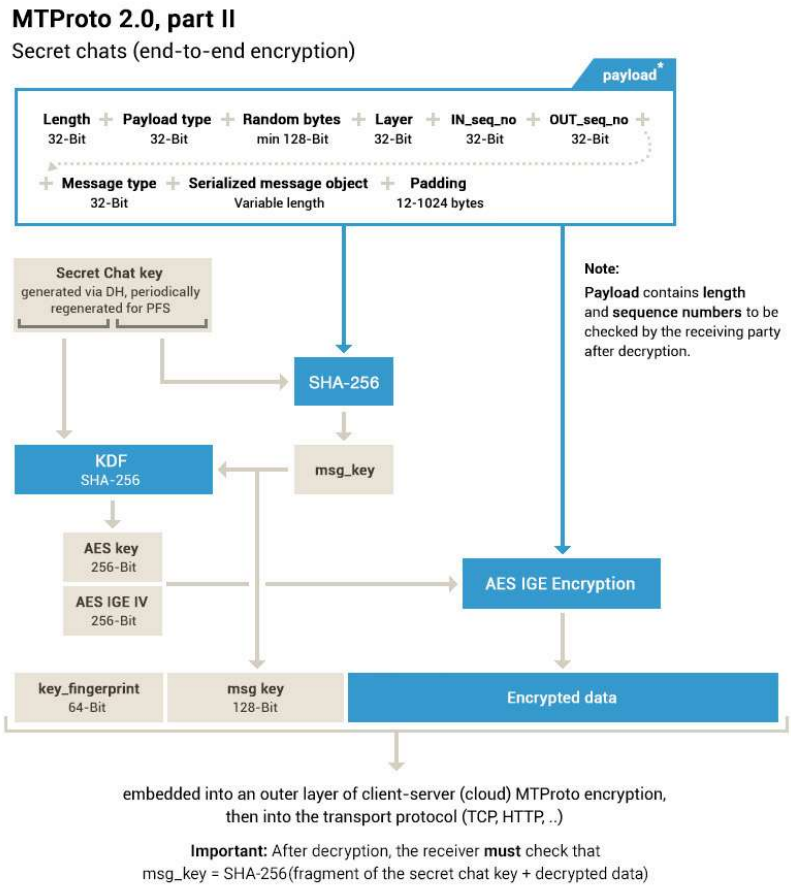
- За словами Нікласа Зеннстрьома, співзасновника Skype , який у 2004 році відповів на питання про модель безпеки Skype, сказавши: «Ми могли б це зробити, але лише якщо ми переробили те, як це працює, і зараз у нас немає часу». Якби джерело програмного забезпечення було доступним, експертна перевірка могла б перевірити його безпеку.
- 13 листопада 2012 року російський користувач опублікував недолік у безпеці Skype, який дозволяв будь-якому непрофесійному зловмиснику заволодіти обліковим записом Skype, знаючи лише електронну адресу жертви за сім простих кроків. Стверджувалося, що ця вразливість існує протягом місяців, і її не було виправлено до 12 годин після широкого оприлюднення.
- Той факт, що Skype записує дані про дзвінки (але не вміст повідомлення) у файл «Історія», який зберігається на комп'ютері користувача. Зловмисники, які отримали доступ до комп'ютера, можуть отримати файл.
- Skype може використовувати пропускну здатність інших користувачів. Хоча це задокументовано в ліцензійній угоді (EULA), неможливо визначити, яка пропускна здатність використовується таким чином.
- Існує близько 20 000 супервузлів із багатьох мільйонів користувачів, які ввійшли в систему. Посібник Skype для мережевих адміністраторів стверджує, що супервузли передають лише контрольний трафік до 10 КБ /с, а ретранслятори можуть передавати інший трафік даних користувача до 15 КБ/с (для однієї аудіоконференції). Зазвичай ретранслятор не повинен працювати з більш ніж одним "ретрансльованим з'єднанням".
- Функція передачі файлів Skype не інтегрується з жодними антивірусними продуктами , хоча Skype стверджує, що перевірив свій продукт на антивірусних продуктах «Shield».
- Skype не документує всю комунікаційну діяльність. Ця відсутність ясності щодо вмісту означає, що системні адміністратори не можуть бути впевнені, що він робить. (Комбінація запрошеного та реверсивного дослідження в сукупності свідчить про те, що Skype не робить нічого ворожого). Skype можна легко заблокувати брандмауерами .
- Skype споживає пропускну здатність мережі, навіть коли він неактивний (навіть для несупервузлів, наприклад, для обходу NAT). Наприклад, якби у світі було лише 3 користувача Skype і 2 спілкувалися, третій комп'ютер оподатковувався б для підтримки програми, навіть якщо він не використовував Skype у той час. Велика кількість комп'ютерів Skype означає, що ця діяльність є дифузною, може призвести до проблем із продуктивністю користувачів Skype у режимі очікування та є каналом для порушень безпеки.
- Skype неявно довіряє будь-якому потоку повідомлень, який підкоряється його протоколам.
- Skype не забороняє паралельну мережу, схожу на Skype.

- Skype до версії 3.0.0.216 створив файл під назвою 1.com у тимчасовому каталозі, який міг читати всі дані BIOS з ПК. За даними Skype, це використовувалося для ідентифікації комп'ютерів і забезпечення захисту DRM для плагінів. Пізніше вони видалили цей файл, але невідомо, чи було видалено поведінку читання BIOS.
- Обробник URI, який перевіряє URL-адреси для перевірки певних розширень файлів і форматів файлів, використовує методи порівняння з урахуванням регістру та не перевіряє всі потенційні формати файлів.
- Хоча Skype дійсно шифрує більшість своїх комунікацій, незашифровані пакети, що містять рекламу, витягуються з кількох місць, виявляючи вразливість міжсайтового сценарію . Цю рекламу можна легко зламати та замінити шкідливими даними.
- Конфіденційність трафіку Skype може мати обмеження. Хоча Skype шифрує спілкування між користувачами, представник Skype не заперечував можливості компанії перехоплювати спілкування. На питання про те, чи міг Skype прослуховувати спілкування своїх користувачів, Курт Зауер, глава відділу безпеки Skype, відповів ухильно: «Ми надаємо безпечні засоби зв'язку. Я не скажу, прослуховуємо ми чи ні. " У Китаї текст фільтрується відповідно до державних вимог. Це свідчить про те, що Skype має можливість підслуховувати з'єднання. Один із міноритарних власників Skype, eBay , розкрив інформацію про користувачів уряду США.
- Дослідники безпеки Біонді та Дескло припустили, що у Skype може бути чорні двері , оскільки Skype надсилає трафік, навіть коли він вимкнений, і тому, що Skype вжив екстремальних заходів, щоб приховати їхній трафік і функціонування своєї програми.
- Кілька джерел ЗМІ повідомили, що на нараді щодо «Законного перехоплення IP-сервісів», яка відбулася 25 червня 2008 року, високопоставлені, але неназвані чиновники міністерства внутрішніх справ Австрії заявили, що вони можуть без проблем прослуховувати розмови Skype. Австрійська служба громадського мовлення ORF , посилаючись на протокол зустрічі, повідомила, що "австрійська поліція може прослуховувати з'єднання Skype". Skype відмовився коментувати повідомлення.
- Помічено, що клієнт Skype для Linux звертається до папки профілю Firefox під час виконання. Ця папка містить усі збережені паролі у вигляді звичайного тексту, якщо не використовується головний пароль, а також історію веб-перегляду користувача. Доступ до цього файлу було підтверджено шляхом відстеження системних викликів, зроблених двійковим файлом Skype під час виконання.
- Помічено, що клієнт Skype для Mac отримує доступ до захищеної інформації в системній адресній книзі, навіть якщо інтеграцію з адресною книгою (увімкнено за замовчуванням) вимкнено в налаштуваннях Skype. Користувачі можуть бачити попередження про спробу Skype.app отримати доступ до захищеної інформації в адресній книзі за певних умов, наприклад, запуск Skype під час синхронізації з мобільним пристроєм.

