End-to-end-kryptering, förklarat på riktigt

Vad leverantörer säger när de säger E2EE, och vad de inte säger. En didaktisk förklaring av mekanismen och dess begränsningar, utan reklamförpackning.

Låt oss vara tydliga: WhatsApp säger att dina meddelanden är end-to-end-krypterade. Det är sant — och det räcker inte. Om säkerhetskopian går till iCloud eller Google Drive utan ytterligare kryptering bryts krypteringen i din egen telefon. Den operativa frågan är inte om det är krypterat, utan var nycklarna finns.

Vad kryptering egentligen betyder

Att kryptera ett meddelande innebär att omvandla det till något som ser ut som brus för alla som inte besitter en viss information som kallas nyckel. Operationen görs på sändarens enhet och med rätt nyckel återställs det på mottagarens enhet. Däremellan färdas meddelandet som en följd av bytes utan uppenbar innebörd. Det är den enkla idén. Resten av artikeln handlar om de nyanser som, beroende på fall, gör den till en verklig garanti eller en marknadsföringsetikett.

Adjektivet *end-to-end* — på svenska *totalsträckskryptering*, förkortat E2EE — lägger till en precision. Kryptering görs inte för att en mellanliggande server ska kunna läsa och leverera den. Det görs för att endast de två ändarna — sändarens enhet och mottagarens enhet — ska besitta nyckeln. Varje server som meddelandet passerar ser bruset, inte meddelandet. Det är den tekniska skillnaden mot kryptering *under transport*, där innehållet färdas krypterat från en server till nästa, men varje server den passerar dekrypterar det för att vidarebefordra det, vilket tillfälligt återställer texten i klartext.

Paradoxen om den delade hemligheten

Det finns ett uppenbart problem. För att två personer ska kunna kryptera och dekryptera meddelanden mellan sig behöver båda samma nyckel. Men hur kommer de överens om denna nyckel om allt de skickar till varandra, per definition, passerar genom en kanal där någon kan lyssna? Att komma överens om nyckeln i samma kanal som de senare ska använda den i verkar omöjligt: om angriparen hör den vid överenskommelsen kommer hen att kunna dekryptera allt efterföljande. Under decennier löste klassisk kryptografi detta på den hårda vägen: nycklarna lämnades över personligen, innan de började användas, vid fysiska möten. Ambassadörer bar väskor med nycklar fastsydda i fodret på sin kappa.

I modern e-post är den lösningen inte skalbar. Om vi var tvungna att fysiskt gå hem till varje person som vi hade för avsikt att kommunicera krypterat med skulle vi aldrig komma till skott att prata med någon. Frågan som ställdes för femtio år sedan av kryptografisamhället var denna: är det möjligt för två personer som inte känner varandra och som bara delar en offentlig kanal att komma överens om en hemlighet i samma offentliga kanal, som ingen som lyssnar på kanalen kan känna till?

Elegansen i Diffie-Hellman

År 1976 demonstrerade två matematiker vid namn Whitfield Diffie och Martin Hellman något till synes omöjligt: att två personer som bara pratar genom en offentlig kanal — en kanal där vem som helst kan höra allt de säger — kan komma överens om ett hemligt lösenord utan att någon lyssnare kan upptäcka det. Det låter som magi. Det är det inte: det är matematik. Diffie-Hellman-nyckelutbyte, som det har varit känt som sedan dess, är basen för praktiskt taget all krypterad kommunikation på internet, och ett halvsekel av intensiv användning och global akademisk granskning bekräftar dess soliditet. Den som vill se den visuella intuitionen eller matematiken kan läsa vidare. Den som föredrar att lita på att det fungerar kan också fortsätta utan att tappa tråden i artikeln.

