Criptografia de ponta a ponta, explicada de verdade

O que os fornecedores dizem quando dizem E2EE, e o que não dizem. Uma explicação didática do mecanismo e dos seus limites, sem o invólucro publicitário.

Para sermos claros: O WhatsApp diz que as tuas mensagens são criptografadas de ponta a ponta. É verdade — e não é suficiente. Se o backup for para o iCloud ou Google Drive sem criptografia adicional, a criptografia é quebrada no teu próprio telemóvel. A questão operacional não é se está criptografado, mas onde residem as chaves.

O que criptografar significa, de verdade

Criptografar uma mensagem significa transformá-la em algo que parece ruído para qualquer pessoa que não possua uma determinada informação chamada chave. A operação é feita no dispositivo de quem envia e, com a chave correta, é desfeita no dispositivo de quem recebe. No meio, a mensagem viaja como uma sucessão de bytes sem significado aparente. Essa é a ideia simples. O resto do artigo trata das nuances que a convertem, dependendo do caso, numa garantia real ou numa etiqueta de mercado.

O adjetivo *de ponta a ponta* — em inglês *end-to-end*, abreviado E2EE — acrescenta uma precisão. A criptografia não é feita para que um servidor intermediário possa lê-la e entregá-la. É feita para que apenas as duas extremidades — o dispositivo de quem envia e o dispositivo de quem recebe — possuam a chave. Qualquer servidor pelo qual a mensagem passe vê o ruído, não a mensagem. Essa é a diferença técnica com a criptografia *em trânsito*, onde o conteúdo viaja criptografado de um servidor para o outro, mas cada servidor pelo qual passa o decifra para o reenviar, recuperando temporariamente o texto em claro.

O paradoxo do segredo compartilhado

Há um problema óbvio. Para que duas pessoas possam criptografar e decriptografar mensagens entre si, ambas precisam da mesma chave. Mas, como entram em acordo sobre essa chave se tudo o que enviam uma à outra, por definição, passa por um canal onde alguém poderia estar a ouvir? Acordar a chave no mesmo canal onde depois a usarão parece impossível: se o atacante a ouvir ao acordá-la, poderá decifrar tudo o que vier a seguir. Durante décadas, a criptografia clássica resolveu isso pela via dura: as chaves eram entregues em pessoa, antes de começarem a ser usadas, em encontros físicos. Os embaixadores carregavam maletas de chaves cosidas ao forro do sobretudo.

No correio eletrónico contemporâneo, essa solução não escala. Se tivéssemos de ir fisicamente a casa de cada pessoa com quem pretendêssemos comunicar de forma criptografada, não chegaríamos a falar com ninguém. A pergunta colocada há cinquenta anos pela comunidade criptográfica foi esta: será possível que duas pessoas que não se conhecem e que apenas partilham um canal público acordem, nesse mesmo canal público, um segredo que ninguém que ouça o canal possa conhecer?

A elegância de Diffie-Hellman

Em 1976, dois matemáticos chamados Whitfield Diffie e Martin Hellman demonstraram algo aparentemente impossível: que duas pessoas, falando apenas por um canal público — um canal onde qualquer pessoa pode ouvir tudo o que dizem —, podem pôr-se de acordo numa palavra-passe secreta sem que nenhum ouvinte a possa descobrir. Soa a magia. Não é: é matemática. A troca de chaves Diffie-Hellman, como é conhecida desde então, é a base de praticamente toda a comunicação criptografada da internet, e meio século de uso intensivo e escrutínio académico mundial atestam a sua solidez. Quem quiser ver a intuição visual ou a matemática pode continuar a ler. Quem preferir confiar que funciona também pode continuar sem perder o fio do artigo.

