Crittografia end-to-end, spiegata davvero

Cosa dicono i fornitori quando dicono E2EE, e cosa non dicono. Una spiegazione didattica del meccanismo e dei suoi limiti, senza l'involucro pubblicitario.

Per intenderci: WhatsApp dice che i tuoi messaggi sono crittografati end-to-end. È vero — e non è sufficiente. Se il backup va su iCloud o Google Drive senza crittografia aggiuntiva, la crittografia si rompe sul tuo stesso telefono. La domanda operativa non è se sia crittografato, ma dove risiedono le chiavi.

Cosa significa crittografare, davvero

Crittografare un messaggio significa trasformarlo in qualcosa che sembri rumore per chiunque non possieda una certa informazione chiamata chiave. L'operazione viene eseguita sul dispositivo di chi invia e, con la chiave corretta, viene annullata sul dispositivo di chi riceve. Nel mezzo, il messaggio viaggia come una successione di byte senza significato apparente. Questa è l'idea semplice. Il resto dell'articolo si occupa delle sfumature che la rendono, a seconda dei casi, una garanzia reale o un'etichetta di mercato.

L'aggettivo *end-to-end* — dall'inglese *end-to-end*, abbreviato E2EE — aggiunge una precisione. La crittografia non viene fatta affinché un server intermedio possa leggerla e consegnarla. Viene fatta affinché solo i due estremi — il dispositivo di chi invia e il dispositivo di chi riceve — possiedano la chiave. Qualsiasi server attraverso il quale il messaggio passa vede il rumore, non il messaggio. Questa è la differenza tecnica con la crittografia *in transito*, dove il contenuto viaggia crittografato da un server all'altro, ma ogni server attraverso il quale passa lo decrittografa per reinviarlo, recuperando temporaneamente il testo in chiaro.

Il paradosso del segreto condiviso

C'è un problema ovvio. Affinché due persone possano crittografare e decrittografare messaggi tra loro, entrambe hanno bisogno della stessa chiave. Ma come concordano questa chiave se tutto ciò che si inviano, per definizione, passa attraverso un canale dove qualcuno potrebbe essere in ascolto? Concordare la chiave nello stesso canale dove poi la useranno sembra impossibile: se l'attaccante la sente al momento dell'accordo, potrà decrittografare tutto il seguito. Per decenni, la crittografia classica ha risolto questo problema con le maniere forti: le chiavi venivano consegnate di persona, prima di iniziare a essere utilizzate, in incontri fisici. Gli ambasciatori portavano valigette di chiavi cucite nella fodera del cappotto.

Nella posta elettronica contemporanea, questa soluzione non è scalabile. Se dovessimo andare fisicamente a casa di ogni persona con cui intendiamo comunicare in modo crittografato, non riusciremmo a parlare con nessuno. La domanda posta cinquant'anni fa dalla comunità crittografica era questa: è possibile che due persone che non si conoscono e che condividono solo un canale pubblico concordino, su quello stesso canale pubblico, un segreto che nessuno in ascolto sul canale possa conoscere?

L'eleganza di Diffie-Hellman

Nel 1976, due matematici di nome Whitfield Diffie e Martin Hellman dimostrarono qualcosa di apparentemente impossibile: che due persone, parlando solo attraverso un canale pubblico — un canale dove chiunque può sentire tutto ciò che dicono —, possono concordare una password segreta senza che nessun ascoltatore possa scoprirla. Sembra magia. Non lo è: è matematica. Lo scambio di chiavi Diffie-Hellman, come è noto da allora, è la base di quasi tutta la comunicazione crittografata su Internet, e mezzo secolo di uso intensivo e di scrutinio accademico mondiale ne confermano la solidità. Chi vuole vedere l'intuizione visiva o la matematica può continuare a leggere. Chi preferisce fidarsi del fatto che funzioni può anche proseguire senza perdere il filo dell'articolo.

