Сквозное шифрование, объясненное по-настоящему

Что говорят провайдеры, когда говорят об E2EE, и о чем они умалчивают. Дидактическое объяснение механизма и его пределов без рекламной упаковки.

Давайте проясним: WhatsApp говорит, что ваши сообщения защищены сквозным шифрованием. Это правда — и этого недостаточно. Если резервная копия отправляется в iCloud или Google Drive без дополнительного шифрования, шифрование ломается на вашем собственном телефоне. Оперативный вопрос заключается не в том, зашифровано ли сообщение, а в том, где находятся ключи.

Что шифрование означает на самом деле

Зашифровать сообщение — значит превратить его в нечто, напоминающее шум для любого, кто не обладает определенной информацией, называемой ключом. Операция выполняется на устройстве отправителя и с помощью правильного ключа отменяется на устройстве получателя. В промежутке сообщение передается как последовательность байтов без видимого смысла. В этом и заключается простая идея. Остальная часть статьи посвящена нюансам, которые превращают ее, в зависимости от случая, в реальную гарантию или в маркетинговый ярлык.

Прилагательное *сквозное* — по-английски *end-to-end*, сокращенно E2EE — добавляет точности. Шифрование выполняется не для того, чтобы промежуточный сервер мог его прочитать и доставить. Оно выполняется для того, чтобы только две стороны — устройство отправителя и устройство получателя — владели ключом. Любой сервер, через который проходит сообщение, видит шум, а не сообщение. В этом заключается техническое отличие от шифрования *при транзите*, когда контент передается в зашифрованном виде от одного сервера к другому, но каждый сервер, через который он проходит, расшифровывает его для пересылки, временно восстанавливая открытый текст.

Парадокс общего секрета

Существует очевидная проблема. Чтобы два человека могли шифровать и расшифровывать сообщения друг друга, обоим нужен один и тот же ключ. Но как договориться об этом ключе, если все, что они посылают друг другу, по определению проходит через канал, где кто-то может подслушивать? Договориться о ключе в том же канале, где они позже будут его использовать, кажется невозможным: если злоумышленник услышит его при согласовании, он сможет расшифровать все последующее. На протяжении десятилетий классическая криптография решала эту проблему жестким способом: ключи передавались лично, до начала использования, при физических встречах. Послы носили портфели с ключами, вшитыми в подкладку пальто.

В современной электронной почте такое решение не масштабируется. Если бы нам пришлось физически идти домой к каждому человеку, с которым мы намерены общаться в зашифрованном виде, мы бы ни с кем не успели поговорить. Вопрос, поставленный пятьдесят лет назад криптографическим сообществом, звучал так: возможно ли, чтобы два человека, которые не знают друг друга и которые делят только открытый канал, договорились в этом самом открытом канале о секрете, который никто из слушающих канал не сможет узнать?

Элегантность Diffie-Hellman

В 1976 году два математика, Уитфилд Диффи и Мартин Хеллман, доказали нечто, казавшееся невозможным: два человека, разговаривая только через открытый канал — канал, где любой может слышать все, что они говорят, — могут договориться о секретном пароле так, чтобы ни один слушатель не смог его раскрыть. Это звучит как магия. Но это не так: это математика. Обмен ключами Диффи-Хеллмана, как он известен с тех пор, является основой практически всей зашифрованной связи в интернете, и полвека интенсивного использования и всемирного академического анализа подтверждают его надежность. Тот, кто хочет увидеть визуальную интуицию или математику, может читать дальше. Тот, кто предпочитает верить, что это работает, также может продолжить, не теряя нити статьи.

Для тех, кто хочет представить это наглядно, существует известная аналогия с цветами. Представьте, что Алиса и Бруно открыто договариваются о базовом цвете — скажем, желтом — на глазах у Евы, которая их подслушивает. Каждый из них тайно выбирает второй секретный цвет и смешивает свой секрет с желтым. Алиса получает особый оранжевый; Бруно получает особый зеленый. Они обмениваются результатами на глазах у Евы. Теперь каждый смешивает полученный цвет со своим секретом, и оба приходят к одному и тому же конечному цвету, так как порядок смешивания не имеет значения. Ева видела желтый и две промежуточные смеси, но не секреты; без одного из секретов она не сможет получить конечный цвет. Реальная математика заменяет цвета возведением в степень в модулярных группах или на эллиптических кривых, но идея та же: общий секрет создается публично без возможности его восстановления кем-либо в канале.

