Наскрізне шифрування, пояснене по-справжньому

Що кажуть провайдери, коли кажуть про E2EE, і про що вони замовчують. Дидактичне пояснення механізму та його меж без рекламної упаковки.

Давайте прояснимо: WhatsApp каже, що ваші повідомлення захищені наскрізним шифруванням. Це правда — і цього недостатньо. Якщо резервна копія відправляється в iCloud або Google Drive без додаткового шифрування, шифрування ламається на вашому власному телефоні. Оперативне питання полягає не в тому, чи зашифровано повідомлення, а в тому, де знаходяться ключі.

Що шифрування означає насправді

Зашифрувати повідомлення — значить перетворити його на щось, що виглядає як шум для будь-кого, хто не володіє певною інформацією, яка називається ключем. Операція виконується на пристрої відправника і за допомогою правильного ключа скасовується на пристрої одержувача. У проміжку повідомлення передається як послідовність байтів без видимого сенсу. Це проста ідея. Решта статті присвячена нюансам, які перетворюють її, залежно від випадку, на реальну гарантію або лише на маркетинговий ярлик.

Прикметник *наскрізне* — англійською *end-to-end*, скорочено E2EE — додає точності. Шифрування виконується не для того, щоб проміжний сервер міг його прочитати та доставити. Воно виконується для того, щоб тільки обидві сторони — пристрій відправника та пристрій одержувача — володіли ключем. Будь-який сервер, через який проходить повідомлення, бачить шум, а не повідомлення. У цьому полягає технічна відмінність від шифрування *при транзиті*, коли контент передається у зашифрованому вигляді від одного сервера до іншого, але кожен сервер, через який він проходить, розшифровує його для пересилання, тимчасово відновлюючи відкритий текст.

Парадокс спільного секрету

Існує очевидна проблема. Щоб дві людини могли шифрувати та розшифровувати повідомлення один одного, обом потрібен один і той самий ключ. Але як домовитися про цей ключ, якщо все, що вони надсилають один одному, за визначенням проходить через канал, де хтось може підслуховувати? Домовитися про ключ у тому самому каналі, де вони пізніше будуть його використовувати, здається неможливим: якщо зловмисник почує його при узгодженні, він зможе розшифрувати все подальше. Протягом десятиліть класична криптографія вирішувала це жорстким способом: ключі передавалися особисто, до початку використання, при фізичних зустрічах. Посли носили портфелі з ключами, вшитими в підкладку пальто.

У сучасній електронній пошті таке рішення не масштабується. Якби нам довелося фізично йти додому до кожної людини, з якою ми маємо намір спілкуватися у зашифрованому вигляді, ми б ні з ким не встигли поговорити. Питання, поставлене п'ятдесят років тому криптографічною спільнотою, звучало так: чи можливо, щоб дві людини, які не знають одна одну і які ділять тільки відкритий канал, домовилися у цьому самому відкритому каналі про секрет, який ніхто, хто слухає канал, не зможе дізнатися?

Елегантність Diffie-Hellman

У 1976 році два математики, Уітфілд Діффі та Мартін Хеллман, довели щось на перший погляд неможливе: що дві людини, розмовляючи тільки через відкритий канал — канал, де будь-хто може чути все, що вони говорять, — можуть домовитися про секретний пароль так, щоб жоден слухач не міг його розкрити. Це звучить як магія. Але це не так: це математика. Обмін ключами Діффі-Хеллмана, як він відомий відтоді, є основою практично всього зашифрованого зв'язку в інтернеті, і півстоліття інтенсивного використання та всесвітнього академічного аналізу підтверджують його надійність. Той, хто хоче побачити візуальну інтуїцію або математику, може читати далі. Той, хто воліє вірити, що це працює, також може продовжити, не втрачаючи нитки статті.

Для тих, хто хоче уявити це наочно, існує відома аналогія з кольорами. Уявіть, що Аліса та Бруно відкрито домовляються про базовий колір — скажімо, жовтий — на очах у Єви, яка їх підслуховує. Кожен з них таємно вибирає другий секретний колір і змішує свій секрет із жовтим. Аліса отримує особливий помаранчевий; Бруно отримує особливий зелений. Вони обмінюються результатами на очах у Єви. Тепер кожен змішує отриманий колір зі своїм секретом, і обидва приходять до одного й того самого кінцевого кольору, оскільки порядок змішування не має значення. Єва бачила жовтий і дві проміжні суміші, але не секрети; без одного із секретів вона не зможе отримати кінцевий колір. Реальна математика замінює кольори піднесенням до степеня у модулярних групах або на еліптичних кривих, але ідея та сама: спільний секрет створюється публічно без можливості його відновлення будь-ким у каналі.