- Skype не має законних підстав для доступу до адресної книги, якщо інтеграцію не ввімкнено. Крім того, ступінь інтеграції полягає в тому, щоб додати всі картки з адресної книги до списку контактів Skype разом із їхніми номерами телефонів, що можна здійснити без доступу до будь-якої захищеної інформації (ні ім'я, ні номери на картках не захищені) і, таким чином, спроба отримати доступ до інформації, що виходить за межі інтеграції, незалежно від того, увімкнено цю інтеграцію чи ні, викликає глибші питання щодо можливого шпигунства за користувачами.
- Федеральна комісія зі зв'язку Сполучених Штатів (FCC) витлумачила Закон про допомогу правоохоронним органам у зв'язку (CALEA) як вимогу до цифрових телефонних мереж дозволяти прослуховування телефонних розмов , якщо це дозволено на підставі ордеру ФБР, так само, як і інші телефонні служби.
- У лютому 2009 року Skype заявив, що, не будучи телефонною компанією, яка володіє телефонними лініями, на неї не поширюється дія CALEA та подібних законів, які регулюють діяльність телефонних компаній США. Також не зрозуміло, чи технічно можливо прослуховування спілкування в Skype. Відповідно до ACLU , Закон суперечить початковій меті Четвертої поправки до Конституції США ; Нещодавно ACLU висловив занепокоєння тим, що тлумачення Закону FCC є неправильним.

Telegram

Секретні чати — це бесіди один на один, у яких повідомлення шифруються ключем, яким володіють лише учасники чату. Зверніть увагу, що схема для цих наскрізних зашифрованих секретних чатів відрізняється від тієї, що використовується для хмарних чатів :



Примітка щодо MTProto 2.0

У цій статті описано рівень наскрізного шифрування в протоколі MTProto версії 2.0 .

Основні відмінності від версії 1.0 (описано тут для довідки) такі:

- SHA-256 використовується замість SHA-1;
- Байти заповнення беруть участь у обчисленні msg_key;
- msg_key залежить не лише від повідомлення, яке потрібно зашифрувати, але й від частини секретного ключа чату;
- 12..1024 байтів заповнення використовуються замість 0..15 байтів заповнення у v.1.0.

Генерація ключів

Ключі генеруються за допомогою протоколу Діффі-Хеллмана .

Давайте розглянемо наступний сценарій: користувач **A** хоче ініціювати наскрізне зашифроване спілкування з користувачем **B**.

Надсилання запиту

Користувач **A** виконує messages.getDhConfig , щоб отримати параметри Діффі-Хеллмана: простий **p** і старший елемент **g** .

Виконання цього методу перед кожною новою процедурою генерації ключа є життєво важливим. Має сенс кешувати значення параметрів разом із версією, щоб уникнути необхідності кожного разу отримувати всі значення. Якщо версія, збережена на клієнті, все ще актуальна, сервер поверне конструктор messages.dhConfigNotModified .

Очікується, що клієнт перевірить, чи **p** є безпечним 2048-бітним простим числом (це означає, що і **p** , і **(p-1)/2** є простими, і що 2 2047 < p < 2 2048), і що **g** генерує циклічну підгрупу простий порядок **(p-1)/2** , тобто є квадратичним залишком **mod p** . Оскільки **g** завжди дорівнює 2, 3, 4, 5, 6 або 7, це легко зробити, використовуючи квадратичний закон взаємності, що дає просту умову на **p mod 4g** — а саме, **p mod 8 = 7** для **g = 2** ; **p mod 3 = 2** для **g = 3** ; немає додаткової умови для **g = 4** ; **p mod 5 = 1 або 4** для **g = 5** ; **p mod 24 = 19 або 23** для **g = 6** ; і **p mod 7 = 3, 5 або 6** для **g = 7** . Після того, як **g** і **p** були перевірені клієнтом, має сенс кешувати результат, щоб уникнути повторення тривалих обчислень у майбутньому. Цей кеш може використовуватися для генерації ключа авторизації .

Якщо клієнту потрібна додаткова ентропія для генератора випадкових чисел, він може передати параметр **random_length** (random_length> 0), щоб сервер генерував власну випадкову послідовність **відповідної** довжини.
Важливо : використання випадкової послідовності сервера в необробленому вигляді може бути небезпечним, її потрібно поєднувати з послідовністю клієнта.

Клієнт **A** обчислює 2048-бітне число **a (використовуючи достатню ентропію або випадковий параметр** сервера ; див. вище) і виконує messages.requestEncryption після передачі g_a := pow(g, a) mod dh_prime.

Користувач **B** отримує оновлення updateEncryption для всіх пов’язаних ключів авторизації (усіх авторизованих пристроїв) за допомогою конструктора чату encryptedChatRequested . Користувачеві має бути показана основна інформація про користувача **A** та запропоновано прийняти або відхилити запит.

Обидва клієнти мають перевірити, що **g** , **g_a** та **g_b** більші за одиницю та менші за **p-1** . Ми також рекомендуємо перевірити, чи **g_a** і **g_b** знаходяться між **2^{2048-64}** і **p - 2^{2048-64}**

Прийняття запиту

Після того, як користувач **B** підтвердить створення секретного чату з **A** в інтерфейсі клієнта, клієнт **B** також отримує оновлені параметри конфігурації для методу Діффі-Хеллмана. Після цього він генерує випадкове 2048-бітне число **b** , використовуючи правила, подібні до тих, що застосовуються для **a** .

Отримавши **g_a** від оновлення з encryptedChatRequested , він може негайно згенерувати остаточний спільний ключ: key = (pow(g_a, b) mod dh_prime). Якщо довжина ключа < 256 байт, додайте кілька початкових нульових байтів як заповнення, щоб ключ мав рівно 256 байт. Його відбиток, **key_fingerprint** , дорівнює останнім 64 бітам SHA1 (ключа).

Примітка 1: у цьому конкретному випадку SHA1 використовується тут навіть для секретних чатів MTProto 2.0.