В арифметике, для тех, кто предпочитает видеть механизм: Алиса выбирает секретное число a, Бруно выбирает b. Они открыто обмениваются g^a и g^b по каналу. Алиса вычисляет $(g^b)^a$, а Бруно вычисляет $(g^a)^b$; оба приходят к одному и тому же g^{ab}. Ева видит g, g^a и g^b, проходящие по каналу, но восстановление a из g^a — так называемая проблема дискретного логарифма — требует астрономического времени вычислений, превышающего возраст Вселенной, когда g выбирается в подходящей математической группе.

Для тех, кто хочет проверить это на маленьких числах. Обмен Диффи-Хеллмана можно пройти от начала до конца с числами, достаточно маленькими, чтобы производить вычисления вручную. Тот, кто предпочитает не вникать в арифметику, может пропустить этот блок, не теряя нити статьи; тот, кто хочет увидеть, как механизм работает шаг за шагом, найдет это здесь. **Публичные правила**, которые может прочитать любой: простое число $p = 11$ (в реальном алгоритме Диффи-Хеллмана оно состоит примерно из трехсот цифр; мы используем одиннадцать, чтобы вычисления уместились на одной странице), основание $g = 2$, и соглашение о том, что вся арифметика выполняется по *модулю p* — вычисляется, делится на p, и сохраняется остаток, как часы с одиннадцатью позициями, которые возвращаются к нулю после прохождения десятки. **Приватные выборы**, по одному у каждого, которые никогда не разглашаются: Алиса выбирает $a = 4$. Бруно выбирает $b = 7$.

Шаг 1. Алиса вычисляет $2^4 = 16$, затем 16 mod 11 = **5**. Она отправляет пятерку. Ева записывает ее.

Шаг 2. Бруно вычисляет $2^7 = 128$, затем 128 mod 11 = **7**. Он отправляет семерку. Ева тоже ее записывает. После двух отправок блокнот Евы содержит четыре значения: $p = 11$, $g = 2$, A = 5, B = 7. Ей не хватает общего числа, которое Алиса и Бруно собираются вычислить — и которое Ева не сможет восстановить.

Шаг 3. Алиса берет семерку, которую ей прислал Бруно, и возводит ее в свою приватную степень $a = 4$. Чтобы избежать работы с $7^4 = 2401$, вычисления производятся по частям с применением модуля на каждом шаге:

$7^2 = 49$

49 mod 11 = 5

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

25 mod 11 = 3

Алиса получает число **3**.

Шаг 4. Бруно берет пятерку, которую ему прислала Алиса, и возводит ее в свою приватную степень $b = 7$. Снова по частям:

$5^2 = 25$ 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

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

Бруно также получает **3**.

Оба пришли к одному и тому же числу, 3, работая параллельно. Ни один из них ни в какой момент не отправлял свою приватную степень. Алиса не знает, что $b = 7$; Бруно не знает, что $a = 4$. Каждый использовал публичное значение, отправленное другим, в комбинации со своей собственной приватной степенью, и они встретились в одном и том же пункте назначения. **Почему они приходят к одному и тому же числу?** Вот что каждый вычислил: Алиса, $(g^b)^a = 2^{7\times4} = 2^{28}$ mod 11. Бруно, $(g^a)^b = 2^{4\times7} = 2^{28}$ mod 11. Это одна и та же величина, потому что порядок умножения степеней не имеет значения ($7 \times 4 = 4 \times 7$). Каждый пришел к одному и тому же месту назначения разным путем.