Para quem quiser intuir isso numa imagem, há uma analogia conhecida com cores. Imagine que a Alice e o Bruno acordam abertamente uma cor base — digamos amarelo — à vista da Eva, que os ouve. Cada um escolhe em privado uma segunda cor secreta e mistura o seu segredo com o amarelo. A Alice obtém um laranja particular; o Bruno obtém um verde particular. Trocam os resultados à vista da Eva. Agora cada um mistura a cor recebida com o seu próprio segredo, e ambos chegam à mesma cor final, porque a ordem das misturas não importa. A Eva viu o amarelo e as duas misturas intermédias, mas não os segredos; sem algum dos segredos não pode chegar à cor final. A matemática real troca as cores por exponenciações em grupos modulares ou curvas elípticas, mas a ideia é a mesma: o segredo compartilhado é construído em público sem que ninguém no canal o possa reconstruir.

Em aritmética, para quem preferir ver o mecanismo: A Alice escolhe um número secreto a, o Bruno escolhe b. Trocam g^a e g^b abertamente sobre o canal. A Alice calcula $(g^b)^a$ e o Bruno calcula $(g^a)^b$; ambos chegam ao mesmo g^{ab}. A Eva vê g, g^a e g^b passar pelo canal, mas recuperar a a partir de g^a — o chamado problema do logaritmo discreto — requer um tempo de computação astronomicamente superior à idade do universo quando g é escolhido num grupo matemático adequado.

Para quem quiser comprovar com números pequenos. A troca Diffie-Hellman pode ser percorrida na íntegra com números suficientemente pequenos para fazer as contas à mão. Quem preferir não entrar em aritmética pode saltar este bloco sem perder o fio do artigo; quem quiser ver o

mecanismo a funcionar passo a passo vai encontrá-lo aqui. **As regras públicas**, que qualquer um pode ler: um primo $p = 11$ (no Diffie-Hellman real é de umas trezentas cifras; usamos onze para que as contas caibam numa página), uma base $g = 2$, e a convenção de que toda a aritmética se faz *módulo p* — calcula-se, divide-se por *p*, e conserva-se o resto, como um relógio de onze posições que volta ao zero ao passar do dez. **As escolhas privadas**, uma de cada e nunca partilhadas: a Alice escolhe $a = 4$. O Bruno escolhe $b = 7$.

Passo 1. A Alice calcula $2^4 = 16$, depois $16 \bmod 11 = \mathbf{5}$. Envia o cinco. A Eva anota-o.

Passo 2. O Bruno calcula $2^7 = 128$, depois $128 \bmod 11 = \mathbf{7}$. Envia o sete. A Eva também o anota. Após os dois envios, o caderno da Eva contém quatro dados: $p = 11$, $g = 2$, A = 5, B = 7. Falta-lhe o número partilhado que a Alice e o Bruno estão prestes a derivar — e que a Eva não poderá reconstruir.

Passo 3. A Alice pega no sete que o Bruno lhe enviou e eleva-o ao seu expoente privado $a = 4$. Para evitar manusear $7^4 = 2401$, calcula-se por partes aplicando o módulo em cada passo:

$7^2 = 49$

$49 \bmod 11 = 5$

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

$25 \bmod 11 = 3$

A Alice obtém o número **3**.

Passo 4. O Bruno pega no cinco que a Alice lhe enviou e eleva-o ao seu expoente privado $b = 7$. De novo por partes:

$5^2 = 25 \bmod 11 = 3$

$5^4 = (5^2)^2 = 3^2 = 9 \bmod 11 = 9$

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

Finalmente $5^7 = 5^6 \times 5 = 5 \times 5 = 25 \bmod 11 = \mathbf{3}$.

O Bruno obtém também **3**.

Os dois chegaram ao mesmo número, 3, trabalhando em paralelo. Nenhum deles enviou o seu expoente privado em momento algum. A Alice não sabe que $b = 7$; o Bruno não sabe que $a = 4$. Cada um usou o valor público que o outro enviou combinado com o seu próprio expoente privado, e encontraram-se no mesmo destino. **Por que chegam ao mesmo número?** O que cada um calculou: a Alice, $(g^b)^a = 2^{7\times4} = 2^{28} \bmod 11$. O Bruno, $(g^a)^b = 2^{4\times7} = 2^{28} \bmod 11$. É a mesma quantidade porque a ordem de multiplicação de expoentes não importa ($7 \times 4 = 4 \times 7$). Cada um chegou por um caminho diferente ao mesmo destino.

