Cifrado de extremo a extremo, explicado de verdade

O que din os provedores cando din E2EE, e o que non din. Unha explicación didáctica do mecanismo e os seus límites, sen o envoltorio publicitante.

Para entendernos: WhatsApp di que as túas mensaxes están cifradas de extremo a extremo. É verdade — e non é suficiente. Se a copia de seguridade vai a iCloud ou Google Drive sen cifrado adicional, o cifrado rompe no teu propio teléfono. A pregunta operativa non é se está cifrado, senón onde residen as claves.

O que cifrar significa, de verdade

Cifrar unha mensaxe é transformala en algo que pareza ruído para calquera que non posúa certa información chamada clave. A operación faise no dispositivo de quen envía e, coa clave correcta, desfaise no dispositivo de quen recibe. No medio, a mensaxe viaxa como unha sucesión de bytes sen significado aparente. Esa é a idea sinxela. O resto do artigo ocúpase dos matices que a converten, segundo o caso, nunha garantía real ou nunha etiqueta de mercado.

O adxectivo *de extremo a extremo* —en inglés *end-to-end*, abreviado E2EE— engade unha precisión. O cifrado non se fai para que un servidor intermedio poida lelo e entregalo. Faise para que só os dous extremos —o dispositivo de quen envía e o dispositivo de quen recibe— posúan a clave. Calquera servidor polo que a mensaxe pase ve o ruído, non a mensaxe. Esa é a diferenza técnica co cifrado *en tránsito*, onde o contido vai cifrado dun servidor ao seguinte, pero cada servidor polo que pasa descífrao para reenvialo, recuperando temporalmente o texto en claro.

O paradoxo do segredo compartido

Hai un problema obvio. Para que dúas persoas poidan cifrar e descifrar mensaxes entre si, ambas necesitan a mesma clave. Pero, como se poñen dacordo nesa clave se todo o que se mandan, por definición, pasa por unha canle onde alguén podería estar escoitando? Acordar a clave na mesma canle onde despois a usarán parece imposible: se o atacante a escoita ao acordala, poderá descifrar todo o posterior. Durante decenios, a criptografía clásica resolveu isto pola vía dura: as claves entregábanse en persoa, antes de empezar a usarse, en encontros físicos. Os embaixadores cargaban con maletíns de claves cosidos ao forro do abrigo.

No correo electrónico contemporáneo, esa solución non escala. Se tivésemos que ir fisicamente á casa de cada persoa coa que pretendésemos comunicarnos de forma cifrada, non chegariamos a falar con ninguén. A pregunta formulada hai cincuenta anos pola comunidade criptográfica era esta: é posible que dúas persoas que non se coñecen e que só comparten unha canle pública acorden, nesa mesma canle pública, un segredo que ninguén que escoite a canle poida coñecer?

A elegancia de Diffie-Hellman

En 1976, dous matemáticos chamados Whitfield Diffie e Martin Hellman demostraron algo aparentemente imposible: que dúas persoas, falando só por unha canle pública —unha canle onde calquera pode escoitar todo o que din—, poden poñerse dacordo nun contrasinal segredo sen que ningún oínte poida descubrilo. Soa a maxia. Non o é: é matemática. O intercambio de claves Diffie-Hellman, como se coñece desde entón, é a base de practicamente toda a comunicación cifrada de internet, e medio século de uso intensivo e escrutinio académico mundial avalan a súa solidez. Quen queira ver a intuición visual ou a matemática pode seguir lendo. Quen prefira confiar en que funciona tamén pode continuar sen perder o fío do artigo.

Para quen queira intuílo nunha imaxe, hai unha analoxía coñecida con cores. Imaxina que Alicia e Bruno acordan en aberto unha cor base —digamos amarelo— á vista de Eva, que os escoita. Cada un elixe en privado unha segunda cor segreda e mestura o seu segredo co amarelo. Alicia obtén un laranxa particular; Bruno obtén un verde particular. Intercambian os resultados á vista de Eva. Agora cada un mestura a cor recibida co seu propio segredo, e ambos chegan á mesma cor final, porque a orde das mesturas non importa. Eva viu o amarelo e as dúas mesturas intermedias, pero non os segredos; sen algún dos segredos non pode chegar á cor final. A matemática real cambia as cores por exponenciacións en grupos modulares ou curvas elípticas, pero a idea é a mesma: o segredo compartido constrúese en público sen que ninguén na canle poida reconstruílo.