Примітка 2: цей відбиток використовується як перевірка працездатності для процедури обміну ключами для виявлення помилок під час розробки клієнтського програмного забезпечення — він не пов'язаний із візуалізацією ключів, що використовується на клієнтах як засіб зовнішньої автентифікації в секретних чатах. Ключові візуалізації на клієнтах генеруються з використанням перших 128 біт SHA1 (початковий ключ), а потім перших 160 біт SHA256 (ключ, який використовувався під час оновлення секретного чату до рівня 46).

Клієнт **B** виконує messages.acceptEncryption після передачі його g_b := pow(g, b) mod dh_primeта **key_fingerprint** .

Для всіх авторизованих пристроїв клієнта **Б** , крім поточного, оновлення updateEncryption надсилаються з конструктором encryptedChatDiscarded . Після цього єдиним пристроєм, який матиме доступ до секретного чату, є пристрій **B** , який зробив виклик messages.acceptEncryption .

Користувачеві **A** буде надіслано оновлення updateEncryption з конструктором encryptedChat для ключа авторизації, який ініціював чат.

За допомогою **g_b** з оновлення клієнт **A** також може обчислити спільний ключ key = (pow(g_b, a) mod dh_prime). Якщо довжина ключа < 256 байт, додайте кілька початкових нульових байтів як заповнення, щоб ключ мав рівно 256 байт. Якщо відбиток отриманого ключа ідентичний тому, який було передано encryptedChat , вхідні повідомлення можна надсилати та обробляти. В іншому випадку необхідно виконати messages.discardEncryption і повідомити користувача.

Ідеальна передня секретність

Щоб зберегти минуле спілкування в безпеці, офіційні клієнти Telegram ініціюють повторне введення ключа, коли ключ було використано для дешифрування та шифрування понад 100 повідомлень або він використовувався більше одного тижня, якщо ключ використовувався для шифрування. хоча б одне повідомлення. Потім старі ключі надійно викидаються, і їх неможливо відновити, навіть якщо є доступ до нових ключів, які зараз використовуються.

Протокол повторного введення ключів докладніше описано в цій статті: Ідеальна конфіденційність у секретних чатах .

Зауважте, що ваш клієнт має підтримувати Forward Secrecy у секретних чатах, щоб бути сумісним з офіційними клієнтами Telegram.

Надсилання та отримання повідомлень у секретному чаті. Серіалізація та шифрування вихідних повідомлень.

Створюється об’єкт TL типу DecryptedMessage , який містить повідомлення у вигляді звичайного тексту. Для зворотної сумісності об’єкт має бути загорнутий у конструктор decryptedMessageLayer із зазначенням підтримуваного рівня (починаючи з 46).

Схема TL для вмісту повідомлень із наскрізним шифруванням доступна тут »

Отримана конструкція серіалізується як масив байтів за допомогою загальних правил TL. До результуючого масиву додається 4 байти, що містять довжину масиву без урахування цих 4 байтів.

Масив байтів доповнюється від 12 до 1024 випадкових байтів доповнення, щоб його довжина ділилася на 16 байтів. (У старішому шифруванні MTProto 1.0 використовувалося лише від 0 до 15 байтів заповнення.)

Ключ повідомлення, **msg_key** , обчислюється як 128 середніх бітів SHA256 даних, отриманих на попередньому кроці, до яких додається 32 байти з ключа спільного **ключа** . (Для старішої версії шифрування MTProto 1.0 **msg_key** обчислювався по-іншому, оскільки 128 молодших бітів SHA1 даних, отриманих на попередніх кроках, за винятком байтів заповнення .)

Для MTProto 2.0 ключ AES **aes_key** і вектор ініціалізації **aes_iv** обчислюються (**ключ** — спільний ключ, отриманий під час генерації ключа) таким чином:

- msg_key_large = SHA256 (substr (ключ, 88+x, 32) + відкритий текст + випадкове_доповнення);
- msg_key = substr (msg_key_large, 8, 16);
- sha256_a = SHA256 (msg_key + substr (ключ, x, 36));
- sha256_b = SHA256 (substr (ключ, 40+x, 36) + msg_key);
- aes_key = substr (sha256_a, 0, 8) + substr (sha256_b, 8, 16) + substr (sha256_a, 24, 8);
- aes_iv = substr (sha256_b, 0, 8) + substr (sha256_a, 8, 16) + substr (sha256_b, 24, 8);

Для MTProto 2.0 **x=0** для повідомлень від автора секретного чату, **x=8** для повідомлень у протилежному напрямку.

Для застарілої версії MTProto 1.0 msg_key, aes_key і aes_iv обчислювалися інакше (див. цей документ для довідки).

Дані шифруються за допомогою 256-бітного ключа **aes_key** та 256-бітного вектора ініціалізації **aes-iv** за допомогою шифрування AES-256 із нескінченним розширенням спотворення (IGE). Відбиток ключа шифрування **key_fingerprint** і

ключ повідомлення **msg_key** додаються у верхній частині результуючого масиву байтів.

Зашифровані дані вбудовуються в виклик API messages.sendEncrypted і передаються на сервер Telegram для доставки іншій стороні секретного чату.

Оновлення до MTProto 2.0 з MTProto 1.0

Якщо обидві сторони в секретному чаті використовують принаймні рівень 73, вони повинні використовувати лише MTProto 2.0 для всіх вихідних повідомлень. Деякі з перших отриманих повідомлень можуть використовувати MTProto 1.0, якщо під час створення секретного чату не було узгоджено достатньо високий початковий рівень. Після отримання першого повідомлення, зашифрованого за допомогою MTProto 2.0 (або першого повідомлення з рівнем 73 або вище), усі повідомлення з вищими порядковими номерами також мають бути зашифровані за допомогою MTProto 2.0.

Поки поточний рівень нижчий за 73, кожна сторона має спробувати розшифрувати отримані повідомлення за допомогою MTProto 1.0, і якщо це не вдасться (msg_key не збігається), спробуйте MTProto 2.0. Коли надійде перше повідомлення, зашифроване MTProto 2.0 (або рівень оновлено до 73), немає потреби пробувати розшифровку MTProto 1.0 для будь-яких наступних повідомлень (якщо клієнт все ще не чекає на закриття деяких прогалин).

Розшифровка вхідного повідомлення

Наведені вище кроки виконуються у зворотному порядку.
Коли отримано зашифроване повідомлення, ви **повинні** переконатися, що msg_key **насправді** дорівнює 128 середнім бітам хешу SHA256 розшифрованого повідомлення, до якого додаються 32 байти, взяті зі спільного **ключа** .
Якщо рівень повідомлень більший за той, який підтримується клієнтом, користувач повинен отримати сповіщення про те, що версія клієнта застаріла, і запропонувати оновити.

Порядкові номери

Необхідно інтерпретувати всі повідомлення в їх початковому порядку, щоб захистити від можливих маніпуляцій. Секретні чати підтримують спеціальний механізм для обробки лічильників seq_no незалежно від сервера.

> Правильне поводження з цими лічильниками описано далі в цій статті: Порядкові номери в секретних чатах .