В арифметиці, для тих, хто воліє бачити механізм: Аліса вибирає секретне число a, Бруно вибирає b. Вони відкрито обмінюються g^a та g^b по каналу. Аліса обчислює $(g^b)^a$, а Бруно обчислює $(g^a)^b$; обидва приходять до одного й того самого g^{ab}. Єва бачить g, g^a та g^b, що проходять по каналу, але відновлення a з g^a — так звана проблема дискретного логарифма — вимагає астрономічного часу обчислень, що перевищує вік Всесвіту, коли g вибирається у відповідній математичній групі.

Для тих, хто хоче перевірити це на малих числах. Обмін Діффі-Хеллмана можна пройти від початку до кінця з цифрами, достатньо малими, щоб робити обчислення вручну. Той, хто воліє не заглиблюватися в арифметику, може пропустити цей блок, не втрачаючи нитки статті; той, хто хоче побачити, як механізм працює крок за кроком, знайде це тут. **Публічні правила**, які може прочитати будь-хто: просте число $p = 11$ (у справжньому Діффі-Хеллмані воно складається приблизно з трьохсот цифр; ми використовуємо одинадцять, щоб розрахунки вмістилися на одній сторінці), основа $g = 2$, і правило, що вся арифметика виконується за *модулем p* — ви обчислюєте, ділите на p і залишаєте остачу, як годинник з одинадцятьма позиціями, який повертається до нуля після проходження десятки. **Приватні вибори**, по одному в кожного, які ніколи не розголошуються: Аліса вибирає $a = 4$. Бруно вибирає $b = 7$.

Крок 1. Аліса обчислює $2^4 = 16$, потім 16 mod 11 = **5**. Вона відправляє п'ятірку. Єва записує її.

Крок 2. Бруно обчислює $2^7 = 128$, потім 128 mod 11 = **7**. Він відправляє сімку. Єва теж її записує. Після двох відправок блокнот Єви містить чотири значення: $p = 11$, $g = 2$, A = 5, B = 7. Їй не вистачає спільного числа, яке Аліса та Бруно збираються вивести — і яке Єва не зможе відновити.

Крок 3. Аліса бере сімку, яку їй надіслав Бруно, і підносить її до свого приватного степеня $a = 4$. Щоб уникнути роботи з $7^4 = 2401$, розрахунки проводяться частинами із застосуванням модуля на кожному кроці:

$7^2 = 49$

49 mod 11 = 5

$7^4 = (7^2)^2 = 5^2 = 25$

25 mod 11 = 3

Аліса отримує число **3**.

Крок 4. Бруно бере п'ятірку, яку йому надіслала Аліса, і підносить її до свого приватного степеня $b = 7$. Знову по частинах:

$5^2 = 25$ mod 11 = 3

$5^4 = (5^2)^2 = 3^2 = 9$ mod 11 = 9

$5^6 = 5^4 \times 5^2 = 9 \times 3 = 27$ mod 11 = 5

Нарешті $5^7 = 5^6 \times 5 = 5 \times 5 = 25$ mod 11 = **3**.

Бруно також отримує **3**.

Обидва прийшли до одного й того самого числа, 3, працюючи паралельно. Жоден із них ні в який момент не відправляв свій приватний степінь. Аліса не знає, що $b = 7$; Бруно не знає, що $a = 4$. Кожен використав публічне значення, надіслане іншим, у поєднанні зі своїм власним приватним степенем, і вони зустрілися в одному місці призначення. **Чому вони приходять до одного й того самого числа?** Ось що кожен з них обчислив: Аліса, $(g^b)^a = 2^{7 \times 4} = 2^{28}$ mod 11. Бруно, $(g^a)^b = 2^{4 \times 7} = 2^{28}$ mod 11. Це одна і та ж величина, тому що порядок множення степенів не має значення ($7 \times 4 = 4 \times 7$). Кожен дійшов до одного пункту призначення різним шляхом.

