Ende-til-ende-kryptering, forklaret for alvor

Hvad udbydere siger, når de siger E2EE, og hvad de ikke siger. En pædagogisk forklaring af mekanismen og dens begrænsninger, uden reklameindpakning.

For at vi forstår hinanden: WhatsApp siger, at dine beskeder er ende-til-ende-krypterede. Det er sandt — og det er ikke nok. Hvis din backup går til iCloud eller Google Drive uden yderligere kryptering, brydes krypteringen på din egen telefon. Det operationelle spørgsmål er ikke, om det er krypteret, men hvor nøglerne befinder sig.

Hvad kryptering egentlig betyder

At kryptere en besked betyder at transformere den til noget, der ligner støj for enhver, der ikke besidder en bestemt information kaldet en nøgle. Operationen udføres på afsenderens enhed og føres tilbage på modtagerens enhed med den korrekte nøgle. Imellem de to rejser beskeden som en række bytes uden tilsyneladende betydning. Det er den enkle idé. Resten af artiklen beskæftiger sig med de nuancer, der alt efter tilfældet gør den til en reel garanti eller et marketingmærkat.

Adjektivet *ende-til-ende* — på engelsk *end-to-end*, forkortet E2EE — tilføjer en præcision. Kryptering udføres ikke for at en mellemliggende server kan læse og levere den. Det gøres for at kun de to ender — afsenderens enhed og modtagerens enhed — besidder nøglen. Enhver server, som beskeden passerer igennem, ser støjen, ikke beskeden. Det er den tekniske forskel fra kryptering *under transit*, hvor indholdet krypteres fra den ene server til den næste, men hver server, det passerer, dekrypterer det for at videresende det, hvilket midlertidigt genopretter teksten i klar tekst.

Paradokset om den delte hemmelighed

Der er et indlysende problem. For at to personer kan kryptere og dekryptere beskeder indbyrdes, har de begge brug for den samme nøgle. Men hvordan bliver de enige om denne nøgle, hvis alt, hvad de sender til hinanden, per definition passerer gennem en kanal, hvor nogen kan lytte med? At aftale nøglen i den samme kanal, som de senere skal bruge den i, virker umuligt: hvis angriberen hører den ved aftalen, vil vedkommende kunne dekryptere alt efterfølgende. I årtier løste klassisk kryptografi dette på den hårde måde: nøglerne blev overleveret personligt, før de blev taget i brug, ved fysiske møder. Ambassadører bar kufferter med nøgler syet ind i foret på deres frakker.

I moderne e-mail kan denne løsning ikke skaleres. Hvis vi skulle gå fysisk hjem til hver person, som vi havde til hensigt at kommunikere krypteret med, ville vi aldrig komme til at tale med nogen. Spørgsmålet, som kryptografmiljøet stillede for halvtreds år siden, var dette: er det muligt for to personer, der ikke kender hinanden, og som kun deler en offentlig kanal, at aftale en hemmelighed i den samme offentlige kanal, som ingen, der lytter til kanalen, kan kende?

Elegancen i Diffie-Hellman

I 1976 demonstrerede to matematikere ved navn Whitfield Diffie og Martin Hellman noget tilsyneladende umuligt: at to personer, der kun taler via en offentlig kanal — en kanal, hvor alle kan høre alt, hvad de siger — kan blive enige om en hemmelig adgangskode, uden at nogen lytter kan opdage den. Det lyder som magi. Det er det ikke: det er matematik. Diffie-Hellman-nøgleudveksling, som det har været kendt siden da, er grundlaget for praktisk talt al krypteret kommunikation på internettet, og et halvt århundredes intensiv brug og global akademisk granskning bekræfter dets soliditet. Enhver, der ønsker at se den visuelle intuition eller matematikken, kan læse videre. Enhver, der foretrækker at stole på, at det virker, kan også fortsætte uden at miste tråden i artiklen.

For dem, der vil visualisere det, er der en velkendt analogi med farver. Forestil dig, at Alice og Bruno bliver enige om en grundfarve offentligt — lad os sige gul — for øjnene af Eva, der lytter til dem. Hver vælger en anden hemmelig farve privat og blander sin hemmelighed med den gule. Alice får en bestemt orange; Bruno får en bestemt grøn. De udveksler resultaterne for øjnene af Eva. Nu blander hver den modtagne farve med sin egen hemmelighed, og begge når frem til den samme slutfarve, fordi blandingsrækkefølgen ikke betyder noget. Eva har set den gule og de to mellemblandinger, men ikke hemmelighederne; uden en af hemmelighederne kan hun ikke nå frem til slutfarve. Den virkelige matematik erstatter farverne med eksponentieringer i modulære grupper eller elliptiske kurver, men idéen er den samme: den delte hemmelighed opbygges offentligt, uden at nogen i kanalen kan rekonstruere den.