Зверніть увагу, що ваш клієнт повинен підтримувати порядкові номери в секретних чатах, щоб бути сумісним з офіційними клієнтами Telegram.

Надсилання зашифрованих файлів

сі файли, надіслані в секретні чати, зашифровані одноразовими ключами, які жодним чином не пов’язані зі спільним ключем чату. Перед тим, як зашифрований файл буде надіслано, передбачається, що адреса зашифрованого файлу буде додана до зовнішньої частини зашифрованого повідомлення за допомогою параметра file **методу** messages.sendEncryptedFile і що ключ для прямого розшифрування буде надіслано в тілі повідомлення. повідомлення (**ключовий** параметр у конструкторах decryptedMessageMediaPhoto , decryptedMessageMediaVideo та de cryptedMessageMediaFile .

Перед тим, як файл буде надіслано в секретний чат, обчислюються 2 випадкових 256-бітних числа, які слугуватимуть ключем AES і вектором ініціалізації для шифрування файлу. Подібним чином використовується шифрування AES-256 із нескінченним розширенням спотворення (IGE).

Відбиток ключа обчислюється наступним чином:

- дайджест = md5(ключ + iv)
- відбиток = substr(дайджест, 0, 4) XOR substr(дайджест, 4, 4)

Зашифрований вміст файлу зберігається на сервері майже так само, як вміст файлу в хмарних чатах : частина за частиною за допомогою викликів upload.saveFilePart .
Подальший виклик messages.sendEncryptedFile призначить ідентифікатор збереженому файлу та надішле адресу разом із повідомленням. Одержувач отримає оновлення з encryptedMessage , а параметр **file** міститиме інформацію про файл.

Вхідні та вихідні зашифровані файли можна пересилати в інші секретні чати за допомогою конструктора inputEncryptedFile , щоб уникнути подвійного збереження одного й того самого вмісту на сервері.

Робота з вікном оновлення

Секретні чати пов'язані з конкретними пристроями (точніше з ключами авторизації), а не з користувачами. Звичайне вікно повідомлень, яке використовує **pts** для опису статусу клієнта, не підходить, оскільки воно призначене для тривалого зберігання повідомлень і доступу до повідомлень з різних пристроїв.

Як вирішення цієї проблеми вводиться додаткова тимчасова черга повідомлень. Коли надсилається оновлення щодо повідомлення з секретного чату, надсилається нове значення **qts** , яке допомагає відновити різницю, якщо була довга перерва в з’єднанні або в разі втрати оновлення.

Зі збільшенням кількості подій значення **qts** збільшується на 1 із кожною новою подією. Початкове значення не може (і не буде) дорівнювати 0.

Той факт, що події з тимчасової черги були отримані та збережені клієнтом, підтверджується явно викликом методу messages.receivedQueue або неявно викликом updates.getDifference (передано значення **qts** , а не кінцевий стан). Усі повідомлення, підтверджені клієнтом як доставлені, а також будь-які повідомлення старше 7 днів можуть (і будуть) видалятися з сервера.

Після скасування авторизації чергу подій відповідного пристрою буде примусово очищена, а значення **qts** стане неактуальним.

Оновлення до нових шарів

Ваш клієнт повинен завжди зберігати максимальний рівень, який, як відомо, підтримується клієнтом з іншого боку секретного чату. Коли секретний чат створюється вперше, це значення має бути ініціалізовано рівним 46. Це значення віддаленого рівня завжди має оновлюватися одразу після отримання *будь-якого* пакета, що містить інформацію верхнього рівня, тобто:

- будь-яке секретне повідомлення чату, що містить *layer_no* у своєму decryptedMessageLayerз *шаром* >=46, або
- службове повідомлення decryptedMessageActionNotifyLayer , укладене так, ніби це конструктор decryptedMessageService застарілого рівня 8 (конструктор decryptedMessageService#aa48327d).

Повідомлення віддаленого клієнта про ваш локальний рівень

Щоб повідомити віддаленого клієнта про ваш локальний рівень, ваш клієнт повинен надіслати повідомлення типу decryptedMessageActionNotifyLayer. Це сповіщення має бути упаковане в конструктор відповідного шару.

Є два випадки, коли ваш клієнт повинен повідомити віддаленого клієнта про свій локальний рівень:

1. Як тільки буде створено новий секретний чат, одразу після успішного обміну секретним ключем.
2. Одразу після оновлення локального клієнта для підтримки нового секретного рівня чату. У цьому випадку сповіщення повинні бути надіслані в **усі** існуючі секретні чати. Зауважте, що це необхідно лише під час оновлення до нових рівнів, які містять зміни в реалізації секретних чатів (наприклад, вам не потрібно робити це, коли ваш клієнт оновлено з рівня 46 до рівня 47).

Голосові та відеодзвінки з наскрізним шифруванням

У цій статті описано наскрізне шифрування, яке використовується для **голосових** і **відеодзвінків** Telegram .

Встановлення дзвінків

Перш ніж дзвінок буде готовий, необхідно виконати деякі попередні дії. Сторона, що телефонує, повинна зв’язатися зі стороною, якій потрібно викликати, і перевірити, чи готова вона прийняти виклик. Крім того, сторони повинні узгодити протоколи, які будуть використовуватися, дізнатися IP-адреси одна одної або серверів ретрансляції Telegram, які будуть використовуватися (так звані рефлектори), і створити одноразовий ключ шифрування для цього голосового дзвінка за допомогою за допомогою обміну ключами Діффі--Хеллмана . Усе це виконується паралельно за допомогою кількох методів API Telegram і відповідних сповіщень. У цьому документі описано деталі, пов’язані з генерацією ключів, шифруванням і безпекою.

Генерація ключів

Обмін ключами Діффі-Хеллмана, а також весь протокол, який використовується для створення нового голосового виклику, дуже схожий на той, який використовується для секретних чатів . Ми рекомендуємо вивчити статтю за посиланням, перш ніж продовжити.

Однак ми внесли деякі важливі зміни, щоб полегшити процес перевірки ключа . Нижче наведено весь обмін між двома сторонами, що спілкуються, абонентом (A) і абонентом (B), через сервери Telegram (S).

- A виконує messages.getDhConfig , щоб знайти 2048-бітне просте число Діффі-Хеллмана p і генератор g . Очікується, що клієнт перевірить, чи є p безпечним простим числом, і виконає всі перевірки безпеки , необхідні для секретних чатів.
- A вибирає випадкове значення a , 1 < a < p-1, і обчислює g_a:=power(g,a) mod p (256-байтове число) і g_a_hash:=SHA256(g_a) (32 байти).
- A викликає (надсилає на сервер S) phone.requestCallg_a_hash:bytes , який , серед іншого, містить поле . Для цього виклику це поле має бути заповнене g_a_hash , **а не самим** g_a .
- Сервер S виконує перевірку конфіденційності та надсилає оновлення updatePhoneCall з конструктором phoneCallRequested на всі активні пристрої B. Це оновлення, окрім ідентичності A та інших відповідних параметрів, містить поле g_a_hash , заповнене значенням, отриманим з A .
- B приймає виклик на одному зі своїх пристроїв, зберігає отримане значення g_a_hash для цього екземпляра протоколу створення голосового виклику, вибирає випадкове значення b , 1 < b < p-1, обчислює g_b:=power(g,b) mod p , виконує всі необхідні перевірки безпеки та викликає метод phone.acceptCall , який має поле g_b:bytes (серед іншого), яке потрібно заповнити значенням самого g_b (а не його хеш-функцією).
- Сервер S надсилає updatePhoneCall з конструктором phoneCallDiscarded на всі інші пристрої, авторизовані B , щоб запобігти прийняттю того самого виклику на будь-якому з інших пристроїв. З цього моменту сервер S працює лише з тими пристроями B , які першими викликали phone.acceptCall .