А як щодо Єви? У неї в блокноті є $p = 11$, $g = 2$, A = 5, B = 7, і вона хотіла б отримати 3. Щоб обчислити це, їй потрібно було б знати a або b — але жоден з них не передавався по каналу. Її єдиний вихід — запитати себе: «для якого степеня a виконується умова 2^a mod 11 = 5?». З таким малим p вона може спробувати 0, 1, 2, 3, 4... і знайти відповідь менш ніж за хвилину. Але якщо замінити 11 на просте число з трьохсот цифр, простір можливих степенів міститиме більше елементів, ніж атомів у видимому всесвіті. **На сьогоднішній день людству не відомий жоден алгоритм, здатний пройти цей простір швидше, ніж за мільярди років.** Це так звана *проблема дискретного логарифма*: легко обчислити в прямому напрямку, обчислювально неможливо в зворотному. І саме тому шифрування витримує, навіть якщо Єва стежила за всією розмовою літера за літерою.

Три прості інгредієнти — арифметика циферблата, піднесення до степеня і комутативність множення ($a \cdot b = b \cdot a$) — у поєднанні створюють протокол, від якого половина людства щодня залежить у своїх приватних комунікаціях. Жодна з цих трьох частин окремо не здається особливою. Вирішальним є збирання.

Від Diffie-Hellman до протоколу Signal

Наскрізне шифрування, яке використовують сьогодні професійні додатки для обміну повідомленнями, майже без винятку спирається на елегантну та посилену версію обміну Діффі-Хеллмана. Протокол Signal, розроблений Тревором Перріном та Моксі Марлінспайком у період з 2013 по 2016 рік, є еталоном. Він поєднує в собі дві ключові ідеї. Перша — обмін ключами на еліптичних кривих (X25519), який створює початковий спільний секрет між двома пристроями. Друга — так званий Double Ratchet — подвійний храповик, — який автоматично оновлює ключі з кожним повідомленням, так що компрометація пристрою сьогодні не дозволяє розшифрувати повідомлення з минулого, так само як і повідомлення з майбутнього після повороту храповика.

У Zig обмін X25519, що створює спільний секрет між двома пристроями, вміщується у шість рядків з використанням стандартної бібліотеки:

```
const std = @import("std");
const X25519 = std.crypto.dh.X25519;
```

```
// Alicia y Bruno generan cada uno un par (privada, pública).
const par_alicia = X25519.KeyPair.generate(io);
const par_bruno  = X25519.KeyPair.generate(io);

// Cada parte recibe la clave pública de la otra y deriva el mismo secreto.
const secreto_alicia = X25519.scalarmult(par_alicia.secret_key, par_bruno.public_key) catch unreachable;
const secreto_bruno  = X25519.scalarmult(par_bruno.secret_key,  par_alicia.public_key) catch unreachable;
// secreto_alicia == secreto_bruno  (32 bytes)
```

Що відбувається у цих шести рядках: Публічні ключі передаються відкрито. Приватні ключі ніколи не покидають відповідний пристрій. Кожна сторона виводить зі свого приватного та публічного ключа іншої сторони один і той самий секрет із тридцяти двох байтів, який ніхто в каналі не може відновити. Цей секрет пізніше служить основою (seed) для шифрування повідомлень, якими обмінюються. Double Ratchet протоколу Signal додає постійну ротацію цього матеріалу, щоб компрометація одного моменту не ставила під загрозу решту розмови.

І що саме знаходиться всередині `std.crypto.dh.X25519`? Жодної прихованої магії. Це дві короткі функції, які можна прочитати повністю в самій стандартній бібліотеці Zig. Перша виводить публічний ключ із приватного — «g^a» обміну:

```
pub fn recoverPublicKey(secret_key: [secret_length]u8) IdentityElementError![public_length]u8 {
    const q = try Curve.basePoint.clampedMul(secret_key);
    return q.toBytes();
}
```

Мовою статті: приватний ключ «множиться» — в еліптичному сенсі, а не в елементарному арифметичному — на базову точку кривої Curve25519, і результат серіалізується в тридцять два байти. Операція `clampedMul` — це посилена версія цього скалярного множення: вона включає в себе заходи захисту, які криптографічна спільнота додавала протягом багатьох років, щоб протистояти відомим родинам атак. Два рядки тіла функції.

Друга функція поєднує ваш приватний ключ із публічним ключем, який надсилає вам інша сторона. Це «$(g^b)^a$» обміну, те, що створює спільний секрет у тридцять два байти, який ніхто з вас ніколи не передавав:

```
pub fn scalarmult(secret_key: [secret_length]u8, public_key: [public_length]u8) IdentityElementError![shared_length]u8 {
    const q = try Curve.fromBytes(public_key).clampedMul(secret_key);
    return q.toBytes();
}
```

Ще два рядки. Отриманий публічний ключ інтерпретується як точка на кривій і «множиться» на власний приватний ключ. Завдяки комутативності операції над кривою — аналогічній до комутативності множення степенів, яку ми бачили у числовому прикладі, — обидві сторони в кінцевому підсумку отримують одну й ту саму серіалізовану точку: саме той спільний секрет, про який йдеться у статті.

