Дослідження криптографічних протоколів систем WebMoney, PayPal

Недождій Максим, Буржимський Ростислав

ФІ-42мн

Мета

Дослідження особливостей реалізації криптографічних механізмів платіжних систем

Група 1. Дослідити особливості реалізації криптографічних протоколів, а також особливості роботи з електронними гаманцями систем WebMoney та PayPal.

1 SET

Протокол Secure Electronic Transaction (SET) був розроблений у 1996 році за участю Visa, MasterCard, Microsoft та Netscape. Він був спробою створити ідеальне середовище для електронних платежів.

Комерційне впровадження SET не досягло масового успіху, але його архітектурні принципи, зокрема використання подвійного цифрового підпису (Dual Signature), заклали фундамент для розуміння вимог до безпеки фінансових транзакцій.

Специфікація SET базується на суворій ієрархічній моделі довіри. При цьому SET фокусується не на захисті каналу передачі даних, як SSL або TLS, а на захисті транзакції, як об'єкта даних.

Система передбачає обов'язкову автенттифікацію всіх учасників процесу через сертифікати стандарту X.509v3. У ньому зазначено наступні сутності:

- Власник картки (Cardholder) - використовує електронний гаманець, який зберігає цифрові сертифікати та генерує підписи.
- Торговець (Merchant) - має серверне ПЗ, яке здатне обробляти повідомлення SET та взаємодіяти з платіжним шлюзом.
- Платіжний шлюз (Payment Gateway) - виступає з'єднанням між інтернетом і банківською мережею, виконує функції еквайра.

- Центр сертифікації (Certificate Authority) - довірена третя сторона, що видає сертифікати та пов'язує відкриті ключі з користувачами.

Система вимагала розгортання інфраструктури відкритих ключів, де кожен користувач мав отримати персональний сертифікат, що зменшило використання сертифікату через високий поріг входу.

Інновацією у SET став концепт подвійного підпису, який використовувався для вирішення проблеми конфіденційності даних у багатосторонній транзакції. Суть полягала у тому, що торговець має знати деталі замовлення, але не має знати платіжну інформацію покупця, а банк повинен отримати платіжну інформацію, але не обов'язково має знати деталі замовлення. Подвійний підпис дозволяв клієнту зв'язати платіжну інформацію з замовленням таким чином, щоб кожна сторона могла отримати лише свою частину інформації.

Для обчислення подвійного підпису клієнт має:

1. Обчислити геші від платіжної інформації (PI) та деталей замовлення (OI) (за допомогою SHA-1 в оригінальному тексті (хаха)), отримати PIMD та OIMD (payment information message digest та order information message digest)
2. Конкатенувати отримані геші і загешувати їх ще раз, отримати POMD
3. Зашифрувати повідомлення своїм ключем, отримати DS
4. Передати торговцю OI, PIMD, DS, свій відкритий ключ
5. Передати банку PI, OIMD, DS, свій відкритий ключ

Для перевірки торговець виконує наступні дії:

1. Обчислює геш від OI, отримує OIMD
2. Конкатенує його з PIMD і гешує результат
3. Розшифровує DS відкритим ключем
4. Порівнює отримані значення

При цьому, SET виявився комерційно невдалим у порівнянням з SSL та TLS. Основні причини цього:

- Складність використання для користувача, що конкурує з ідеєю максимального спрощення транзакції у сучасній електронній комерції
- Занадто висока вартість інфраструктури відкритих ключів для банків

- У час виходу протоколу, обчислення займали багато часу на тогочасних пристроях
- SSL (а пізніше TLS) забезпечували "достатній рівень безпеки" , який у комбінації з політикою нульової відповідальності клієнтів, впровадженій Visa та MasterCard повністю знищили потребу у більш захищеному, але дорожчому і важчому протоколі.

2 WebMoney

WebMoney використовує декілька варіантів поширення свого сервісу, який включає роботу з файлами, мобільний застосунок, локальну браузерну версію і серверну версію.

Першим був WebMoney Keeper WinPro. Він використовував специфічний формат файлу ключів з розширенням .kwm, при втраті якого кошти також було втрачено. Файл .kvm мав розмів 164 байти, його вміст шифрувався за допомогою потокового шифру RC4, ключ шифрування формувався на основі пароля користувача і його ідентифікатора WebMoneyID. Припускається, що ключ формувався за допомогою застосування MD5 до конкатенованих пароля та WMID користувача.

Після дешифрування файлу .kwm у пам'яті відновлюється приватний ключ RSA. При цьому доступно експортування ключів у різні формати.

Автентифікація відбувається за рахунок Challenge-Response.

1. Сервер надсилає випадковий рядок (nonce/challenge)
2. Клієнт підписує цей ряжок своїм приватним ключем RSA, отриманим з .kwm
3. Підпис надсилається на сервер і порівнюється за допомогою збереженого відкритого ключа користувача.