En aritmética, para quen prefira ver o mecanismo: Alicia elixe un número segredo a, Bruno elixe b. Intercambian g^a e g^b en aberto sobre a canle. Alicia calcula $(g^b)^a$ e Bruno calcula $(g^a)^b$; ambos chegan ao mesmo g^{ab}. Eva ve g, g^a e g^b pasar pola canle, pero recuperar a desde g^a —o chamado problema do logaritmo discreto— require un tempo de cómputo astronomicamente superior á idade do universo cando g se elixe nun grupo matemático adecuado.

Para quen queira comprobalo con números pequenos. O intercambio Diffie-Hellman pódese percorrer enteiro con cifras o bastante reducidas como para facer as contas a man. Quen prefira non entrar en aritmética pode saltarse este bloque sen perder o fío do artigo; quen queira ver o mecanismo funcionando paso a paso atoparao aquí. **As regras públicas**, que calquera pode ler: un primo $p = 11$ (no Diffie-Hellman real é

dunhas trescentas cifras; usamos once para que as contas caiban nunha páxina), unha base $g = 2$, e a convención de que toda a aritmética faise *módulo p* — calcúlase, divídese entre *p*, e consérvase o resto, como un reloxo de once posicións que volve ao cero ao pasar do dez. **As eleccións privadas**, unha cada un e nunca compartidas: Alicia elixe $a = 4$. Bruno elixe $b = 7$.

Paso 1. Alicia calcula $2^4 = 16$, despois 16 mod 11 = **5**. Envía o cinco. Eva anótao.

Paso 2. Bruno calcula $2^7 = 128$, despois 128 mod 11 = **7**. Envía o sete. Eva tamén o anota. Tras os dous envíos, a libreta de Eva contén catro datos: $p = 11$, $g = 2$, A = 5, B = 7. Fáltalle o número compartido que Alicia e Bruno están a piques de derivar — e que Eva non poderá reconstruír.

Paso 3. Alicia colle o sete que lle enviou Bruno e elévao ao seu exponente privado $a = 4$. Para evitar manexar $7^4 = 2401$, calcúlase por partes aplicando o módulo en cada paso:

$7^2 = 49$

49 mod 11 = 5

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

25 mod 11 = 3

Alicia obtén o número **3**.

Paso 4. Bruno colle o cinco que lle enviou Alicia e elévao ao seu exponente privado $b = 7$. De novo por partes:

$5^2 = 25$ 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

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

Bruno obtén tamén **3**.

Os dous chegaron ao mesmo número, 3, traballando en paralelo. Ningún deles enviou o seu exponente privado en ningún momento. Alicia non sabe que $b = 7$; Bruno non sabe que $a = 4$. Cada cal usou o valor público que o outro enviou combinado co seu propio exponente privado, e atopáronse no mesmo destino. **Por que chegan ao mesmo número?** O que calculou cada un: Alicia, $(g^b)^a = 2^{7\times4} = 2^{28}$ mod 11. Bruno, $(g^a)^b = 2^{4\times7} = 2^{28}$ mod 11. É a mesma cantidade porque a orde de multiplicación de exponentes non importa ($7 \times 4 = 4 \times 7$). Cada cal chegou por un camiño distinto ao mesmo destino.