- Сервер S надсилає на A оновлення updatePhoneCall з конструктором phoneCallAccepted , що містить значення g_b , отримане від B.
- A виконує всі звичайні перевірки безпеки g_b і a , обчислює ключ Діффі--Хеллмана key :=power(g_b,a) mod p і його відбиток key_fingerprint:long , що дорівнює молодшим 64 бітам SHA1(key) , так само, як і з секретними чатами. Потім A викликає метод phone.confirmCall , що містить g_a:bytesi key_fingerprint:long.
- Сервер S надсилає B оновлення updatePhoneCall з конструктором phoneCall , що містить значення g_a в полі g_a_or_b:bytes і key_fingerprint:long
- У цей момент B отримує значення g_a . Він перевіряє, що SHA256(g_a) справді дорівнює попередньо отриманому значенню g_a_hash , виконує всі звичайні перевірки безпеки Діффі-Хеллмана та обчислює ключ key :=power(g_a,b) mod p і його відбиток, рівний молодші 64 біти SHA1 (ключ) . Потім він перевіряє, чи цей відбиток дорівнює значенню, key_fingerprint:longотриманому від іншої сторони, як перевірку правильності реалізації.

На цьому обмін ключами Діффі--Хеллмана завершено, і обидві сторони мають спільний секретний ключ розміром 256 байт, який використовується для шифрування всіх подальших обмінів між A і B.

Надзвичайно важливо приймати кожне оновлення лише один раз для кожного екземпляра протоколу генерації ключів, відкидаючи будь-які дублікати або альтернативні версії вже отриманих і оброблених повідомлень (оновлень).

Шифрування

У цьому документі описано шифрування **голосових і відеодзвінків** , реалізоване в програмах Telegram версій **7.0** і вище. Перегляньте цей документ , щоб дізнатися більше про шифрування, яке використовується під час **голосових викликів** у версіях програми, випущених до **14 серпня 2020 року** .
Бібліотека голосових і відеодзвінків Telegram використовує оптимізовану версію MTProto 2.0 для надсилання та отримання **пакетів** , що складаються з одного або кількох наскрізних зашифрованих **повідомлень** різних типів (список кандидатів на льоду , формати відео, стан віддаленого відео, дані аудіопотоку , дані відеопотоку, підтвердження повідомлення або пусте).

Цей документ описує лише процес шифрування, оминаючи кодування та мережеві частини.

Бібліотека починає працювати з:

- Ключ шифрування key , спільний між сторонами, як згенеровано вище.
- **Інформація про вихідний** чи **вхідний** виклик .
- Два канали передачі даних: **сигналізація** , запропонована Telegram API, і **транспорт** на основі WebRTC.

Обидва канали передачі даних ненадійні (повідомлення можуть бути втрачені), але **сигналізація** повільніша та надійніша.

Шифрування даних виклику

Тіло пакета (decrypted_body) складається з кількох повідомлень та їхніх відповідних seqномерів, об’єднаних разом.

- decrypted_body = message_seq1 + message_body1 + message_seq2 + message_body2

Кожен decrypted_bodyунікальний, тому що жодні seqномери першого повідомлення не можуть бути однаковими. Якщо потрібно повторно надіслати лише старі повідомлення, до пакета спочатку додається порожнє повідомлення з новим унікальним повідомленням .seq

Ключ шифрування використовується keyдля обчислення 128-бітного msg_key, а потім 256- aes_keyі 128-бітного aes_iv:

- msg_key_large = SHA256 (substr(key, 88+x, 32) + decrypted_body);
- msg_key = substr (msg_key_large, 8, 16);
- sha256_a = SHA256 (msg_key + substr (ключ, x, 36));
- sha256_b = SHA256 (substr (ключ, 40+x, 36) + msg_key);
- aes_key = substr (sha256_a, 0, 8) + substr (sha256_b, 8, 16) + substr (sha256_a, 24, 8);
- aes_iv = substr (sha256_b, 0, 4) + substr (sha256_a, 8, 8) + substr (sha256_b, 24, 4);

x **залежить від того, вихідний** чи **вхідний** дзвінок, а також від типу з’єднання:

- x = 0 для **вихідних** + **транспортних**
- x = 8 для **вхідних** + **транспортних**
- x = 128 для **вихідних** + **сигналізація**
- x = 136 для **вхідної** + **сигналізації**

Це дозволяє додаткам вирішувати, які типи пакетів буде надіслано до яких з’єднань, і працювати в цих з’єднаннях незалежно (кожне має свій власний seqлічильник).

Отримані aes_keyта aes_ivвикористовуються для шифрування decrypted_body:

- encrypted_body = AES_CTR (decrypted_body, aes_key, aes_iv)

Пакет, який надсилається, складається msg_keyз encrypted_body:

- packet_bytes = msg_key + encrypted_body

Після отримання пакет розшифровується за допомогою keyта msg_key, після чого msg_keyперевіряється на відповідність

відповідному SHA256підрядку. Якщо перевірка не вдається, пакет **потрібно** відкинути.

Захист від атак Replay

Кожен з вузлів підтримує свій власний 32-розрядний монотонно зростаючий лічильник для вихідних повідомлень, seqпочинаючи з 1. Цей seqлічильник додається до кожного надісланого повідомлення та збільшується на 1для кожного нового повідомлення. Немає двох seqоднакових номерів першого повідомлення в пакеті. Якщо потрібно повторно надіслати лише старі повідомлення, до пакета спочатку додається порожнє повідомлення з новим унікальним . seqКоли seqлічильник досягає 2^30, виклик потрібно перервати. Кожен вузол зберігає seqзначення всіх отриманих (і оброблених) повідомлень, які перевищують max_received_seq - 64, де max_received_seqє найбільше seqотримане число.

Якщо отримано пакет, перше повідомлення якого має seqменше або дорівнює max_received_seq - 64або його seqвже було отримано, повідомлення відхиляється. В іншому випадку seqзначення всіх вхідних повідомлень запам'ятовуються та max_received_seqкоригуються. Це гарантує, що два пакети не будуть оброблені двічі.