Per chi vuole intuirlo in un'immagine, c'è un'analogia nota con i colori. Immagina che Alice e Bruno concordino apertamente un colore di base — diciamo il giallo — sotto gli occhi di Eva, che li ascolta. Ognuno sceglie in privato un secondo colore segreto e mescola il proprio segreto con il giallo. Alice ottiene un arancione particolare; Bruno ottiene un verde particolare. Si scambiano i risultati sotto gli occhi di Eva. Ora ognuno mescola il colore ricevuto con il proprio segreto, ed entrambi arrivano allo stesso colore finale, perché l'ordine delle mescolanze non conta. Eva ha visto il giallo e le due mescolanze intermedie, ma non i segreti; senza uno dei segreti non può arrivare al colore finale. La matematica reale sostituisce i colori con elevamenti a potenza in gruppi modulari o curve ellittiche, ma l'idea è la stessa: il segreto condiviso viene costruito in pubblico senza che nessuno nel canale possa ricostruirlo.

In aritmetica, per chi preferisce vedere il meccanismo: Alice sceglie un numero segreto a, Bruno sceglie b. Si scambiano g^a e g^b in chiaro sul canale. Alice calcola $(g^b)^a$ e Bruno calcola $(g^a)^b$; entrambi arrivano allo stesso g^{ab}. Eva vede g, g^a e g^b passare attraverso il canale, ma recuperare a da g^a — il cosiddetto problema del logaritmo discreto — richiede un tempo di calcolo astronomicamente superiore all'età dell'universo quando g viene scelto in un gruppo matematico adeguato.

Para quien quiera comprobarlo con números pequeños. El intercambio Diffie-Hellman se puede recorrer entero con cifras lo bastante reducidas como para hacer las cuentas a mano. Quien prefiera no entrar en aritmética puede saltarse este bloque sin perder el hilo del artículo; quien quiera ver el mecanismo funcionando paso a paso lo encontrará aquí. **Las reglas públicas**, que cualquiera puede leer: un primo $p = 11$ (en el Diffie-Hellman real es de unas trescientas cifras; usamos once para que las cuentas quepan en una página), una base $g = 2$, y la convención de que toda la aritmética se hace *módulo p* — se calcula, se divide entre *p*, y se conserva el resto, como un reloj de once posiciones que vuelve al cero al rebasar el diez. **Las elecciones privadas**, una cada uno y jamás compartidas: Alicia elige $a = 4$. Bruno elige $b = 7$.

Paso 1. Alicia calcula $2^4 = 16$, luego $16 \bmod 11 = \mathbf{5}$. Envía el cinco. Eva lo anota.

Paso 2. Bruno calcula $2^7 = 128$, luego $128 \bmod 11 = \mathbf{7}$. Envía el siete. Eva también lo anota. Tras los dos envíos, la libreta de Eva contiene cuatro datos: $p = 11$, $g = 2$, A = 5, B = 7. Le falta el número compartido que Alicia y Bruno están a punto de derivar — y que Eva no podrá reconstruir.

Paso 3. Alicia toma el siete que Bruno le envió y lo eleva a su exponente privado $a = 4$. Para evitar manejar $7^4 = 2401$, se calcula por partes aplicando el módulo en cada paso:

$7^2 = 49$

$49 \bmod 11 = 5$

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

$25 \bmod 11 = 3$

Alicia obtiene el número **3**.

Paso 4. Bruno toma el cinco que Alicia le envió y lo eleva a su exponente privado $b = 7$. De nuevo 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 $5^7 = 5^6 \times 5 = 5 \times 5 = 25 \bmod 11 = \mathbf{3}$.

Bruno obtiene también **3**.

Los dos han llegado al mismo número, 3, trabajando en paralelo. Ninguno envió su exponente privado en ningún momento. Alicia no sabe que $b = 7$; Bruno no sabe que $a = 4$. Cada cual usó el valor público que el otro envió combinado con su propio exponente privado, y se encontraron en el mismo destino. **¿Por qué llegan al mismo número?** Lo que calculó cada uno: Alicia, $(g^b)^a = 2^{7 \times 4} = 2^{28} \bmod 11$. Bruno, $(g^a)^b = 2^{4 \times 7} = 2^{28} \bmod 11$. Es la misma cantidad porque el orden de multiplicación de exponentes no importa ($7 \times 4 = 4 \times 7$). Cada cual llegó por un camino distinto al mismo destino.