E Eva? Ten na súa libreta $p = 11$, $g = 2$, A = 5, B = 7, e querería o 3. Para calculalo necesitaría coñecer *a* ou *b* — pero ningún dos dous viaxou pola canle. A súa única vía é preguntarse: «para que exponente *a* se cumpre 2^a mod 11 = 5?». Con *p* tan pequeno pode probar 0, 1, 2, 3, 4... e atopalo en menos dun minuto. Pero ao substituír 11 por un primo de trescentas cifras, o espazo de exponentes posibles ten máis elementos ca átomos hai no universo observable. **Non existe a día de hoxe ningún algoritmo coñecido pola humanidade que poida percorrer ese espazo en menos de miles de millóns de anos.** É o chamado *problema do logaritmo discreto*: fácil cara adiante, computacionalmente imposible cara atrás. E é a razón pola que o cifrado resiste aínda que Eva seguise toda a conversa letra por letra.

Tres ingredientes simples —aritmética sobre un reloxo, exponenciación, e conmutatividade da multiplicación ($a \cdot b = b \cdot a$)— combinados producen un protocolo do que media humanidade depende cada día para as súas comunicacións privadas. Ningunha das tres pezas, por separado, parece especial. O decisivo é a ensamblaxe.

De Diffie-Hellman ao protocolo Signal

O cifrado de extremo a extremo que usan hoxe as aplicacións de mensaxería profesional descansa, case sen excepción, sobre unha versión elegante e endurecida do intercambio Diffie-Hellman. O protocolo Signal, deseñado por Trevor Perrin e Moxie Marlinspike entre 2013 e 2016, é a referencia. Combina dúas ideas clave. A primeira, o intercambio de claves en curvas elípticas (X25519), que produce o segredo compartido inicial entre dous dispositivos. A segunda, o chamado Double Ratchet —dobre engrenaxe—, que renova as claves automaticamente con cada mensaxe, de modo que comprometer o dispositivo hoxe non permite descifrar mensaxes pasadas, nin mensaxes futuras unha vez se rotou a engrenaxe.