Перевірка ключа

Щоб перевірити ключ і переконатися, що атака MITM не відбувається, обидві сторони об’єднують ключ секретного ключа зі значенням g_a абонента (A), обчислюють SHA256 і використовують його для створення послідовності смайлів. Точніше, хеш SHA256 розбивається на чотири 64-розрядні цілі числа; кожен з них ділиться на загальну кількість використовуваних смайлів (наразі 333), а залишок використовується для вибору конкретних смайлів. Специфіка протоколу гарантує, що порівняння чотирьох смайликів із набору з 333 є достатнім для запобігання прослуховування (атака MiTM на DH) з імовірністю **0,9999999999** .

Це тому, що замість стандартного обміну ключами Діффі-Хеллмана, який вимагає лише двох повідомлень між сторонами:

- A->B : (генерує a і) надсилає g_a := g^a
- B->A : (генерує b і справжній ключ (g_a)^b, потім) надсилає g_b := g^b
- A : обчислює ключ (g_b)^a

ми використовуємо його **модифікацію з трьома повідомленнями** , яка добре працює, коли обидві сторони онлайн (що також є вимогою для голосових дзвінків):

- A->B : (генерує a і) надсилає g_a_hash := hash(g^a)
- B->A : (зберігає g_a_hash, генерує b і) надсилає g_b := g^b
- A->B : (обчислює ключ (g_b)^a, потім) надсилає g_a := g^a
- B : перевіряє hash(g_a) == g_a_hash, потім обчислює ключ (g_a)^b

Ідея тут полягає в тому, що A бере на себе певне значення a (і g_a), не повідомляючи його B . B має вибрати своє значення b і g_b , не знаючи справжнього значення g_a , щоб він не міг спробувати різні значення b , щоб змусити кінцевий ключ (g_a)^b мати будь-які специфічні властивості (наприклад, фіксовані молодші 32 біти SHA256(ключ)). На цьому етапі B приймає певне значення g_b , не знаючи g_a . Тоді A має надіслати своє значення g_a ; він не може змінити його, навіть якщо зараз знає g_b , оскільки інша сторона B прийме лише значення g_a , яке має хеш, вказаний у першому повідомленні обміну.

Якщо якийсь самозванець прикидається або A , або Б і намагається здійснити атаку "Людина посередині" на цьому обміні ключами Діффі-Хеллмана, вищезазначене все ще актуально. Сторона A згенерує спільний ключ із B — або тим, хто прикидається B — не маючи другого шансу змінити свій експонент a залежно від значення g_b , отриманого від іншої сторони; і самозванець не матиме шансу адаптувати своє значення b залежно від g_a , тому що він повинен прийняти значення g_b перед вивченням g_a . Те саме стосується генерації ключа між самозванцем і стороною B.

Використання хеш-зобов’язання в обміні DH обмежує зловмисника лише **однією здогадкою** , щоб створити правильну візуалізацію в своїй атаці, що означає, що використання трохи більше 33 бітів ентропії, представленої чотирма емодзі у візуалізації, достатньо для успішної атаки. неймовірно.

Ідеальна передня секретність

Telegram підтримує Perfect Forward Secrecy (PFS).

Щоб зробити це можливим, клієнт генерує постійний ключ авторизації за допомогою **p_q_inner_data** та тимчасовий ключ за допомогою **p_q_inner_data_temp** . (Додаткову інформацію див. у розділі Створення ключа авторизації .) Ці 2 операції можна виконувати паралельно за допомогою різних з’єднань. Клієнт повинен зберегти позначку часу **expires_at** unix expires_at = time + expires_in.

Важливо : щоб досягти PFS, клієнт **ніколи не** повинен використовувати постійний auth_key_id безпосередньо. Кожне повідомлення, яке надсилається до MTProto, має бути зашифровано за допомогою **temp_auth_key_id** , який був прив’язаний до **perm_auth_key_id** .

Незв’язаний **temp_auth_key_id** можна використовувати лише з такими методами:

- auth.bindTempAuthKey
- help.getConfig
- help.getNearestDc

Щоб зв’язати тимчасовий ключ авторизації з постійним ключем, клієнт створює спеціальне повідомлення зв’язування та виконує метод auth.bindTempAuthKey за допомогою **temp_auth_key** . Після успішного виконання **auth.bindTempAuthKey клієнт може продовжувати використовувати API як зазвичай;** клієнт також

повинен переписати інформацію клієнта за допомогою initConnection після кожного зв'язування. Кожен постійний ключ може бути прив'язаний лише до **одного** тимчасового ключа за раз, зв'язування нового тимчасового ключа перезаписує попередній.

Після закінчення терміну дії тимчасового ключа клієнт повинен створити новий тимчасовий ключ за допомогою **p_q_inner_data_temp** . Потім йому потрібно повторно прив'язати цей новий тимчасовий ключ до початкового постійного ключа. Новий ключ також можна створити заздалегідь, щоб у клієнта був готовий новий ключ до того моменту, коли закінчиться термін дії старого.

Для додаткової безпеки клієнт може зберігати тимчасовий ключ авторизації лише в оперативній пам'яті та ніколи не зберігати його в постійному сховищі.

Термін дії тимчасового ключа авторизації може закінчитися в будь-який момент до **expires_at** , оскільки такі ключі також зберігаються лише в оперативній пам'яті на стороні сервера. Будьте готові правильно обробляти отримані помилки MTProto (неіснуючий auth_key_id призводить до помилки 404).

WhatsApp

https://arxiv.org/pdf/2209.11198.pdf

https://medium.com/@panghalamit/whatsapp-s-end-to-end-encryption-how-does-it-work-80020977caa0

https://vdocs.ro/download/whatsapp-security-whitepaper-v4-preview-3md8dk2dq3?hash=f5712a03d880ffedcb18c28843e1e081

https://www.ijsr.net/archive/v7i11/ART20192911.pdf

Введення

Цей білий документ містить технічне пояснення система наскрізного шифрування WhatsApp. WhatsApp Messenger дозволяє людям обмінюватися повідомлення (включаючи чати, групові чати, зображення, відео, голосові повідомлення та файли) і здійснювати дзвінки WhatsApp навколо світу. Повідомлення WhatsApp, голос і відео дзвінки між відправником і одержувачем, які використовують WhatsApp клієнтське програмне забезпечення, випущене після 31 березня 2016 року, кінець зашифровано. Протокол сигналу, розроблений Open Whisper Systems, є основою наскрізного шифрування WhatsApp. Це протокол наскрізного шифрування призначений для запобігання третім сторонам і WhatsApp від доступу до відкритого тексту на повідомлення або дзвінки. Більше того, навіть якщо ключі шифрування з пристрою користувача будь-коли фізично скомпрометовані їх не можна використовувати для повернення в минуле для розшифровки передані раніше повідомлення. Цей документ містить огляд протоколу сигналів і його використання в WhatsApp.

Терміни

a) Типи відкритих ключів

• Пара ключів ідентифікації - довгострокова пара ключів Curve25519, генерується під час встановлення.

• Підписаний попередній ключ - середньостроковий ключ Curve25519 пара, згенерована під час встановлення, підписана ідентифікатором ключ і періодично чергуються.