¿Y Eva? Tiene en su libreta $p = 11$, $g = 2$, A = 5, B = 7, y quisiera el 3. Para calcularlo necesitaría conocer *a* o *b* — pero ninguno ha viajado por el canal. Su única vía es preguntarse: «¿para qué exponente *a* se cumple $2^a \bmod 11 = 5$?». Con *p* tan pequeño puede probar 0, 1, 2, 3, 4… y encontrarlo en menos de un minuto. Pero al sustituir 11 por un primo de trescientas cifras, el espacio de exponentes posibles tiene más elementos que átomos hay en el universo observable. **No existe a día de hoy ningún algoritmo conocido por la humanidad que pueda recorrer ese espacio en menos de miles de millones de años.** Es el llamado *problema del logaritmo discreto*: fácil hacia adelante, computacionalmente imposible hacia atrás. Y es la razón por la que el cifrado resiste aunque Eva haya seguido toda la conversación letra por letra.

Tres ingredientes simples —aritmética sobre un reloj, exponenciación, y conmutatividad de la multiplicación ($a \cdot b = b \cdot a$)— combinados producen un protocolo del que media humanidad depende cada día para sus comunicaciones privadas. Ninguna de las tres piezas, por separado, parece especial. Lo decisivo es el ensamblaje.

Da Diffie-Hellman al protocollo Signal

La crittografia end-to-end utilizzata oggi dalle applicazioni di messaggistica professionale poggia, quasi senza eccezioni, su un'elegante e indurita versione dello scambio Diffie-Hellman. Il protocollo Signal, progettato da Trevor Perrin e Moxie Marlinspike tra il 2013 e il 2016, è il riferimento. Combina due idee chiave. La prima, lo scambio di chiavi in curve ellittiche (X25519), che produce il segreto condiviso iniziale tra due dispositivi. La seconda, il cosiddetto Double Ratchet — ingranaggio doppio —, che rinnova le chiavi automaticamente con ogni messaggio, in modo che la compromissione del dispositivo oggi non permetta di decrittografare i messaggi passati, né i messaggi futuri una volta che l'ingranaggio è stato ruotato.

In Zig, lo scambio X25519 che produce il segreto condiviso tra due dispositivi entra in sei righe, utilizzando la libreria standard:

```
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)
```

Cosa succede in quelle sei righe: Le chiavi pubbliche viaggiano apertamente. Le chiavi private non escono mai dal rispettivo dispositivo. Ogni parte deriva, a partire dalla propria chiave privata e dalla chiave pubblica dell'altra, uno stesso segreto di trentadue byte che nessuno nel canale può recuperare. Quel segreto serve poi come seme per crittografare i messaggi scambiati. Il Double Ratchet del protocollo Signal aggiunge una rotazione costante di quel materiale affinché la compromissione di un istante non comprometta il resto della conversazione.

E cosa c'è esattamente all'interno di `std.crypto.dh.X25519`? Nessuna magia nascosta. Sono due brevi funzioni che possono essere lette per intero nella libreria standard di Zig. La prima deriva la chiave pubblica da quella privata — il «g^a» dello scambio:

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

Nel linguaggio dell'articolo: la chiave privata viene «moltiplicata» — in senso ellittico, non in senso aritmetico elementare — per il punto base della curva Curve25519, e il risultato viene serializzato in trentadue byte. L'operazione `clampedMul` è la versione rafforzata di tale moltiplicazione scalare: incorpora le salvaguardie che la comunità crittografica ha aggiunto negli anni per resistere a famiglie di attacchi note. Due righe di corpo della funzione.

La seconda funzione combina la tua chiave privata con la chiave pubblica che l'altra parte ti invia. È il «$(g^b)^a$» dello scambio, quello che produce il segreto condiviso di trentadue byte che nessuno dei due ha mai trasmesso:

```
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();
}
```

Altre due righe. La chiave pubblica ricevuta viene interpretata come un punto sulla curva e «moltiplicata» per la propria chiave privata. Per la commutatività dell'operazione sulla curva — analoga alla commutatività della moltiplicazione degli esponenti che abbiamo visto nell'esempio numerico — entrambe le parti ottengono lo stesso punto serializzato: esattamente il segreto condiviso di cui parla l'articolo.