En Zig, o intercambio X25519 que produce o segredo compartido entre dous dispositivos cabe en seis liñas, usando a biblioteca estándar:

```
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 pasa nesas seis liñas: As claves públicas viaxan abertamente. As claves privadas non saen nunca do dispositivo respectivo. Cada parte deriva, a partir da súa privada e a pública da outra, un mesmo segredo de trinta e dous bytes que ninguén na canle pode recuperar. Ese segredo serve despois como semente para cifrar as mensaxes intercambiadas. O Double Ratchet do protocolo Signal engade unha rotación constante deste material para que o compromiso dun instante non comprometa o resto da conversa.

E dentro de `std.crypto.dh.X25519`, que hai exactamente? Non hai maxia oculta. Son dúas funcións curtas que se poden ler enteiras na propia biblioteca estándar de Zig. A primeira deriva a clave pública desde a privada — o «g^a» do intercambio:

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

Na linguaxe do artigo: a clave privada «multiplícase» —no sentido elíptico, non no aritmético elemental— polo punto base da curva Curve25519, e o resultado serialízase en trinta e dous bytes. A operación `clampedMul` é a versión endurecida desa multiplicación escalar: incorpora as salvagardas que a comunidade criptográfica foi engadindo ao longo de anos para resistir familias coñecidas de ataques. Dúas liñas de corpo de función.

A segunda función combina a túa clave privada coa clave pública que a outra parte che envía. É o «$(g^b)^a$» do intercambio, o que produce o segredo compartido de trinta e dous bytes que ningún dos dous 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 dúas liñas. A clave pública recibida interprétase como un punto sobre a curva, e «multiplícase» pola clave privada propia. Pola conmutatividade da operación de curva —análoga á conmutatividade da multiplicación de exponentes que vimos no exemplo numérico—, ambas partes acaban co mesmo punto serializado: exactamente o segredo compartido do que fala o artigo.

Iso é todo. O que nunha aplicación parece maxia é, na realidade, dúas funcións de tres liñas cada unha. A complexidade técnica concéntrase nunha soa operación, `clampedMul`, que está escrita máis adiante na mesma biblioteca estándar, revisada durante décadas pola comunidade criptográfica internacional, e dispoñible para calquera que queira lela letra por letra. Non hai caixa negra nin na nosa aplicación nin na biblioteca estándar de Zig. Hai código aberto que un humano pode entender, elixindo o ritmo ao que quere entrar nel.

Que protexe o cifrado de extremo a extremo

O que o E2EE protexe ben, asumindo unha implementación correcta, é o contido da mensaxe en tránsito. Un servidor intermedio que reciba e reenvíe os datos cifrados verá unha sucesión de bytes inintelixibles. Un atacante con acceso ao cable, ao router, ao punto de acceso wifi, verá o mesmo. Un provedor do servizo que conserve copias do tráfico non poderá lelo a posteriori. Un Goberno que ordene ao operador do servizo entregar o contido recibirá os mesmos bytes inintelixibles que tiña o servidor en primeiro lugar.

Isto, en termos prácticos, é moito. É a diferenza entre escribir unha carta dentro dun sobre opaco e escribila nunha postal. As dúas chegan. Só unha preserva o contido ante o carteiro.

Que non protexe o cifrado de extremo a extremo

Convén sabelo igual de ben. O E2EE non protexe os metadatos: o servidor segue sabendo que o usuario A envía datos ao usuario B, a que hora, con que frecuencia e desde onde, aínda que non saiba que di. Estes metadatos, xa o argumentamos en *Cifrar non é ser privado*, son a miúdo máis reveladores que o contido. Saber que alguén chamou a un despacho de avogados especializado en divorcios un venres ás 22:00 durante trinta minutos conta unha historia que o contido da chamada nunca contou. É a mesma situación que ver a unha persoa entrar e saír varias veces dunha clínica oncolóxica: non fai falta ouvir nada do que se fala dentro para imaxinar o que está pasando. Un só metadato solto pode non significar nada; varios cruzados entre si debuxan algo demasiado parecido á verdade. O E2EE non protexe os extremos: se o dispositivo do receptor está comprometido por un programa malicioso, a mensaxe descífrase normalmente para ese receptor e o programa malicioso léeo. O E2EE non protexe contra a identidade do interlocutor en si: se Alicia cre estar falando con Bruno pero un atacante interpúxose ao inicio (un *man in the middle*) e o protocolo non inclúe verificación independente, as dúas partes acaban falando co intruso pensando que falan entre si.

Hai unha cuarta cousa que convén formular sen ambigüidade. O E2EE non impide que un provedor que afirma ofrecelo garde, ademais, unha copia da mensaxe sen cifrar nos seus propios sistemas. A afirmación «as miñas mensaxes están cifradas de extremo a extremo» e a afirmación «o provedor non conserva o meu contido» non son a mesma. Unha aplicación pode cumprir a primeira mentres incumpre a segunda; vímolo en titulares de prensa repetidamente desde 2018. O usuario, salvo que o código do cliente sexa verificable, non ten forma técnica de distinguir un caso do outro sen investigación experta. O caso máis coñecido no público xeral: WhatsApp cifra as mensaxes de extremo a extremo en tránsito, pero se o usuario activa a copia de seguridade en iCloud ou Google Drive sen cifrado adicional, esa copia almacénase lexible en infraestrutura dun terceiro, e o cifrado rómprese no extremo do propio usuario.