För den som vill visualisera det finns en känd analogi med färger. Tänk dig att Alice och Bruno kommer överens offentligt om en grundfärg — låt oss säga gult — inför ögonen på Eva som lyssnar på dem. Var och en väljer privat en andra hemlig färg och blandar sin hemlighet med den gula. Alice får en viss orange; Bruno får en viss grön. De byter resultaten med varandra inför ögonen på Eva. Nu blandar var och en den mottagna färgen med sin egen hemlighet, och båda når fram till samma slutliga färg, eftersom ordningen på blandningarna inte spelar någon roll. Eva har sett det gula och de två mellanliggande blandningarna, men inte hemligheterna; utan någon av hemligheterna kan hon inte nå fram till slutfärgen. Den verkliga matematiken ersätter färgerna med exponentiering i modulära grupper eller elliptiska kurvor, men idén är densamma: den delade hemligheten byggs upp offentligt utan att någon i kanalen kan rekonstruera den.

I aritmetik, för de som föredrar att se mekanismen: Alice väljer ett hemligt tal a, Bruno väljer b. De utbyter g^a och g^b öppet över kanalen. Alice beräknar $(g^b)^a$ och Bruno beräknar $(g^a)^b$; båda når fram till samma g^{ab}. Eva ser g, g^a och g^b passera genom kanalen, men att återställa a från g^a — det så kallade diskreta logaritmproblemet — kräver en astronomisk beräkningstid som överstiger universums ålder när g väljs i en lämplig matematisk grupp.

För de som vill testa det med små siffror. Diffie-Hellman-utbytet kan gås igenom i sin helhet med siffror som är tillräckligt små för att räknas ut för hand. Den som föredrar att inte ge sig in på aritmetik kan hoppa över detta block utan att tappa tråden i artikeln; den som vill se

mekanismen fungera steg för steg hittar det här. **De offentliga reglerna**, som vem som helst kan läsa: ett primtal $p = 11$ (i verkliga Diffie-Hellman är det cirka trehundra siffror; vi använder elva för att beräkningarna ska rymmas på en sida), en bas $g = 2$, och konventionen att all aritmetik utförs *modulo p* — man beräknar, delar med p och behåller resten, som en klocka med elva positioner som återgår till noll när den passerar tio. **De privata valen**, ett vardera och aldrig delade: Alice väljer $a = 4$. Bruno väljer $b = 7$.

Steg 1. Alice beräknar $2^4 = 16$, sedan 16 mod 11 = **5**. Hon skickar femman. Eva antecknar den.

Steg 2. Bruno beräknar $2^7 = 128$, sedan 128 mod 11 = **7**. Han skickar sjuan. Eva antecknar även den. Efter de två överföringarna innehåller Evas anteckningsbok fyra uppgifter: $p = 11$, $g = 2$, A = 5, B = 7. Hon saknar det gemensamma talet som Alice och Bruno är på väg att härleda — och som Eva inte kommer att kunna rekonstruera.

Steg 3. Alice tar sjuan som Bruno skickade henne och upphöjer den till sin privata exponent $a = 4$. För att undvika att hantera $7^4 = 2401$ beräknas det i delar genom att modulo tillämpas i varje steg:

$7^2 = 49$

49 mod 11 = 5

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

25 mod 11 = 3

Alice får talet **3**.

Steg 4. Bruno tar femman som Alice skickade honom och upphöjer den till sin privata exponent $b = 7$. Återigen i delar:

$5^2 = 25$ 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

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

Bruno får också **3**.

Båda har kommit fram till samma tal, 3, genom att arbeta parallellt. Ingen av dem skickade sin privata exponent vid något tillfälle. Alice vet inte att $b = 7$; Bruno vet inte att $a = 4$. Var och en använde det offentliga värdet som den andra skickade kombinerat med sin egen privata exponent, och de möttes vid samma destination. **Varför når de samma tal?** Vad var och en beräknade: Alice, $(g^b)^a = 2^{7\times4} = 2^{28}$ mod 11. Bruno, $(g^a)^b = 2^{4\times7} = 2^{28}$ mod 11. Det är samma mängd eftersom ordningen för multiplikation av exponenterna inte spelar någon roll ($7 \times 4 = 4 \times 7$). Var och en kom via en annan väg till samma destination.