А что же Ева? У нее в блокноте есть $p = 11$, $g = 2$, A = 5, B = 7, и она хотела бы получить 3. Чтобы вычислить это, ей нужно было бы знать a или b — но ни то, ни другое не передавалось по каналу. Ее единственный путь — спросить себя: «для какой степени a выполняется условие 2^a mod 11 = 5?». При таком маленьком p она может попробовать 0, 1, 2, 3, 4... и найти ответ меньше чем за минуту. Но если заменить 11 простым числом из трехсот цифр, пространство возможных степеней будет содержать больше элементов, чем атомов в наблюдаемой вселенной. **На сегодняшний день человечеству не известен ни один алгоритм, который мог бы пройти это пространство быстрее, чем за миллиарды лет.** Это так называемая *проблема дискретного логарифма*: легко вычислить в прямом направлении, вычислительно невозможно в обратном. И именно поэтому шифрование выдерживает, даже если Ева следила за всем разговором буква за буквой.

Три простых ингредиента — арифметика циферблата, возведение в степень и коммутативность умножения ($a \cdot b = b \cdot a$) — в комбинации создают протокол, от которого половина человечества каждый день зависит в своих частных коммуникациях. Ни один из этих трех элементов по отдельности не кажется особенным. Решающим является их объединение.

От Diffie-Hellman до протокола Signal

Сквозное шифрование, которое используют сегодня профессиональные приложения для обмена сообщениями, почти без исключения опирается на элегантную и усиленную версию обмена Диффи-Хеллмана. Протокол Signal, разработанный Тревором Перрином и Мокси Марлинспайком в период с 2013 по 2016 год, является эталоном. Он сочетает в себе две ключевые идеи. Первая — обмен ключами на

эллиптических кривых (X25519), который создает первоначальный общий секрет между двумя устройствами. Вторая — так называемый Double Ratchet — двойной храповик, — который автоматически обновляет ключи с каждым сообщением, так что компрометация устройства сегодня не позволяет расшифровать сообщения из прошлого, равно как и сообщения из будущего после поворота храповика.

В Zig обмен X25519, создающий общий секрет между двумя устройствами, умещается в шесть строк с использованием стандартной библиотеки:

```
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)
```

Что происходит в этих шести строках: Публичные ключи передаются открыто. Приватные ключи никогда не покидают соответствующее устройство. Каждая сторона выводит из своего приватного и публичного ключа другой стороны один и тот же секрет из тридцати двух байтов, который никто в канале не может восстановить. Этот секрет позже служит основой (seed) для шифрования обмениваемых сообщений. Double Ratchet протокола Signal добавляет постоянную ротацию этого материала, чтобы компрометация одного момента не ставила под угрозу остальную часть разговора.

А что именно находится внутри `std.crypto.dh.X25519`? Никакой скрытой магии. Это две короткие функции, которые можно прочитать целиком в самой стандартной библиотеке Zig. Первая выводит публичный ключ из приватного — «g^a» обмена:

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

Говоря языком статьи: приватный ключ «умножается» — в эллиптическом смысле, а не в элементарном арифметическом — на базовую точку кривой Curve25519, и результат сериализуется в тридцать два байта. Операция `clampedMul` — это усиленная версия этого скалярного умножения: она включает в себя меры защиты, которые криптографическое сообщество добавляло годами, чтобы противостоять известным семействам атак. Две строчки тела функции.

Вторая функция комбинирует ваш приватный ключ с публичным ключом, который посылает вам другая сторона. Это «$(g^b)^a$» обмена, то, что создает общий секрет в тридцать два байта, который никто из вас никогда не передавал:

```
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();
}
```

Еще две строчки. Полученный публичный ключ интерпретируется как точка на кривой и «умножается» на собственный приватный ключ. Благодаря коммутативности операции над кривой — аналогичной коммутативности умножения степеней, которую мы видели в числовом примере, — обе стороны в конечном итоге получают одну и ту же сериализованную точку: именно тот общий секрет, о котором говорится в статье.