• Одноразові попередні ключі – черга з пар ключів Curve25519 для одноразового використання, згенерованого під час встановлення та поповнюється за потреби.

b) Типи ключів сеансу

• Кореневий ключ - 32-байтове значення, яке використовується для створення ланцюжка Ключі.

• Chain Key - 32-байтове значення, яке використовується для створення Ключі повідомлень.

• Ключ повідомлення – 80-байтове значення, яке використовується для шифрування вміст повідомлення. 32 байти використовуються для AES-256 ключа, 32 байти для ключа HMAC-SHA256 і 16 байтів для IV.

c) Реєстрація клієнта

Під час реєстрації клієнт WhatsApp передає свої публічний ідентифікаційний ключ, публічний підписаний попередній ключ (з його підпис) і пакет загальнодоступних одноразових попередніх ключів до серверу. Сервер WhatsApp зберігає ці відкриті ключі пов’язані з ідентифікатором користувача. Жодного разу сервер WhatsApp не має доступ до будь-якого приватного клієнтських ключів.

d) Початок налаштування сеансу

Щоб спілкуватися з іншим користувачем WhatsApp, а Клієнт WhatsApp спочатку повинен встановити зашифрований сесії. Після встановлення сеансу клієнти цього не роблять потрібно перебудувати новий сеанс один з одним,

доки існуючий стан сеансу втрачається через зовнішню подію, таку як перевстановлення програми або зміна пристрою.

Щоб встановити сеанс:

1. Клієнт-ініціатор («ініціатор») запитує публічний ключ ідентифікації, публічний підписаний попередній ключ і один відкритий одноразовий попередній ключ для одержувача.

2. Сервер повертає запитані значення відкритого ключа. Одноразовий попередній ключ використовується лише один раз, тому його видаляють із сховища сервера після запиту. Якщо остання партія одноразових попередніх ключів одержувача була спожиті, а одержувач не поповнив їх, одноразовий попередній ключ не повертається.

3. Ініціатор зберігає ідентифікаційний ключ одержувача як I-одержувача, підписаний попередній ключ як S-одержувача та одноразовий попередній ключ як O-одержувач.

4. Ініціатор генерує ефемерний ключ Curve25519 пара, E-ініціатор.

5. Ініціатор завантажує свій власний ключ ідентифікації як I-ініціатор.

5. Ініціатор обчислює master_secret як:

master_secret = ECDH (I-ініціатор, S-реципієнт) || ECDH (E-ініціатор, Я-реципієнт) || ECDH (E-ініціатор, S-реципієнт) || ECDH(E-ініціатор, O-реципієнт).

Якщо немає одноразової попереднього ключа, остаточний(завершальний) ECDH опущено.

6. Ініціатор використовує HKDF для створення кореневого ключа таланцюжок ключів від master_secret.

е) Налаштування сеансу отримання

Після створення тривалого сеансу шифрування, ініціатор може негайно розпочати надсилання повідомлень на одержувача, навіть якщо одержувач

офлайн. До одержувач відповідає, ініціатор включає інформацію (у заголовку всіх надісланих повідомлень), що одержувач вимагає створення відповідного сеансу. Це включає E-ініціатор та I-ініціатор ініціатора.

Коли одержувач отримує повідомлення, яке містить інформація про налаштування сесії:

1. Одержувач розраховує відповідний master_secret, використовуючи власні приватні та публічні ключі ключі, рекламовані в заголовку вхідного повідомлення.

2. Одержувач видаляє одноразовий попередній ключ, який використовує ініціатор.

2. Ініціатор використовує HKDF для отримання відповідного кореня ключ і ланцюжок ключів від master_secret.

f) Обмін повідомленнями

Після встановлення сеансу клієнти обмінюються повідомлення, захищені за допомогою ключа повідомлення AES256 у режимі CbC для шифрування та HMAC- SHA256 для автентифікації. Ключ повідомлення змінюється для кожного переданого повідомлення, і є ефемерним; таким, яким раніше був ключ повідомлення зашифрувати повідомлення неможливо відновити з стан сесії після передачі повідомлення або отримано. Ключ повідомлення походить від ланцюгового ключа відправника що «храпово» вперед із кожним надісланим повідомленням. Крім того, укладається новий договір ECDH з кожного повідомлення, щоб створити новий ключ ланцюга. Це забезпечує пряму секретність завдяки поєднанню обох негайний «храповий тріск» і поїздка туди й назад «DH тріскачка».

g) Обчислення ключа повідомлення з ключа ланцюга

Кожного разу, коли повідомлення потребує нового ключа повідомлення відправника, обчислюється як:

1. Ключ повідомлення = HMAC-SHA256 (ключ ланцюга, 0x01).

2. Потім ключ ланцюга оновлюється як ключ ланцюга HMAG-SHA256 (ключ ланцюжка, 0x02).

Це призводить до того, що ключ ланцюга «храповий» рухається вперед і інше означає, що збережений ключ повідомлення не можна використовувати для отримання поточні або минулі значення ключа ланцюга.

h) Обчислення ланцюгового ключа з кореневого ключа

Щоразу, коли передається повідомлення, ефемера відкритий ключ Curve25519 рекламується разом із ним. Один раз отримано відповідь, новий ключ ланцюга та ключ кореня розраховується як:

1. ephemeral_secret= ECDH (Ephemeral-sender, Ефемерний-реципієнт).

2.Ланцюговий ключ, кореневий ключ = HKDF (кореневий ключ ефемерний_секрет).

Ланцюжок використовується лише для надсилання повідомлень від одного користувача тому ключі повідомлень не використовуються повторно. Через шлях ключі повідомлень і ключі ланцюга обчислюються, повідомленняможе прибути із затримкою, вийти з ладу або може бути повністю втрачено без проблем.

i) Перевірка ключів

Користувачі WhatsApp додатково мають можливість перевірити ключі інших користувачів, з якими вони спілкування, щоб вони могли підтвердити, що неавторизована третя сторона (або WhatsApp) не ініціювала а атака "людина посередині". Це можна зробити шляхом сканування а QR-кодом або порівнянням 60-значного числа.

QR-код містить:

1. Версія.
2. Ідентифікатор користувача для обох сторін.
3. Повний 32-байтовий відкритий ключ ідентифікації для обох сторін.