Другим є WebMoney Keeper WebPro (Light). Ця версія відмовилась від файла з ключем у форматі .kwm і натомість використовує стандартний сертифікат X.509, що у певному сенсі наближує його до SET.

Для використання, користувач генерує пару ключів RSA локально (стандартний розмір 2048 бітів). Для цього можуть використовуватись засоби браузера, такі, як застарілий HTML5 Keygen або Web Crypto API, або зовнішній OpenSSL. Далі сертифікат разом з приватним ключем пакується у контейнер PKCS#12 для імпорту у браузер. Для автентифікації використовується двосторонній TLS, де клієнт автентифікується перед сервером своїм сертифікатом під час рукостискання.

Третій варіант автентифікації передбачає використання їх мобільного застосунку E-num. Приватний ключ зберігається на мобільному пристрої (наприклад смартфон), що зменшує ризик втратити його через шкідливе пз на вашому комп'ютері.

При авторизації користувач вводить число-запитання, яке показано на сайті у мобільний додаток, до якого застосовується секретний ключ з захищеної пам'яті і шифрується, ймовірно, за допомогою HMAC-SHA. Цей процес дозволяє логін з будь-яких платформ, за умови, що ваш мобільний пристрій під рукою.

Останній доступний варіант автентифікації це WebMoney Keeper Standard (Mini), де вхід відбувається на сайт через смс повідомлення, і усі дії також захищені підтвердженням через смс. Обмеженням є неможливість створити кілька гаманців однакового типу при використанні цього варіанту авторизації. Плюсом є те, що в такому варіанті ніщо не зберігається у пам'яті, що користувач міг би зламати.

Плюсом підходу WebMoney є те, що у еру коли все у хмарі, збереження ключа локально дає захист від можливих поламок у мережі.

Недоліком підходу WebMoney є те, що користувачі можуть самостійно скомпроментувати свій ключ, а також їх девайс вразливий до зовнішнього впливу і необережного використання.

3 PayPal

PayPal використовує більш "сучасний" підхід, притаманний фінтех-платформам, відмову від клієнтської криптографії на користь захисту транспортного рівня та токенізації на стороні сервера. При цьому, хоча основний підхід саме через сервер, компанія все одно пропонує криптографічні інструменти для інтеграції під назвою Encrypted Website Payments (EWP).

Згідно до вимог PSI DSS та внутрішніх стандартів безпеки, PayPal примусово вимагає використання TLS 1.2 або вище для усіх з'єднань API та веб-інтерфейсів. PayPal надає перевагу комбінації шифрів, яка забезпечує досконалу пряму секретність (Perfect Forward Secrecy) та автентифіковане шифрування.

Основні підтримувані набори це TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 та TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256. Тут алгоритмом для обміну ключів є ECDHE (Ефемерний Діфі Хелман на еліптичних кривих), автентифікація за допомогою RSA, шифрування через AES-256-GCM або AES-128-GCM відповідно, псевдовипадковою функцією виступає SHA-384 або SHA-256 відповідно.

ECDHE забезпечує досконалу пряму секретність. Навіть, якщо у довгостроковий приватний ключ буде перехоплено у майбутньому, зловмисник не зможе розшифрувати перехоплений раніше трафік, бо сеансові ключі ефемерні.

AES-GCM (AES у режимі роботи лічильник з автентифікацією Галуа) надає автентифіковане шифрування, у якому легко помітити підробку, і усуває вразливість до атак активного MITM, атак на основі оракула та Padding oracle.

SHA-384/256 замінює слабкий SHA-1 у TLS.

Якщо ж використовувати EWP, то він захищає платіжні кнопки HTML від підробок (наприклад зміни ціни зловмисником у коді сторінки). Реалізація базується на стандартах S/MIME та PKCS#7.

Інтеграція EWP вимагає від мерчанта генерації власної пари ключів та обміну сертифікатами з PayPal. Процес включає такі кроки:

- Мерчант створює приватний ключ RSA (рекомендовано 2048 біт) та сертифікат X.509.
- Публічний сертифікат завантажується в акаунт PayPal, а сертифікат PayPal завантажується на сервер мерчанта.
- Дані форми (назва товару, ціна) підписуються приватним ключем мерчанта (для забезпечення цілісності та автентичності) та шифруються публічним сертифікатом PayPal (для забезпечення конфіденційності).

Для шифрування даних використовується симетричний алгоритм, ключ якого зашифровано асиметрично. Історично використовувалися слабкі алгоритми (DES, RC2), але сучасні вимоги диктують перехід на AES-256.

Для EWP критичним є саме формат S/MIME (PKCS#7), де дані підписуються та шифруються в конверт, який може розпакувати тільки сервер PayPal.

Для програмного доступу PayPal використовує протокол OAuth 2.0. Клієнт (додаток) обмінює Client ID та Client Secret на тимчасовий токен доступу (Access Token) типу Bearer. Для зменшення області дії PCI DSS, PayPal використовує мережеву токенізацію. Реальні номери карток (PAN) замінюються на унікальні цифрові токени, які генеруються платіжними мережами (Visa/Mastercard). Це дозволяє мерчанту зберігати токен замість чутливих даних, знижуючи ризики витоку інформації.