Questo è tutto. Ciò che in un'applicazione sembra magia è, in realtà, costituito da due funzioni di tre righe ciascuna. La complessità tecnica si concentra in un'unica operazione, `clampedMul`, scritta più avanti nella stessa libreria standard, revisionata per decenni dalla comunità crittografica internazionale e a disposizione di chiunque voglia leggerla lettera per lettera. Non esiste alcuna scatola nera né nella nostra applicazione né nella libreria standard di Zig. C'è codice open source che un essere umano può comprendere, scegliendo il ritmo al quale vuole addentrarvisi.

Cosa protegge la crittografia end-to-end

Ciò che l'E2EE protegge bene, assumendo un'implementazione corretta, è il contenuto del messaggio in transito. Un server intermedio che riceva e reinvii i dati crittografati vedrà una successione di byte inintelligibili. Un attaccante con accesso al cavo, al router, al punto di accesso wifi, vedrà lo stesso. Un fornitore di servizi che conservi copie del traffico non potrà leggerlo a posteriori. Un Governo che ordini all'operatore del servizio di consegnare il contenuto riceverà gli stessi byte inintelligibili che il server aveva in primo luogo.

Questo, in termini pratici, è molto. È la differenza tra lo scrivere una lettera dentro una busta opaca e lo scriverla su una cartolina. Entrambe arrivano. Solo una preserva il contenuto di fronte al postino.

Cosa non protegge la crittografia end-to-end

Conviene saperlo altrettanto bene. L'E2EE non protegge i metadati: il server sa ancora che l'utente A invia dati all'utente B, a che ora, con quale frequenza e da dove, anche se non sa cosa dice. Questi metadati, come abbiamo già argomentato in *Crittografare non significa essere privati*, sono spesso più rivelatori del contenuto. Sapere che qualcuno ha chiamato uno studio legale specializzato in divorzi un venerdì alle 22:00 per trenta minuti racconta una storia che il contenuto della chiamata non ha mai raccontato. È la stessa situazione di vedere una persona entrare e uscire più volte da una clinica oncologica: non c'è bisogno di sentire nulla di ciò che si dice all'interno per immaginare cosa stia succedendo. Un singolo metadato isolato può non significare nulla; diversi incrociati tra loro disegnano qualcosa di troppo simile alla verità. L'E2EE non protegge gli estremi: se il dispositivo del ricevente è compromesso da un programma dannoso, il messaggio viene decrittografato normalmente per quel ricevente e il programma dannoso lo legge. L'E2EE non protegge dall'identità dell'interlocutore in sé: se Alice crede di parlare con Bruno ma un attaccante si è interposto all'inizio (un *man in the middle*) e il protocollo non include una verifica indipendente, le due parti finiscono per parlare con l'intruso pensando di parlare tra loro.

C'è una quarta cosa che conviene formulare senza ambiguità. L'E2EE non impedisce che un fornitore che afferma di offrirlo conservi, inoltre, una copia del messaggio non crittografato nei propri sistemi. L'affermazione «i miei messaggi sono crittografati end-to-end» e l'affermazione «il fornitore non conserva il mio contenuto» non sono la stessa cosa. Un'applicazione può rispettare la prima mentre viola la seconda; lo abbiamo visto nei titoli di stampa ripetutamente dal 2018. L'utente, a meno che il codice del client non sia verificabile, non ha alcun modo tecnico di distinguere un caso dall'altro senza un'investigazione esperta. Il caso più noto al grande pubblico: WhatsApp crittografa i messaggi end-to-end in transito, ma se l'utente attiva la copia di sicurezza su iCloud o Google Drive senza crittografia aggiuntiva, quella copia viene memorizzata leggibile nell'infrastruttura di un terzo, e la crittografia si rompe all'estremo dell'utente stesso.

La domanda che l'operatore non vuole sentire

Un'applicazione che afferma di crittografare end-to-end può, tecnicamente, fare una di tre cose riguardo alle chiavi:

1. **Le chiavi risiedono solo sui dispositivi.** Vengono generate e risiedono esclusivamente sui dispositivi degli utenti; l'operatore non le conosce né le memorizza. È il caso ottimale.
2. **L'operatore può accedere se vuole.** L'operatore possiede le chiavi degli utenti (o può generarle a piacimento) e le conserva nei suoi database. Se vuole o se è costretto, può leggere il contenuto. Questo è il caso della maggior parte dei servizi «cloud».
3. **L'operatore non può accedere per progettazione, ma controlla l'accesso.** L'operatore non ha le chiavi, ma ha il controllo dell'applicazione che le genera. Se costretto, può inviare un aggiornamento malevolo che catturi le chiavi o il contenuto prima della crittografia. Questo è il caso di molti servizi E2EE commerciali.