E a Eva? Tem no seu caderno $p = 11$, $g = 2$, A = 5, B = 7, e queria o 3. Para o calcular precisaria de conhecer *a* ou *b* — mas nenhum dos dois viajou pelo canal. A sua única via é perguntar-se: «para que expoente *a* se cumpre $2^a \bmod 11 = 5$?». Com *p* tão pequeno pode tentar 0, 1, 2, 3, 4... e encontrá-lo em menos de um minuto. Mas ao substituir 11 por um primo de trezentas cifras, o espaço de expoentes possíveis tem mais elementos do que átomos há no universo observável. **Não existe hoje em dia nenhum algoritmo conhecido pela humanidade que possa percorrer esse espaço em menos de milhares de milhões de anos.** É o chamado *problema do logaritmo discreto*: fácil para a frente, computacionalmente impossível para trás. E é a razão pela qual a criptografia resiste mesmo que a Eva tenha seguido toda a conversa letra por letra.

Três ingredientes simples —aritmética sobre um relógio, exponenciação, e comutatividade da multiplicação ($a \cdot b = b \cdot a$)— combinados produzem um protocolo do qual meia humanidade depende todos os dias para as suas comunicações privadas. Nenhuma das três peças, separadamente, parece especial. O que é decisivo é a montagem.

De Diffie-Hellman ao protocolo Signal

A criptografia de ponta a ponta que as aplicações de mensagens profissionais usam hoje descansa, quase sem exceção, sobre uma versão elegante e endurecida da troca Diffie-Hellman. O protocolo Signal, desenhado por Trevor Perrin e Moxie Marlinspike entre 2013 e 2016, é a referência. Combina duas ideias-chave. A primeira, a troca de chaves em curvas elípticas (X25519), que produz o segredo compartilhado inicial entre dois dispositivos. A segunda, o chamado Double Ratchet — engrenagem dupla —, que renova as chaves automaticamente com cada mensagem, de modo que comprometer o dispositivo hoje não permite decifrar mensagens passadas, nem mensagens futuras uma vez que a engrenagem tenha rodado.

Em Zig, a troca X25519 que produz o segredo compartilhado entre dois dispositivos cabe em seis linhas, usando a biblioteca padrão:

```
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)
```

O que acontece nessas seis linhas: As chaves públicas viajam abertamente. As chaves privadas não saem nunca do dispositivo respetivo. Cada parte deriva, a partir da sua privada e da pública da outra, um mesmo segredo de trinta e dois bytes que ninguém no canal pode recuperar. Esse segredo serve depois como semente para criptografar as mensagens trocadas. O Double Ratchet do protocolo Signal adiciona uma rotação constante desse material para que o compromisso de um instante não comprometa o resto da conversa.

E o que há exatamente dentro de `std.crypto.dh.X25519`? Nenhuma magia oculta. São duas funções curtas que se podem ler inteiras na própria biblioteca padrão do Zig. A primeira deriva a chave pública a partir da privada — o «g^a» da troca:

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

Na linguagem do artigo: a chave privada é «multiplicada» — no sentido elíptico, não no aritmético elementar — pelo ponto base da curva Curve25519, e o resultado é serializado em trinta e dois bytes. A operação `clampedMul` é a versão endurecida dessa multiplicação escalar: incorpora as salvaguardas que a comunidade criptográfica foi adicionando ao longo de anos para resistir a famílias conhecidas de ataques. Duas linhas de corpo de função.

A segunda função combina a tua chave privada com a chave pública que a outra parte te envia. É o «$(g^b)^a$» da troca, o que produz o segredo partilhado de trinta e dois bytes que nenhum dos dois chegou a transmitir:

```
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();
}
```

Outras duas linhas. A chave pública recebida é interpretada como um ponto sobre a curva, e «multiplicada» pela própria chave privada. Pela comutatividade da operação de curva — análoga à comutatividade da multiplicação de expoentes que vimos no exemplo numérico —, ambas as partes acabam com o mesmo ponto serializado: exatamente o segredo partilhado de que fala o artigo.