Вот и все. То, что в приложении кажется магией, на самом деле представляет собой две функции по три строчки каждая. Техническая сложность сконцентрирована в одной операции `clampedMul`, которая написана ниже в той же стандартной библиотеке, десятилетиями проверялась международным криптографическим сообществом и доступна любому, кто хочет прочитать ее буква за буквой. Черного ящика нет ни в нашем приложении, ни в стандартной библиотеке Zig. Есть открытый исходный код, который человек может понять, выбрав темп, в котором он хочет в него вникнуть.

Что защищает сквозное шифрование

Что E2EE защищает хорошо, при условии правильной реализации, — это содержимое сообщения при транзите. Промежуточный сервер, принимающий и пересылающий зашифрованные данные, увидит последовательность непонятных байтов. Злоумышленник с доступом к кабелю, роутеру, точке доступа Wi-Fi увидит то же самое. Провайдер услуг, сохраняющий копии трафика, не сможет прочитать его впоследствии. Правительство, приказавшее оператору связи выдать содержимое, получит те же непонятные байты, которые были у сервера изначально.

Это, в практическом плане, очень много. Это разница между написанием письма внутри непрозрачного конверта и написанием его на открытке. И то, и другое доходит до адресата. Но только одно сохраняет содержимое в тайне от почтальона.

Что не защищает сквозное шифрование

Это стоит знать так же хорошо. E2EE не защищает метаданные: сервер по-прежнему знает, что пользователь А отправляет данные пользователю Б, в какое время, с какой частотой и откуда, хотя и не знает, что именно он говорит. Эти метаданные, как мы уже аргументировали в статье *Шифровать — не значит быть приватным*, часто более показательны, чем сам контент. Знание того, что кто-то звонил в адвокатскую контору, специализирующуюся на разводах, в пятницу в 22:00 в течение тридцати минут, рассказывает историю, которую содержание звонка никогда бы не рассказало. Это та же ситуация, что и наблюдение за человеком, несколько раз входящим и выходящим из онкологической клиники: не нужно слышать ничего из того, о чем говорят внутри, чтобы представить, что происходит. Один изолированный метаданный может ничего не значить; несколько перекрестных данных рисуют нечто слишком похожее на правду. E2EE не защищает оконечные устройства: если устройство получателя скомпрометировано вредоносной программой, сообщение расшифровывается для этого получателя в обычном режиме, и вредоносная программа его читает. E2EE не защищает от подмены личности собеседника как таковой: если Алиса верит, что разговаривает с Бруно, но злоумышленник вклинился в самом начале (атака *man in the middle*), а протокол не предусматривает независимой проверки, обе стороны в итоге разговаривают с незваным гостем, думая, что говорят друг с другом.

Есть четвертая вещь, которую стоит сформулировать недвусмысленно. E2EE не мешает провайдеру, утверждающему, что он его предлагает, дополнительно сохранять копию незашифрованного сообщения в своих собственных системах. Утверждение «мои сообщения зашифрованы сквозным шифрованием» и утверждение «провайдер не сохраняет мой контент» — не одно и то же. Приложение может выполнять первое, нарушая второе; мы неоднократно видели это в заголовках газет с 2018 года. Пользователь, если только код клиента не является проверяемым, не имеет технического способа отличить один случай от другого без экспертного расследования. Самый известный случай среди широкой публики: WhatsApp шифрует сообщения сквозным шифрованием при транзите, но если пользователь активирует резервное копирование в iCloud или Google Drive без дополнительного шифрования, эта копия сохраняется в читаемом виде в инфраструктуре третьей стороны, и шифрование нарушается на стороне самого пользователя.

Вопрос, который оператор не хочет слышать

Приложение, утверждающее, что оно шифрует сквозным шифрованием, технически может делать одну из трех вещей в отношении ключей:

1. **Ключи находятся только на устройствах.** Они генерируются и находятся исключительно на устройствах пользователей; оператор их не знает и не хранит. Это оптимальный вариант.
2. **Оператор может получить доступ, если захочет.** Оператор владеет ключами пользователей (или может генерировать их по своему желанию) и хранит их в своих базах данных. Если он захочет или будет вынужден, он сможет прочитать содержимое. Это характерно для большинства «облачных» сервисов.
3. **Оператор не может получить доступ по дизайну, но он контролирует доступ.** У оператора нет ключей, но он контролирует приложение, которое их генерирует. Если его заставят, он может отправить вредоносное обновление, которое перехватит ключи или контент до шифрования. Это характерно для многих коммерческих сервисов E2EE.