La domanda operativa, quindi, non è se qualcosa sia crittografato, ma chi abbia il controllo del dispositivo e del software che gestisce le chiavi. In Solo2, le chiavi risiedono unicamente nella tua Bóveda (IndexedDB crittografata con la tua password) e il software è codice sorgente aperto verificabile.

Per il lettore professionale

La crittografia end-to-end è uno strumento di sovranità digitale. Ma come ogni strumento, la sua efficacia dipende dalla mano che lo impugna e dal suolo su cui poggia.

1. Dove vengono generate le chiavi crittografiche e dove risiedono fisicamente? Se l'operatore può accedervi (anche temporaneamente, anche sotto le spoglie del recupero), l'E2EE è nominale.
2. Esiste una verifica indipendente dell'interlocutore (numeri di sicurezza, codici QR, confronto fuori banda) che prevenga un attacco man-in-the-middle durante lo stabilimento della conversazione?
3. Il codice del client è verificabile — aperto, pubblicato, riproducibile — o richiede di fidarsi della parola del fornitore su ciò che il client fa realmente?
4. Quali metadati genera e conserva il servizio, e per quanto tempo? Anche se il contenuto è opaco, i metadati possono ricostruire buona parte delle informazioni sensibili.

Queste quattro domande non richiedono informazioni tecniche avanzate; richiedono informazioni a cui qualsiasi operatore onesto può rispondere nella sua documentazione pubblica. La qualità e la precisione della risposta dicono sul prodotto tanto quanto la risposta stessa.

La crittografia end-to-end, fatta bene, è una delle costruzioni più raffinate che la crittografia contemporanea abbia consegnato alla pratica quotidiana. L'idea originale — due persone possono concordare un segreto su un canale pubblico — appartiene a Whitfield Diffie e Martin Hellman, 1976; mezzo secolo dopo continuiamo a viverne le conseguenze. Ma, come accade con ogni promessa tecnica, il suo valore dipende dall'effettivo adempimento, non dall'etichetta. La domanda del professionista onesto non è «è crittografato?», ma «chi ha le chiavi?». Le risposte hanno conseguenze diverse. Conviene saperlo.

Fonti e letture aggiuntive

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, novembre 1976. Articolo fondante della crittografia a chiave pubblica.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, specifica pubblica di Open Whisper Systems, revisione 2016. Base del protocollo Signal e dei suoi derivati industriali.
- RFC 7748 — Elliptic Curves for Security (IETF, gennaio 2016). Specifica normativa delle curve X25519 e X448 utilizzate nei moderni scambi di chiavi.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Capitoli sullo scambio di chiavi e sui protocolli di crittografia autenticata.
- Regolamento (UE) 2024/1183 sul quadro europeo relativo a un'identità digitale (eIDAS 2) — stabilisce quadri in cui la verifica indipendente dell'interlocutore acquisisce supporto istituzionale, e dove la distinzione tra crittografia nominale e reale ha conseguenze giuridiche diverse.

← PrecedenteKill switch e cattura istituzionaleSuccessivo →Il modello di business come segnale di fiducia

Letture recenti

- Analisi · 18 maggio 2026 Privacy reale vs apparente: le domande da porsi
- Analisi · 18 maggio 2026 Self-hosting come pratica professionale
- Concetto · 18 maggio 2026 Le 24 parole: cos'è un'identità crittografica

Porta questo articolo dove ne hai bisogno.

↓ Markdown ↓ Testo semplice ↓ PDF

Il file viene scaricato sul tuo dispositivo. Da lì puoi salvarlo, importarlo in Solo2 o condividerlo come preferisci. Cuadernos non decide la destinazione per te.

Sigillo di cera · SHA-256 54f402f54c2052fe840b4ad4e14f698433dcec3604ac4fbe7a5517b83ca3a861

Cuadernos Lacre · Una pubblicazione di Menzuri Gestión S.L. ·
scritta da R.Eugenio · a cura del team di Solo2.