Ось і все. Те, що в додатку здається магією, насправді є двома функціями по три рядки кожна. Технічна складність зосереджена в одній операції `clampedMul`, яка написана нижче в тій самій стандартній бібліотеці, десятиліттями перевірялася міжнародною криптографічною спільнотою і доступна будь-кому, хто хоче прочитати її буква за буквою. Чорного ящика немає ані в нашому додатку, ані в стандартній бібліотеці Zig. Є відкритий вихідний код, який людина може зрозуміти, вибравши темп, в якому вона хоче в нього заглибитися.

Що захищає наскрізне шифрування

Що E2EE захищає добре, за умови правильної реалізації, — це вміст повідомлення при транзиті. Проміжний сервер, що приймає та пересилає зашифровані дані, побачить послідовність незрозумілих байтів. Зловмисник із доступом до кабелю, роутера, точки доступу Wi-Fi побачить те саме. Провайдер послуг, що зберігає копії трафіку, не зможе прочитати його згодом. Уряд, що наказав оператору зв'язку видати вміст, отримає ті самі незрозумілі байти, які сервер мав спочатку.

Це, у практичному плані, дуже багато. Це різниця між написанням листа всередині непрозорого конверта та написанням його на листівці. І те, і інше доходить до адресата. Але тільки одне зберігає вміст у таємниці від листоноші.

Що не захищає наскрізне шифрування

Це варто знати так само добре. E2EE не захищає метадані: сервер все одно знає, що користувач А надсилає дані користувачу Б, о котрій годині, з якою частотою і звідки, хоча й не знає, що саме він каже. Ці метадані, як ми вже аргументували у статті *Шифрувати — не означає бути приватним*, часто є більш показовими, ніж сам контент. Знання того, що хтось телефонував до адвокатської контори, яка спеціалізується на розлученнях, у п'ятницю о 22:00 протягом тридцяти хвилин, розповідає історію, яку зміст дзвінка ніколи б не розповів. Це та сама ситуація, що й спостереження за людиною, яка кілька разів входить і виходить з онкологічної клініки: не потрібно чути нічого з того, про що говорять всередині, щоб уявити, що відбувається. Один ізольований метаданий може нічого не значити; кілька перехресних даних малюють щось занадто схоже на правду. E2EE не захищає кінцеві пристрої: якщо пристрій одержувача скомпрометований шкідливою програмою, повідомлення розшифровується для цього одержувача у звичайному режимі, і шкідлива програма його читає. E2EE не захищає від підміни особи співрозмовника як такої: якщо Аліса вірить, що розмовляє з Бруно, але зловмисник вклинився на самому початку (атака *man in the middle*), а протокол не передбачає незалежної перевірки, обидві сторони в результаті розмовляють із непроханим гостем, думаючи, що говорять одна з одною.

Є четверта річ, яку варто сформулювати недвозначно. E2EE не заважає провайдеру, який стверджує, що він його пропонує, додатково зберігати копію незашифрованого повідомлення у своїх власних системах. Твердження «мої повідомлення зашифровані наскрізним шифруванням» і твердження «провайдер не зберігає мій контент» — не одне й те саме. Додаток може виконувати перше, порушуючи друге; ми неодноразово бачили це у заголовках газет з 2018 року. Користувач, якщо тільки код клієнта не є таким, що можна перевірити, не має технічного способу відрізнити один випадок від іншого без експертного розслідування. Найвідоміший випадок серед широкої публіки: WhatsApp шифрує повідомлення наскрізним шифруванням при транзиті, але якщо користувач активує резервне копіювання в iCloud або Google Drive без додаткового шифрування, ця копія зберігається у придатному для читання вигляді в інфраструктурі третьої сторони, і шифрування порушується на стороні самого користувача.