Och Eva? Hon har i sin anteckningsbok $p = 11$, $g = 2$, A = 5, B = 7, och hon skulle vilja ha 3. För att beräkna det skulle hon behöva veta a eller b — men ingendera har färdats genom kanalen. Hennes enda utväg är att fråga sig själv: «för vilken exponent a gäller 2^a mod 11 = 5?». Med ett så litet p kan hon prova 0, 1, 2, 3, 4... och hitta det på under en minut. Men om man ersätter 11 med ett primtal på trehundra siffror har rymden av möjliga exponenter fler element än det finns atomer i det observerbara universum. **Idag finns ingen algoritm känd av mänskligheten som kan gå igenom det utrymmet på mindre än miljarder år.** Detta är det så kallade *diskreta logaritmproblemet*: enkelt framåt, beräkningsmässigt omöjligt bakåt. Och det är anledningen till varför krypteringen står emot även om Eva har följt hela konversationen bokstav för bokstav.

Tre enkla ingredienser — klockaritmetik, exponentiering och kommutativiteten vid multiplikation ($a \cdot b = b \cdot a$) — i kombination producerar ett protokoll som halva mänskligheten är beroende av varje dag för sina privata kommunikationer. Ingen av de tre delarna, var för sig, verkar speciell. Det avgörande är sammansättningen.

Från Diffie-Hellman till Signal-protokollet

End-to-end-kryptering som används av dagens professionella meddelandeappar vilar, nästan utan undantag, på en elegant och härdad version av Diffie-Hellman-utbytet. Signal-protokollet, designat av Trevor Perrin och Moxie Marlinspike mellan 2013 och 2016, är referensen. Det kombinerar två nyckelidéer. Den första är nyckelutbyte i elliptiska kurvor (X25519), som producerar den ursprungliga delade hemligheten mellan två enheter. Den andra är det så kallade Double Ratchet — dubbelt spärrskaft —, som förnyar nycklarna automatiskt med varje meddelande, så att kompromettering av enheten idag inte tillåter dekryptering av tidigare meddelanden, och inte heller framtida meddelanden när spärrskaftet har roterats.

I Zig tar X25519-utbytet som producerar den delade hemligheten mellan två enheter sex rader i anspråk, med användning av standardbiblioteket:

```
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)
```

Vad som händer på de sex raderna: De publika nycklarna färdas öppet. De privata nycklarna lämnar aldrig respektive enhet. Varje part härleder, utifrån sin privata och den andras publika, samma hemlighet på trettiotvå bytes som ingen i kanalen kan återställa. Den hemligheten fungerar senare som frö för att kryptera de utbytta meddelandena. Signal-protokollets Double Ratchet lägger till en konstant rotation av det materialet så att kompromettering av ett ögonblick inte komprometterar resten av samtalet.

Och vad finns det exakt inuti `std.crypto.dh.X25519`? Ingen dold magi. Det är två korta funktioner som kan läsas i sin helhet i Zigs eget standardbibliotek. Den första härleder den publika nyckeln från den privata — utbytets «g^a»:

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

På artikelns språk: den privata nyckeln «multipliceras» — i elliptisk mening, inte elementär aritmetisk — med baspunkten för Curve25519-kurvan, och resultatet serialiseras till trettiotvå bytes. Operationen `clampedMul` är den härdade versionen av den skalära multiplikationen: den inkorporerar de skyddsåtgärder som kryptografisamhället har lagt till under årens lopp för att stå emot kända familjer av attacker. Två rader funktionskropp.

Den andra funktionen kombinerar din privata nyckel med den publika nyckeln som den andra parten skickar till dig. Det är utbytets «$(g^b)^a$», som producerar den trettiotvå bytes stora delade hemligheten som ingen av er någonsin sände:

```
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();
}
```

Ytterligare två rader. Den mottagna publika nyckeln tolkas som en punkt på kurvan, och «multipliceras» med ens egen privata nyckel. Genom kommutativiteten hos kurvoperationen — analogt med kommutativiteten vid multiplikationen av exponenter som vi såg i det numeriska exemplet — slutar båda parter med samma serialiserade punkt: exakt den delade hemlighet som artikeln talar om.