Следовательно, оперативный вопрос заключается не в том, зашифровано ли что-то, а в том, кто контролирует устройство и программное обеспечение, управляющее ключами. В Solo2 ключи хранятся исключительно в вашем «Сейфе» (IndexedDB, зашифрованная вашим паролем), а программное обеспечение представляет собой проверяемый открытый исходный код.

Для профессионального читателя

Сквозное шифрование — это инструмент цифрового суверенитета. Но, как и любой инструмент, его эффективность зависит от руки, которая его держит, и почвы, на которой он стоит.

1. Где генерируются криптографические ключи и где они физически находятся? Если оператор может получить к ним доступ (даже временно, даже под предлогом восстановления), то E2EE является лишь номинальным.
2. Существует ли независимая проверка собеседника (коды безопасности, QR-коды, сравнение вне канала), которая предотвращает атаку man-in-the-middle во время установления разговора?
3. Проверяем ли код клиента — открыт ли он, опубликован, воспроизводим — или же он требует доверия слову провайдера о том, что клиент делает на самом деле?
4. Какие метаданные генерирует и хранит сервис, и как долго? Даже если контент непрозрачен, метаданные могут восстановить значительную часть конфиденциальной информации.

Эти четыре вопроса не требуют сложной технической информации; они запрашивают информацию, на которую любой честный оператор может ответить в своей публичной документации. Качество и точность ответа говорят о продукте не меньше, чем сам ответ.

Сквозное шифрование, если оно выполнено правильно, является одной из самых изящных конструкций, которые современная криптография принесла в повседневную практику. Оригинальная идея — два человека могут договориться о секрете через открытый канал — принадлежит Whitfield Diffie и Martin Hellman, 1976 год; полвека спустя мы продолжаем жить в ее последствиях. Но, как и в случае с любым техническим обещанием, его ценность зависит от реального исполнения, а не от ярлыка. Вопрос честного профессионала не в том, «зашифровано ли это?», а в том, «у кого ключи?». Ответы имеют разные последствия. Их стоит знать.

Источники и дополнительная литература

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, ноябрь 1976 г. Основополагающая статья по криптографии с открытым ключом.

- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, публичная спецификация от Open Whisper Systems, редакция 2016 г. Основа протокола Signal и его промышленных производных.
- RFC 7748 — Elliptic Curves for Security (IETF, январь 2016 г.). Нормативная спецификация кривых X25519 и X448, используемых в современных обменах ключами.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Главы об обмене ключами и протоколах аутентифицированного шифрования.
- Регламент (ЕС) 2024/1183 о европейской системе цифровой идентификации (eIDAS 2) — устанавливает рамки, в которых независимая проверка собеседника приобретает институциональную поддержку, и где различие между номинальным и реальным шифрованием имеет разные юридические последствия.

← ПредыдущийKill switch и институциональный захватСледующий →Бизнес-модель как сигнал доверия

Недавние материалы

- Анализ · 18 мая 2026 г. Реальная vs мнимая конфиденциальность: вопросы, которые стоит себе задать
- Анализ · 18 мая 2026 г. Self-hosting как профессиональная практика
- Концепция · 18 мая 2026 г. 24 слова: что такое криптографическая идентичность

Возьмите эту статью с собой туда, где она вам понадобится.

↓ Markdown ↓ Простой текст ↓ PDF

Файл будет загружен на ваше устройство. Оттуда вы можете сохранить его, импортировать в Solo2 или поделиться им где угодно. Cuadernos не решает место назначения за вас.

Сургучная печать · SHA-256 3644891a667ffc6b35ebaeddada005125f517b32cd1926c56ebf124381c352ff

Cuadernos Lacre · Издание Menzuri Gestión S.L. ·
текст R.Eugenio · под редакцией команды Solo2.