I aritmetik, for dem der foretrækker at se mekanismen: Alice vælger et hemmeligt tal a, Bruno vælger b. De udveksler g^a og g^b offentligt over kanalen. Alice beregner $(g^b)^a$ og Bruno beregner $(g^a)^b$; begge når frem til det samme g^{ab}. Eva ser g, g^a og g^b passere gennem kanalen, men at genvinde a fra g^a — det såkaldte diskrete logaritme-problem — kræver en astronomisk beregningstid, der overstiger universets alder, når g vælges i en passende matematisk gruppe.

For dem, der vil bekræfte det med små tal. Diffie-Hellman-udvekslingen kan gennemgås i sin helhed med tal, der er små nok til, at man kan regne det ud i hånden. De, der foretrækker at undgå aritmetik, kan springe denne blok over uden at miste tråden i artiklen; de, der ønsker at se

mekanismen fungere trin for trin, finder det her. **De offentlige regler**, som alle kan læse: et primtal $p = 11$ (i den virkelige Diffie-Hellman er det på cirka tre hundrede cifre; vi bruger elleve, så beregningerne kan være på én side), en base $g = 2$, og konventionen om, at al aritmetik udføres *modulo p* — der beregnes, divideres med p, og resten bevares, som et ur med elleve positioner, der vender tilbage til nul, når det passerer ti. **De private valg**, et hver, der aldrig deles: Alice vælger $a = 4$. Bruno vælger $b = 7$.

Trin 1. Alice beregner $2^4 = 16$, derefter 16 mod 11 = **5**. Hun sender femtallet. Eva noterer det.

Trin 2. Bruno beregner $2^7 = 128$, derefter 128 mod 11 = **7**. Han sender syvtallet. Eva noterer det også. Efter de to afsendelser indeholder Evas notesbog fire data: $p = 11$, $g = 2$, A = 5, B = 7. Hun mangler det fælles tal, som Alice og Bruno er ved at udlede — og som Eva ikke vil kunne rekonstruere.

Trin 3. Alice tager syvtallet, som Bruno sendte hende, og opløfter det til sin private eksponent $a = 4$. For at undgå at håndtere $7^4 = 2401$, beregnes det i dele ved at anvende modulo for hvert trin:

$7^2 = 49$

49 mod 11 = 5

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

25 mod 11 = 3

Alice får tallet **3**.

Trin 4. Bruno tager femtallet, som Alice sendte ham, og opløfter det til sin private eksponent $b = 7$. Igen i dele:

$5^2 = 25$ 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

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

Bruno får også **3**.

Begge er nået frem til det samme tal, 3, ved at arbejde parallelt. Ingen af dem sendte sin private eksponent på noget tidspunkt. Alice ved ikke, at $b = 7$; Bruno ved ikke, at $a = 4$. Hver især brugte den offentlige værdi, som den anden sendte, kombineret med sin egen private eksponent, og de mødtes ved den samme destination. **Hvorfor når de det samme tal?** Det, hver især beregnede: Alice, $(g^b)^a = 2^{7\times4} = 2^{28}$ mod 11. Bruno, $(g^a)^b = 2^{4\times7} = 2^{28}$ mod 11. Det er det samme beløb, fordi rækkefølgen for multiplikation af eksponenterne ikke betyder noget ($7 \times 4 = 4 \times 7$). Hver især nåede frem ad en forskellig vej til den samme destination.

Hvad med Eva? Hun har i sin notesbog $p = 11$, $g = 2$, A = 5, B = 7, og vil gerne have 3. For at beregne det skal hun kende a eller b — men ingen af dem har rejst over kanalen. Hendes eneste udvej er at spørge sig selv: «for hvilken eksponent a gælder 2^a mod 11 = 5?». Med et så lille p kan hun prøve 0, 1, 2, 3, 4... og finde det på under et minut. Men når man erstatter 11 med et primtal på tre hundrede cifre, har rummet af mulige eksponenter flere elementer, end der er atomer i det observerbare univers. **Der findes i dag ingen algoritme kendt af menneskeheden, der kan gennemgå det rum på mindre end milliarder af år.** Det er det såkaldte *diskrete logaritme-problem*: let fremad, beregningsmæssigt umuligt bagud. Og det er grunden til, at krypteringen modstår, selv hvis Eva har fulgt hele samtalen bogstav for bogstav.

Tre enkle ingredienser — ur-aritmetik, eksponentiering og den kommutative lov for multiplikation ($a \cdot b = b \cdot a$) — kombineret producerer en protokol, som den halve menneskehed er afhængig af hver dag til deres private kommunikation. Ingen af de tre dele virker i sig selv specielle. Det afgørende er sammensætningen.