Isso é tudo. O que numa aplicação parece magia é, na realidade, duas funções de três linhas cada. A complexidade técnica concentra-se numa única operação, `clampedMul`, que está escrita mais à frente na mesma biblioteca padrão, revista durante décadas pela comunidade criptográfica internacional, e disponível para qualquer um que queira lê-la letra por letra. Não há caixa negra nem na nossa aplicação nem na biblioteca padrão do Zig. Há código de fonte aberta que um ser humano pode entender, escolhendo o ritmo ao qual se quer aprofundar.

O que a criptografia de ponta a ponta protege

O que o E2EE protege bem, assumindo uma implementação correta, é o conteúdo da mensagem em trânsito. Um servidor intermediário que receba e reenvie os dados criptografados verá uma sucessão de bytes ininteligíveis. Um atacante com acesso ao cabo, ao router, ao ponto de acesso wifi, verá o mesmo. Um fornecedor do serviço que conserve cópias do tráfego não poderá lê-lo a posteriori. Um Governo que ordene ao operador do serviço a entrega do conteúdo receberá os mesmos bytes ininteligíveis que o servidor tinha em primeiro lugar.

Isto, em termos práticos, é muito. É a diferença entre escrever uma carta dentro de um envelope opaco e escrevê-la num postal. As duas chegam. Apenas uma preserva o conteúdo perante o carteiro.

O que a criptografia de ponta a ponta não protege

Convém sabê-lo tão bem quanto. O E2EE não protege os metadados: o servidor continua a saber que o utilizador A envia dados ao utilizador B, a que horas, com que frequência e de onde, embora não saiba o que diz. Estes metadados, como já argumentámos em *Criptografar não é ser privado*, são muitas vezes mais reveladores do que o conteúdo. Saber que alguém ligou para um escritório de advogados especializado em divórcios numa sexta-feira às 22:00 durante trinta minutos conta uma história que o conteúdo da chamada nunca contou. É a mesma situação que ver uma pessoa entrar e sair várias vezes de uma clínica oncológica: não é preciso ouvir nada do que se fala lá dentro para imaginar o que está a acontecer. Um único metadado solto pode não significar nada; vários cruzados entre si desenham algo demasiado parecido com a verdade. O E2EE não protege as extremidades: se o dispositivo do recetor estiver comprometido por um programa malicioso, a mensagem é decifrada normalmente para esse recetor e o programa malicioso lê-a. O E2EE não protege contra a identidade do interlocutor em si: se a Alice acredita estar a falar com o Bruno mas um atacante se interpôs no início (um *man in the middle*) e o protocolo não inclui verificação independente, as duas partes acabam a falar com o intruso pensando que falam entre si.

Há uma quarta coisa que convém formular sem ambiguidade. O E2EE não impede que um fornecedor que afirma oferecê-lo guarde, além disso, uma cópia da mensagem sem criptografar nos seus próprios sistemas. A afirmação «as minhas mensagens estão criptografadas de ponta a ponta» e a afirmação «o fornecedor não conserva o meu conteúdo» não são a mesma. Uma aplicação pode cumprir a primeira enquanto incumpre a segunda; vimo-lo em manchetes de imprensa repetidamente desde 2018. O utilizador, a menos que o código do cliente seja verificável, não tem forma técnica de distinguir um caso do outro sem investigação pericial. O caso mais conhecido no público em geral: o WhatsApp criptografa as mensagens de ponta a ponta em trânsito, mas se o utilizador ativar a cópia de segurança em iCloud ou Google Drive sem criptografia adicional, essa cópia é armazenada legível em infraestrutura de um terceiro, e a criptografia quebra-se na extremidade do próprio utilizador.

A pergunta que o operador não quer ouvir

Uma aplicação que afirma criptografar de ponta a ponta pode, tecnicamente, fazer uma de três coisas em relação às chaves:

1. **As chaves residem apenas nos dispositivos.** Geram-se e residem exclusivamente nos dispositivos dos utilizadores; o operador não as conhece nem as armazena. É o caso ótimo.
2. **O operador pode aceder se quiser.** O operador possui as chaves dos utilizadores (ou pode gerá-las como desejar) e guarda-as nas suas bases de dados. Se quiser ou for forçado, pode ler o conteúdo. Este é o caso da maioria dos serviços «na nuvem».
3. **O operador não pode aceder por design, mas controla o acesso.** O operador não possui as chaves, mas tem o controlo da aplicação que as gera. Se for forçado, pode enviar uma atualização maliciosa que capture as chaves ou o conteúdo antes da criptografia. Este é o caso de muitos serviços E2EE comerciais.