4 Payword

Payword та Micromint це системи мікроплатежів. Вони використовуються коли вартість транзакції більша за вартість суми, яка переводиться. Тому дороге обчислення асиметричного підпису замінюється на ефективну геш-функцію.

PayWord базується на концепції ланцюжків гешів для забезпечення серії платежів одному продавцю.

Користувач обирає випадкове число w_n і обчислює ланцюжок у зворотньому порядку, використовуючи геш-функцію h (наприклад SHA-256). w_0 буде коренем ланцюга.

Користувач створює сертифікат, що містить w_0, інформацію про продавця та термін дії. Цей документ один раз підписується RSA/ECC за допомогою ключа користувача.

Кожна i-та оплата є розкриттям значення w_i. Продавець перевіряє $h(w_i) = w_{i+1}$. Якщо значення співпадає, то транзакція успішна. Оскільки геш-функція односторонння, то підтвердження працює.

Таким чином один підпис застосовується на n транзакцій одночасно. Хеш функції не тільки працюють швидше за підпис, а й використовують простіші операції, що зменшує вартість транзакції.

5 Micromint

Micromint не використовує підпис взагалі, натомість використовують колізії геш-функцій.

Монета в системі Micromint є k-сторонньою колізією геш-функції, тобто набір значень $(x_1, x_2, \ldots, x_k)$, які мають однаковий геш:

$$h(x_1) = h(x_2) = \cdots = h(x_k) = y$$

При цьому y має певну структуру (наприклад перші t бітів дорівнюють нулю)

Безпека Micromint базується на різниці складності обчислення для емітента та зловмисника.

Для знаходження k-сторонньої колізії потрібно $2^{\frac{n(k-1)}{k}}$ спроб на одну конкретну колізію.

При цьому емітент інвестує у обладнання, щоб генерувати будь-які колізії масово, а далі використовує лише ті значення, які задовольняють заданому критерію (наприклад старші біти рівні заданому числу q).

Виникає ситуація, коли для емітента генерація монет є дешевою за рахунок масштаюу, але обчислювально невигідним для зловмисника, який намагається підробити монету.

6 Електронні гроші

Для систем, де пріоритетом є анонімність (цифрова готівка), використовуються протоколи сліпого підпису, запропоновані Девідом Чаумом. Вони дозволяють банку завірити цифрову монету, не бачачи її серійного номера.

На прикладі сліпих підписів через RSA. Протокол забезпечує властивість незв'язності (unlinkability): банк не може пов'язати монету, яку він підписав, з монетою, яку користувач витратив.

Алгортим має 4 етапи: підготовка, підпис, зняття засліплення, доказ коректності

1. • Клієнт обирає серійний номер банкноти m

- Генерує випадковий множник засліплення r такий, що $gcd(r, N) = 1$
- Обчислює засліплене значення m' за допомогою відкритого ключа банку

$$m' \equiv m * r^e \pmod{N}$$

2. - Банк підписує повідомлення закритим ключем

$$s' \equiv (m')^d \pmod{N}$$

- Повертає s' клієнту. При цьому банк не бачив m.

3. Клієнт отримує справжній підпис s для монети m, помноживши s' на обернене до r значення.

$$s \equiv s' * r^{-1} \pmod{N}$$

4. Доказом коректності є $s \equiv (m * r^e)^d * r^{-1} \equiv m^d * r^{ed} * r^{-1} \equiv m^d * r * r^{-1} \equiv m^d \pmod{N}$

Аналіз показує, що розвиток платіжних систем рухається від складних клієнтських рішень до серверних.

WebMoney демонструє приклад консервативної архітектури, де безпека базується на володінні файлом ключів. Використання застарілих алгоритмів (RC4/MD5) у форматі .kwm є потенційним ризиком, який компенсується малими розмірами файлу та впровадженням додаткових факторів автентифікації (E-NUM).

PayPal еволюціонував у бік максимальної сумісності та делегування безпеки стандартним протоколам Інтернету (TLS). Впровадження EWP з використанням AES-256 та S/MIME демонструє, як можна забезпечити цілісність даних у відкритому веб-середовищі без примусу користувача до встановлення складного ПЗ.

Теоретичні протоколи (SET, PayWord, MicroMint), хоч і не стали домінуючими у чистому вигляді, визначили вектори розвитку: SET показав важливість PKI (яка зараз реалізована в SSL/TLS), а ідеї хеш-ланцюжків та сліпих підписів знаходять нове життя у сучасних криптовалютах та технологіях блокчейн, де ефективність верифікації та анонімність знову стають критичними вимогами.

Забезпечення захищеності сучасних систем базується не на виборі одного "ідеального" алгоритму, а на ешелонованому захисті: від використання стійких шифрів (AES-GCM, ECC) на транспортному рівні до токенізації даних та використання апаратних ключів на рівні користувача.