Det är allt. Det som i en applikation ser ut som magi är i verkligheten två funktioner på tre rader vardera. Den tekniska komplexiteten är koncentrerad till en enda operation, `clampedMul`, som är skriven längre ner i samma standardbibliotek, granskad under årtionden av det internationella kryptografiska samhället, och tillgänglig för vem som helst som vill läsa den bokstav för bokstav. Det finns ingen svart låda, varken i vår applikation eller i Zigs standardbibliotek. Det finns öppen källkod som en människa kan förstå, och man kan själv välja i vilken takt man vill sätta sig in i den.

Vad end-to-end-kryptering skyddar

Vad E2EE skyddar väl, förutsatt en korrekt implementering, är meddelandets innehåll under transport. En mellanliggande server som tar emot och vidarebefordrar de krypterade data kommer att se en följd av obegripliga bytes. En angripare med åtkomst till kabeln, routern, wifi-åtkomstpunkten kommer att se detsamma. En tjänsteleverantör som sparar kopior av trafiken kommer inte att kunna läsa den i efterhand. En regering som beordrar tjänsteoperatören att lämna ut innehållet kommer att få samma obegripliga bytes som servern hade från början.

Detta är i praktiska termer mycket. Det är skillnaden mellan att skriva ett brev inuti ett ogenomskinligt kuvert och att skriva det på ett vykort. Båda kommer fram. Endast ett bevarar innehållet inför brevbäraren.

Vad end-to-end-kryptering inte skyddar

Det är värt att veta det lika väl. E2EE skyddar inte metadata: servern vet fortfarande att användare A skickar data till användare B, vid vilken tidpunkt, med vilken frekvens och varifrån, även om den inte vet vad som sägs. Dessa metadata, som vi redan har argumenterat för i *Att kryptera är inte att vara privat*, är ofta mer avslöjande än innehållet. Att veta att någon ringde en advokatbyrå specialiserad på skilsmässor en fredag kl. 22:00 i trettio minuter berättar en historia som innehållet i samtalet aldrig berättade. Det är samma situation som att se en person gå in och ut flera gånger från en onkologklinik: man behöver inte höra något av det som sägs där inne för att föreställa sig vad som händer. En enskild isolerad metadatapunkt behöver inte betyda någonting; flera korsrefererade ritar upp något som är alltför likt sanningen. E2EE skyddar inte ändpunkterna: om mottagarens enhet är komprometterad av ett skadligt program dekrypteras meddelandet normalt för den mottagaren och det skadliga programmet läser det. E2EE skyddar inte mot samtalspartnerns identitet i sig: om Alice tror att hon pratar med Bruno men en angripare har skjutit in sig i början (en *man in the middle*) och protokollet inte inkluderar oberoende verifiering, slutar de två parterna med att prata med inkräktaren i tron att de pratar med varandra.

Det finns en fjärde sak som är värd att formulera utan tvetydighet. E2EE hindrar inte en leverantör som påstår sig erbjuda det från att dessutom spara en kopia av det okrypterade meddelandet i sina egna system. Påståendet ”mina meddelanden är end-to-end-krypterade” och påståendet ”leverantören sparar inte mitt innehåll” är inte samma sak. En app kan uppfylla det första medan den bryter mot det andra; vi har sett det i tidningsrubriker upprepade gånger sedan 2018. Användaren har, såvida inte klientens kod är verifierbar, inget tekniskt sätt att skilja det ena fallet från det andra utan expertundersökning. Det mest kända fallet hos den breda allmänheten: WhatsApp krypterar meddelanden end-to-end under transport, men om användaren aktiverar säkerhetskopiering i iCloud eller Google Drive utan ytterligare kryptering sparas den kopian läsbart i en tredje parts infrastruktur, och krypteringen bryts i användarens egen ände.

Frågan operatören inte vill höra

En app som påstår sig kryptera end-to-end kan tekniskt sett göra en av tre saker beträffande nycklarna:

1. **Nycklarna finns endast på enheterna.** De genereras och finns uteslutande på användarnas enheter; operatören känner inte till dem och lagrar dem inte. Det är det optimala fallet.
2. **Operatören kan få åtkomst om hen vill.** Operatören har användarnas nycklar (eller kan generera dem efter behag) och sparar dem i sina databaser. Om hen vill, eller tvingas till det, kan hen läsa innehållet. Detta är fallet för de flesta ”molntjänster”.
3. **Operatören kan inte få åtkomst genom design, men kontrollerar åtkomsten.** Operatören har inte nycklarna, men har kontroll över appen som genererar dem. Om hen tvingas till det kan hen skicka en skadlig uppdatering som fångar upp nycklarna eller innehållet före kryptering. Detta är fallet för många kommersiella E2EE-tjänster.