Питання, яке оператор не хоче чути

Додаток, що стверджує, що він шифрує наскрізним шифруванням, технічно може робити одну з трьох реєй щодо ключів:

1. **Ключі знаходяться тільки на пристроях.** Вони генеруються і знаходяться виключно на пристроях користувачів; оператор їх не знає і не зберігає. Це оптимальний варіант.
2. **Оператор може отримати доступ, якщо захоче.** Оператор володіє ключами користувачів (або може генерувати їх на свій розсуд) і зберігає їх у своїх базах даних. Якщо він захоче або буде змушений, він зможе прочитати вміст. Це характерно для більшості «хмарних» сервісів.
3. **Оператор не може отримати доступ за дизайном, але він контролює доступ.** У оператора немає ключів, але він контролює додаток, який їх генерує. Якщо його змусять, він може надіслати шкідливе оновлення, яке перехопить ключі або контент до шифрування. Це характерно для багатьох комерційних сервісів E2EE.

Отже, оперативне питання полягає не в тому, чи зашифровано щось, а в тому, хто контролює пристрій і програмне забезпечення, що керує ключами. У Solo2 ключі зберігаються виключно у вашому «Сейфі» (IndexedDB, зашифрована вашим паролем), а програмне забезпечення є відкритим вихідним кодом, що підлягає перевірці.

Для професійного читача

Наскрізне шифрування — це інструмент цифрового суверенітету. Але, як і будь-який інstrument, його ефективність залежить від руки, яка його тримає, і ґрунту, на якому він стоїть.

1. Де генеруються криптографічні ключі і де вони фізично знаходяться? Якщо оператор може отримати до них доступ (навіть тимчасово, навіть під виглядом відновлення), то E2EE є лише номінальним.
2. Чи існує незалежна перевірка співрозмовника (коди безпеки, QR-коди, порівняння поза каналом), яка запобігає атаці man-in-the-middle під час встановлення розмови?
3. Чи підлягає код клієнта аудиту — чи є він відкритим, опублікованим, відтворюваним — або ж він вимагає довіри слову провайдера про те, що клієнт робить насправді?
4. Які метадані генерує та зберігає сервіс, і як довго? Навіть якщо контент є непрозорим, метадані можуть відновити значну частину конфіденційної інформації.

Ці чотири питання не вимагають складної технічної інформації; вони запитують інформацію, на яку будь-який чесний оператор може відповісти у своїй публічній документації. Якість і точність відповіді говорять про продукт не менше, ніж сама відповідь.

Наскрізне шифрування, якщо воно виконане правильно, є однією з найвитонченіших конструкцій, які сучасна криптографія принесла в повсякденну практику. Оригінальна ідея — двоє людей можуть домовитися про секрет через відкритий канал — належить Whitfield Diffie та Martin Hellman, 1976 рік; через півстоліття ми продовжуємо жити в її наслідках. Але, як і з будь-якою технічною обіцянкою, її цінність залежить від реального виконання, а не від ярлика. Питання чесного професіонала не в тому, «чи це зашифровано?», а в тому, «у кого ключі?». Відповіді мають різні наслідки. Варто їх знати.

Джерела та додаткова література

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, листопад 1976 р. Основоположна стаття з криптографії з відкритим ключем.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, публічна специфікація від Open Whisper Systems, редакція 2016 р. Основа протоколу Signal та його промислових похідних.
- RFC 7748 — Elliptic Curves for Security (IETF, січень 2016 р.). Нормативна специфікація кривих X25519 та X448, що використовуються у сучасних обмінах ключами.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Розділи про обмін ключами та протоколи автентифікованого шифрування.
- Регламент (ЄС) 2024/1183 про європейську систему цифрової ідентифікації (eIDAS 2) — встановлює рамки, в яких незалежна перевірка співрозмовника набуває інституційної підтримки, і де відмінність між номінальним та реальним шифруванням має різні юридичні наслідки.

← ПопереднійKill switch та інституційне захопленняНаступний →Бізнес-модель як сигнал довіри

Останні матеріали

- Аналіз · 18 травня 2026 р. Реальна vs уявна конфіденційність: питання, які варто собі поставити
- Аналіз · 18 травня 2026 р. Self-hosting як професійна практика
- Концепція · 18 травня 2026 р. 24 слова: що таке криптографічна ідентичність

Візьміть цю статтю з собою куди завгодно.

↓ Markdown ↓ Звичайний текст ↓ PDF

Файл буде завантажено на ваш пристрій. Звідти ви можете зберегти його, імпортувати в Solo2 або поділитися ним де завгодно. Cuadernos не вирішує місце призначення за вас.

Сургучна печатка · SHA-256 0b66e092e1e8c1e1379e7a9dfbde3d9384979590156063636 57b99a0c016e3f5

Cuadernos Lacre · Видання Menzuri Gestión S.L. ·
автор R.Eugenio · під редакцією команди Solo2.