A pergunta operacional, portanto, não é se algo está criptografado, mas quem tem o controlo do dispositivo e do software que gere as chaves. No Solo2, as chaves residem unicamente na tua Bóveda (IndexedDB criptografada com a tua palavra-passe) e o software é código aberto verificável.

Para o leitor profissional

A criptografia de ponta a ponta é uma ferramenta de soberania digital. Mas, como toda a ferramenta, a sua eficácia depende da mão que a empunha e do solo em que se apoia.

1. Onde são geradas as chaves criptográficas e onde residem fisicamente? Se o operador puder aceder a elas (mesmo temporariamente, mesmo sob o pretexto de recuperação), o E2EE é apenas nominal.
2. Existe verificação independente do interlocutor (números de segurança, códigos QR, comparação out-of-band) que previna um ataque man-in-the-middle durante o estabelecimento da conversa?
3. O código do cliente é auditável — aberto, publicado, reprodutível — ou exige confiar na palavra do fornecedor sobre o que o cliente realmente faz?
4. Quais metadados o serviço gera e mantém, e por quanto tempo? Mesmo que o conteúdo seja opaco, os metadados podem reconstruir boa parte da informação sensível.

Estas quatro perguntas não pedem informações técnicas avançadas; pedem informações às quais qualquer operador honesto pode responder na sua documentação pública. A qualidade e precisão da resposta diz tanto sobre o produto quanto a própria resposta.

A criptografia de ponta a ponta, bem feita, é uma das construções mais finas que a criptografia contemporânea entregou à prática quotidiana. A ideia original — duas pessoas podem acordar um segredo num canal público — pertence a Whitfield Diffie e Martin Hellman, 1976; meio século depois continuamos a viver na sua consequência. Mas, como acontece com qualquer promessa técnica, o seu valor depende do cumprimento real, não da etiqueta. A pergunta do profissional honesto não é «está criptografado?», mas «quem tem as chaves?». As respostas têm consequências distintas. Convém sabê-las.

Fontes e leitura adicional

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, novembro de 1976. Artigo fundamental da criptografia de chave pública.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, especificação pública da Open Whisper Systems, revisão de 2016. Base do protocolo Signal e dos seus derivados industriais.
- RFC 7748 — Elliptic Curves for Security (IETF, janeiro de 2016). Especificação normativa das curvas X25519 e X448 usadas nas trocas de chaves modernas.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Capítulos sobre troca de chaves e protocolos de criptografia autenticados.
- Regulamento (UE) 2024/1183 relativo a um quadro europeu para a identidade digital (eIDAS 2) — estabelece quadros onde a verificação independente do interlocutor ganha apoio institucional, e onde a distinção entre criptografia nominal e real tem consequências legais diferentes.

← AnteriorKill switch e a captura institucionalSeguinte →O modelo de negócio como sinal de confiança

Leituras recentes

- Análise · 18 de maio de 2026 Privacidade real vs aparente: as perguntas que convém fazer-se
- Análise · 18 de maio de 2026 Self-hosting como prática profissional
- Conceito · 18 de maio de 2026 As 24 palavras: o que é uma identidade criptográfica

Leve este artigo para onde precisar.

↓ Markdown ↓ Texto simples ↓ PDF

O arquivo é descarregado no seu dispositivo. A partir daí, pode guardá-lo, importá-lo no Solo2 ou partilhá-lo onde quiser. Cuadernos não decide o destino por si.

Selo de lacre · SHA-256 cf3d67b624e591ef50f6cf447b158a8231cdce2b777b3d48fc7c944cfd69e4f8

Cuadernos Lacre · Uma publicação da Menzuri Gestión S.L. ·
escrita por R.Eugenio · editada pela equipa do Solo2.