Den operativa frågan är därför inte om något är krypterat, utan vem som har kontroll över enheten och programvaran som hanterar nycklarna. I Solo2 finns nycklarna enbart i ditt Valv (IndexedDB krypterad med ditt lösenord) och programvaran är verifierbar öppen källkod.

För den professionelle läsaren

End-to-end-kryptering är ett verktyg för digital suveränitet. Men som alla verktyg beror dess effektivitet på handen som håller i det och marken det vilar på.

1. Var genereras de kryptografiska nycklarna och var finns de fysiskt? Om operatören kan komma åt dem (även tillfälligt, även under sken av återställning) är E2EE endast nominellt.
2. Finns det oberoende verifiering av samtalspartnern (säkerhetsnummer, QR-koder, out-of-band-jämförelse) som förhindrar en man-in-the-middle-attack när samtalet etableras?
3. Kan klientens kod granskas — är den öppen, publicerad, reproducerbar — eller kräver det att man litar på leverantörens ord om vad klienten faktiskt gör?
4. Vilka metadata genererar och sparar tjänsten, och hur länge? Även om innehållet är ogenomskinligt kan metadata rekonstruera en stor del av den känsliga informationen.

Dessa fyra frågor ber inte om avancerad teknisk information; de ber om information som varje ärlig operatör kan svara på i sin offentliga dokumentation. Kvaliteten och precisionen i svaret säger lika mycket om produkten som svaret självt.

End-to-end-kryptering, rätt utfört, är en av de finaste konstruktionerna som modern kryptografi har levererat till daglig praxis. Den ursprungliga idén — att två personer kan enas om en hemlighet via en offentlig kanal — tillhör Whitfield Diffie och Martin Hellman, 1976; ett halvt sekel senare lever vi fortfarande i dess konsekvens. Men som med alla tekniska löften beror dess värde på faktiskt uppfyllande, inte på etiketten. Den ärlige fackmannens fråga är inte ”är det krypterat?”, utan ”vem har nycklarna?”. Svaren har olika konsekvenser. Det är värt att känna till dem.

Källor och vidare läsning

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, november 1976. Grundläggande artikel om kryptografi med öppen nyckel.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, offentlig specifikation från Open Whisper Systems, revision 2016. Grunden för Signal-protokollet och dess industriella derivat.
- RFC 7748 — Elliptic Curves for Security (IETF, januari 2016). Normativ specifikation av kurvorna X25519 och X448 som används i moderna nyckelutbyten.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Kapitel om nyckelutbyte och autentiserade krypteringsprotokoll.
- Förordning (EU) 2024/1183 om en ram för europeisk digital identitet (eIDAS 2) — upprättar ramverk där oberoende verifiering av samtalspartnern får institutionellt stöd, och där åtskillnaden mellan nominell och verklig kryptering har olika juridiska konsekvenser.

← FöregåendeKill switch och institutionell fångstNästa →Affärsmodellen som en signal om förtroende

Senaste läsning

- Analys · 18 maj 2026 Verklig vs skenbar integritet: Frågorna man bör ställa sig
- Analys · 18 maj 2026 Self-hosting som yrkespraxis
- Koncept · 18 maj 2026 De 24 orden: vad en kryptografisk identitet är

Ta med dig den här artikeln dit du behöver den.

↓ Markdown ↓ Klartext ↓ PDF

Filen laddas ner till din enhet. Därifrån kan du spara den, importera den till Solo2 eller dela den var du vill. Cuadernos bestämmer inte destinationen åt dig.

Lacksigill · SHA-256 44f3b48d82b7a8c0063dee4022265e7a5861c6f293e08d0326cb497d43151fc6

Cuadernos Lacre · En utgåva från Menzuri Gestión S.L. ·
skriven av R.Eugenio · redigerad av teamet bakom Solo2.