Fra Diffie-Hellman til Signal-protokollen

Ende-til-ende-kryptering, som professionelle beskedapps bruger i dag, hviler næsten uden undtagelse på en elegant og hærdet version af Diffie-Hellman-udvekslingen. Signal-protokollen, designet af Trevor Perrin og Moxie Marlinspike mellem 2013 og 2016, er referencen. Den kombinerer to nøgleidéer. Den første er nøgleudveksling i elliptiske kurver (X25519), som skaber den oprindelige delte hemmelighed mellem to enheder. Den anden er den såkaldte Double Ratchet — dobbelt skralde — som fornyer nøglerne automatisk med hver besked, så kompromittering af enheden i dag ikke tillader dekryptering af tidligere beskeder, eller fremtidige beskeder, når skralden er blevet drejet.

I Zig fylder X25519-udvekslingen, der skaber den delte hemmelighed mellem to enheder, seks linjer ved hjælp af 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)
```

Hvad der sker på de seks linjer: De offentlige nøgler rejser åbent. De private nøgler forlader aldrig den respektive enhed. Hver part udleder, ud fra sin private og den andens offentlige, den samme hemmelighed på toogtredive bytes, som ingen i kanalen kan genvinde. Denne hemmelighed fungerer senere som frø til at kryptere de udvekslede beskeder. Signal-protokollens Double Ratchet tilføjer en konstant rotation af dette materiale, så kompromittering af et øjeblik ikke kompromitterer resten af samtalen.

Og hvad er der præcist inde i `std.crypto.dh.X25519`? Ingen skjult magi. Det er to korte funktioner, der kan læses i deres helhed i selve Zigs standardbibliotek. Den første udleder den offentlige nøgle fra den private — udvekslingens «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();
}
```

I artiklens sprog: Den private nøgle «ganges» — i elliptisk, ikke elementær aritmetisk forstand — med basispunktet for Curve25519-kurven, og resultatet serialiseres til toogtredive bytes. Operationen `clampedMul` er den hærdede version af denne skalære multiplikation: Den inkorporerer de sikkerhedsforanstaltninger, som det kryptografiske samfund har tilføjet gennem årene for at modstå kendte familier af angreb. To linjer funktionskrop.

Den anden funktion kombinerer din private nøgle med den offentlige nøgle, som den anden part sender dig. Det er udvekslingens «$(g^b)^a$», der producerer den delte hemmelighed på toogtredive bytes, som ingen af jer nogensinde overførte:

```
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();
}
```

To linjer mere. Den modtagne offentlige nøgle tolkes som et punkt på kurven og «ganges» med ens egen private nøgle. På grund af den kommutative egenskab ved kurveoperationen — analog til kommutativiteten af multiplikationen af eksponenter, som vi så i det numeriske eksempel — ender begge parter med det samme serialiserede punkt: nøjagtig den delte hemmelighed, som artiklen taler om.

Det er det hele. Det, der i en applikation virker som magi, er i virkeligheden to funktioner på tre linjer hver. Den tekniske kompleksitet er koncentreret i en enkelt operation, `clampedMul`, som er skrevet længere fremme i det samme standardbibliotek, gennemgået i årtier af det internationale kryptografiske samfund, og tilgængelig for enhver, der ønsker at læse den bogstav for bogstav. Der er ingen sort boks, hverken i vores applikation eller i Zigs standardbibliotek. Der er open source-kode, som et menneske kan forstå, og vælge det tempo, de ønsker at gå ind i det med.

Hvad ende-til-ende-kryptering beskytter

Hvad E2EE beskytter godt, forudsat en korrekt implementering, er indholdet af beskeden under transit. En mellemliggende server, der modtager og videresender de krypterede data, vil se en række uforståelige bytes. En angriber med adgang til kablet, routeren, wifi-adgangspunktet vil se det samme. En tjenesteudbyder, der opbevarer kopier af trafikken, vil ikke kunne læse den efterfølgende. En regering, der beordrer tjenesteoperatøren til at udlevere indholdet, vil modtage de samme uforståelige bytes, som serveren havde i første omgang.

Dette er, i praktiske termer, meget. Det er forskellen på at skrive et brev i en uigennemsigtig konvolut og at skrive det på et postkort. Begge ankommer. Kun den ene bevarer indholdet over for postbuddet.