A pregunta que o operador non quere oír

Unha aplicación que afirma cifrar de extremo a extremo pode, tecnicamente, facer unha de tres cousas con respecto ás claves:

1. **As claves residen só nos dispositivos.** Xéranse e residen exclusivamente nos dispositivos dos usuarios; o operador non as coñece nin as almacena. É o caso óptimo.
2. **O operador pode acceder se quere.** O operador ten as claves dos usuarios (ou pode xeralas ao seu antollo) e gárdaas nas súas bases de datos. Se quere ou se lle obriga, pode ler o contido. Este é o caso da maioría de servizos «na nube».
3. **O operador non pode acceder por deseño, pero controla o acceso.** O operador non ten as claves, pero ten o control da aplicación que as xera. Se se lle obriga, pode enviar unha actualización maliciosa que capture as claves ou o contido antes de cifrar. Este é o caso de moitos servizos E2EE comerciais.

A pregunta operativa, por tanto, non é se algo está cifrado, senón quen ten o control do dispositivo e do software que xestiona as claves. En Solo2, as claves residen unicamente na túa Bóveda (IndexedDB cifrada co teu contrasinal) e o software é código aberto verificable.

Para o lector profesional

O cifrado de extremo a extremo é unha ferramenta de soberanía dixital. Pero como toda ferramenta, a súa eficacia depende da man que a empuña e do chan no que se apoia.

1. Onde se xeran as claves criptográficas e onde residen fisicamente? Se o operador pode acceder a elas (mesmo temporalmente, mesmo baixo formulación de recuperación), o E2EE é nominal.
2. Existe verificación independente do interlocutor (números de seguridade, códigos QR, comparación fóra de banda) que impida un ataque de home no medio durante o establecemento da conversa?
3. O código do cliente é auditable —aberto, publicado, reproducible— ou esixe confiar na palabra do provedor sobre o que o cliente fai en realidade?
4. Que metadatos xera e conserva o servizo, e por canto tempo? Aínda que o contido sexa opaco, os metadatos poden reconstruír boa parte da información sensible.

Estas catro preguntas non piden información técnica avanzada; piden información que calquera operador honesto pode responder na súa documentación pública. A calidade e precisión da resposta di tanto do produto como a resposta mesma.

O cifrado de extremo a extremo, ben feito, é unha das construcións máis finas que a criptografía contemporánea entregou á práctica cotiá. A idea orixinal —dúas persoas poden acordar un segredo nunha canle pública— pertence a Whitfield Diffie e Martin Hellman, 1976; medio século despois seguimos vivindo na súa consecuencia. Pero, como sucede con calquera promesa técnica, o seu valor depende do cumprimento real, non da etiqueta. A pregunta do profesional honesto non é «está cifrado?», senón «quen ten as claves?». As respostas teñen consecuencias distintas. Conveñen sabelas.

Fontes e lectura adicional

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, novembro de 1976. Artigo fundacional da criptografía de clave pública.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, especificación pública de Open Whisper Systems, revisión de 2016. Base do protocolo Signal e os seus derivados industriais.
- RFC 7748 — Elliptic Curves for Security (IETF, xaneiro de 2016). Especificación normativa das curvas X25519 e X448 usadas en intercambios de clave modernos.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Capítulos sobre intercambio de claves e protocolos de cifrado autenticado.
- Regulamento (UE) 2024/1183 de espazo europeo de identidade dixital (eIDAS 2) — establece marcos onde a verificación independente do interlocutor adquire apoio institucional, e onde a distinción entre cifrado nominal e cifrado real ten consecuencias xurídicas distintas.

← AnteriorKill switch e a captura institucionalSeguinte →O modelo de negocio como sinal de confianza

Lecturas recentes

- Análise · 18 de maio de 2026 Privacidade real vs aparente: as preguntas que convén facerse
- Análise · 18 de maio de 2026 Self-hosting como práctica profesional
- Concepto · 18 de maio de 2026 As 24 palabras: que é unha identidade criptográfica

Leva este artigo onde o necesites.

↓ Markdown ↓ Texto plano ↓ PDF

O arquivo descárgase no teu dispositivo. Desde aí podes gardalo, importalo a Solo2, o compartilo onde queiras. Cuadernos no decide o destino por ti.

Selo de lacre · SHA-256 4ef0f95f3b0dcb91ae9f693c00445fe877d153df132b1d7510d47c9ffae5b308

Cuadernos Lacre · Unha publicación de Menzuri Gestión S.L. ·
escrita por R.Eugenio · editada polo equipo de Solo2.

Esta web no usa cookies e non carga recursos de terceiros. Usa un contador anónimo de visitas autohospedado (Umami, no noso servidor europeo) e o mínimo JavaScript necesario para os dous controis do cabezal: tema claro ou escuro, e selector de idioma. Sen trackers, sen perfilado, sen compartir datos. Se queres seguirnos: RSS.