Коли будь-який користувач сканує QR-код іншого, ключі з’являються порівняти, щоб переконатися, що QR-код збігається ключ ідентифікації, отриманий із

сервера. 60-значне число обчислюється шляхом об’єднання два 30-значні цифрові відбитки пальців для ідентифікації кожного користувача ключ. Щоб обчислити 30-значний цифровий відбиток:

1. Ітеративно SHA-512 хешує відкритий ключ ідентифікації та ідентифікатор користувача 5200 разів.
2. Візьміть перші 30 байт остаточного результату хешу.
3. Розділіть 30-байтовий результат на шість 5-байтових фрагментів.
4. Перетворіть кожен 5-байтовий фрагмент у 5 цифр шляхом інтерпретації кожен 5-байтовий фрагмент як ціле число без знаку з порядковим кінцем і зменшивши його по модулю 100000.
5. Об’єднайте шість груп із п’яти цифр у тридцять цифри.

j) Мій відкритий ключ

Мій відкритий ключ: 76057 19600 80014 70022 03552 86012

k) Видобути приватний ключ

Тут ми опишемо, як розшифрувати резервну копію WhatsApp з шифруванням crypt7. Щоб розшифрувати crypt7 WhatsApp резервної бази даних, нам знадобиться ключ шифру. Щоб отримати ключ, пристрій спочатку має бути рутовано. Ключ знаходиться за адресою path/data/data/files/com.whatsapp/key. Якщо у нас є а пристрій, який не рутований, і ми вирішили цього не робити (цифровий найкращі практики криміналістики!), тут буде описано, як за допомогою програма, спочатку розроблена TripCode і оновлено AbinashBishoyi, називається WhatsApp-Key-DB-Екстрактор.

WhatsApp-Key-DB-Extractor

Метою цього сценарію є надання методу для користувачів WhatsApp мають видобувати свій ключ шифру на не рутовані пристрої Android. Цей сценарій також роздобуде остання незашифрована база даних повідомлень WhatsApp (msgstore.db) і база даних контактів (wa.db).

Передумови:

• Mindie Vine або Andris версії 40 або вище

• Мобільний телефон із версією програми WhatsApp створити файли резервної копії за допомогою crypt6 / crypt7 / crypt 8 шифрувань. Зовнішня карта SD не потрібна.

Вилучення кроків

Крок 1. Встановіть Java. Якщо ні, ви можете отримати його безкоштовно сайт завантажити Java.

Крок 2. Установіть драйвери Android Debug Bridge (ADB). Якщо ні, він є у вільному доступі на цьому веб-сайті ADB Installer. Після інсталяції ADB рекомендується мати підключення до Інтернету ввімкнено. Під час встановлення вони можуть потрібно отримати щось з інших сайтів.

Крок 3. Увімкніть USB на своєму мобільному телефоні режим налагодження можна знайти в параметрах розробника.

Крок 4. Завантажте та розпакуйте WhatsAppKeyExtract.zip вашого комп'ютера, зберігаючи структуру каталогів. Розпакувати його на робочий стіл було б досить добре.

Крок 5. Після того, як WhatsAppKeyExtract.zip буде успішно видобуто, перейдіть до папки та запустіть a bat під назвою WhatsAppKeyExtract.bat.

Крок 6. Підключіть мобільний пристрій до робочої станції.

Дочекайтеся встановлення всіх необхідних драйверів. Переконайтеся в усіх встановлені драйвери. Невдача на цьому етапі спричинить ваші мобільний телефон не виявлено WhatsAppKeyExtract.bat.

Крок 7. Розблокуйте заставку мобільного телефону (якщо є) і зачекайте, доки на екрані не з'явиться «Повна резервна копія» пристрою.

Крок 8: Залиште поле пароля порожнім і натисніть «Створити резервну копію моїх даних». Зачекайте деякий час, поки у вікні WhatsAppKeyExtract написано "Готово!".

Крок 9. Перегляньте папку WhatsAppKeyExtract, знайдіть папку під назвою витягнуто.

Крок 10: У розпакованій папці ви знайдете 3 створених файлів Key.db — ключ! Msgstore.db - розшифрована резервна копія WhatsApp Wa.db - контакти

Там ми йдемо. Тепер у нас є ключ, розшифрований crypt7 резервне копіювання WhatsApp і контактів. Тепер, коли ми маємо три витягнуті дані з

WhatsAppKeyExtractor, там є декілька програм для перегляду повідомлень, наприклад переглядач SQLite, переглядач WhatsApp.

A. Транспортна безпека

Весь зв’язок між клієнтами WhatsApp і сервери WhatsApp розташовані в окремому зашифрованому вигляді канал. Це на Windows Phone, iPhone та Android клієнти з наскрізним шифруванням використовують Noise Pipes з Curve25519, AES-GCM і SHA256 від Noise Protocol Framework для тривалої взаємодії з'єднання.

Це надає клієнтам кілька приємних властивостей:

1. Надзвичайно швидке легке налаштування та відновлення підключення

2. Шифрує метадані, щоб приховати їх від несанкціонованої мережі спостерігачі. Немає інформації про підключеного користувача': особистість розкривається.

3. На сервері не зберігаються секрети автентифікації клієнта. Клієнти автентифікуються за допомогою ключа Curve25519 пара, тому сервер зберігає лише публічні дані клієнта ключ автентифікації. Якщо база даних користувачів сервера коли-небудь скомпрометовано, приватні облікові дані автентифікації не будуть бути розкритим.

B. Можливі способи отримання доступу до закритого ключа

Висновок

1.Viber:

-Криптографічні механізми:

- Використовує протокол шифрування TLS для захисту передачі даних.

- Шифрує повідомлення клієнт-сервер та сервер-сервер за допомогою шифру AES.

- Використовує асиметричний алгоритм RSA для обміну ключами.

-Рівень захищеності:

- Взагалі вважається безпечним, але докладніший аналіз важливий для визначення потенційних уразливостей.

2.WhatsApp:

-Криптографічні механізми:

- Використовує протокол Signal для захищеної передачі повідомлень.

- Шифрує повідомлення, дзвінки та файли за допомогою протоколу шифрування Signal.

- Всі ключі зберігаються локально на пристрої користувача.

-Рівень захищеності:

- Загально визнаний як високий рівень захищеності, оскільки використовує відкриті та перевірені криптографічні протоколи.

3.Skype:

-Криптографічні механізми:

- Використовує шифрування на основі протоколу TLS для захисту передачі даних.

- Використовує алгоритми шифрування для конфіденційності даних.

-Рівень захищеності:

- Рівень захищеності Skype вважається досить високим, але іноді були виявлені уразливості в минулому.

4.Telegram:

-Криптографічні механізми:

- Використовує протокол MTProto для шифрування повідомлень.

- Використовує асиметричний алгоритм RSA для обміну ключами.

-Рівень захищеності:

- Є деякі суперечки серед експертів стосовно рівня захищеності. Багато залежить від того, наскільки добре виконано криптографічні рішення.

Порівняльний аналіз:

- WhatsApp вважається однією з найбільш захищених систем завдяки використанню протоколу Signal.

- Viber та Skype мають прийнятний рівень безпеки, але останнім часом були виявлені деякі уразливості в Skype.

- Telegram використовує власний протокол, але думки щодо його захищеності розділені.

Рекомендації для користувачів:

- Завжди використовуйте найновіші версії додатків.

- Використовуйте сильні паролі та функції аутентифікації.

- Уважно керуйте правами доступу до даних.

- Будьте обережні з відправкою конфіденційної інформації в месенджерах.

Кожна з систем має свої переваги та недоліки в плані криптографічних механізмів. Безпека є постійно змінюваним полем, і користувачам рекомендується слідкувати за новинами та оновленнями, щоб залишатися захищеними.