Hvad ende-til-ende-kryptering ikke beskytter

Det er værd at vide det lige så godt. E2EE beskytter ikke metadata: serveren ved stadig, at bruger A sender data til bruger B, på hvilket tidspunkt, med hvilken frekvens og hvorfra, selvom den ikke ved, hvad der bliver sagt. Disse metadata har vi allerede argumenteret for i *Kryptering er ikke at være privat*, er ofte mere afslørende end indholdet. At vide, at nogen ringede til et advokatfirma specialiseret i skilsmisser en fredag kl. 22:00 i tredive minutter, fortæller en historie, som indholdet af opkaldet aldrig fortalte. Det er den samme situation som at se en person gå ind og ud af en onkologisk klinik flere gange: man behøver ikke at høre noget af det, der bliver talt om indenfor, for at forestille sig, hvad der sker. Et enkelt metadata-punkt alene betyder måske ingenting; flere krydsrefererede tegner noget, der minder alt for meget om sandheden. E2EE beskytter ikke endepunkterne: hvis modtagerens enhed er kompromitteret af et ondsindet program, dekrypteres beskeden normalt for den modtager, og det ondsindede program læser den. E2EE beskytter ikke mod samtalepartnerens identitet i sig selv: hvis Alice tror, hun taler med Bruno, men en angriber har indskudt sig i starten (en *man in the middle*), og protokollen ikke inkluderer uafhængig verificering, ender de to parter med at tale med den indtrængende i den tro, at de taler med hinanden.

Der er en fjerde ting, der bør formuleres uden tvetydighed. E2EE forhindrer ikke en udbyder, der påstår at tilbyde det, i også at gemme en kopi af den ukrypterede besked i sine egne systemer. Påstanden "mine beskeder er ende-til-ende-krypterede" og påstanden "udbyderen gemmer ikke mit indhold" er ikke den samme. En app kan opfylde den første, mens den overtræder den anden; vi har set det i avisoverskrifter gentagne gange siden 2018. Brugeren har, medmindre klientens kode er verificerbar, ingen teknisk måde at skelne det ene tilfælde fra det andet uden ekspertundersøgelse. Det mest kendte tilfælde i den brede offentlighed: WhatsApp krypterer beskeder ende-til-ende under transit, men hvis brugeren aktiverer backup i iCloud eller Google Drive uden yderligere kryptering, gemmes den kopi læsbart i en tredjeparts infrastruktur, og krypteringen brydes i brugerens egen ende.

Spørgsmålet, operatøren ikke ønsker at høre

En app, der påstår at kryptere ende-til-ende, kan teknisk set gøre én af tre ting med hensyn til nøglerne:

1. **Nøglerne findes kun på enhederne.** De genereres og findes udelukkende på brugernes enheder; operatøren kender dem ikke og gemmer dem ikke. Det er det optimale tilfælde.
2. **Operatøren kan få adgang, hvis de vil.** Operatøren har brugernes nøgler (eller kan generere dem efter behag) og gemmer dem i sine databaser. Hvis de vil, eller tvinges til det, kan de læse indholdet. Dette er tilfældet for de fleste ”cloud”-tjenester.
3. **Operatøren kan ikke få adgang ved design, men kontrollerer adgangen.** Operatøren har ikke nøglerne, men har kontrol over den app, der genererer dem. Hvis de tvinges til det, kan de sende en ondsindet opdatering, der opsnapper nøglerne eller indholdet før kryptering. Dette er tilfældet for mange kommercielle E2EE-tjenester.

Det operationelle spørgsmål er derfor ikke, om noget er krypteret, men hvem der har kontrollen over enheden og den software, der administrerer nøglerne. I Solo2 findes nøglerne udelukkende i din Boks (IndexedDB krypteret med din adgangskode), og softwaren er verificerbar open source.

Til den professionelle læser

Ende-til-ende-kryptering er et værktøj til digital suverænitet. Men som ethvert værktøj afhænger dets effektivitet af den hånd, der fører det, og den grund, det hviler på.

1. Hvor genereres de kryptografiske nøgler, og hvor er de fysisk placeret? Hvis operatøren kan få adgang til dem (selv midlertidigt, selv under dække af gendannelse), er E2EE kun nominelt.
2. Er der uafhængig verificering af samtalepartneren (sikkerhedsnumre, QR-koder, out-of-band-sammenligning), der forhindrer et man-in-the-middle-angreb under etableringen af samtalen?
3. Kan klientens kode auditeres — er den åben, offentliggjort, reproducerbar — eller kræver den, at vi stoler på udbyderens ord om, hvad klienten rent faktisk gør?
4. Hvilke metadata genererer og gemmer tjenesten, og hvor længe? Selvom indholdet er uigennemskueligt, kan metadata rekonstruere en stor del af de følsomme oplysninger.

Disse fire spørgsmål beder ikke om avanceret teknisk information; de beder om information, som enhver ærlig operatør kan besvare i sin offentlige dokumentation. Kvaliteten og præcisionen af svaret siger lige så meget om produktet som selve svaret.

Ende-til-ende-kryptering, når det er gjort rigtigt, er en af de fineste konstruktioner, som moderne kryptografi har leveret til daglig praksis. Den oprindelige idé — at to personer kan blive enige om en hemmelighed via en offentlig kanal — tilhører Whitfield Diffie og Martin Hellman, 1976; et halvt århundrede senere lever vi stadig i konsekvensen af den. Men som med ethvert teknisk løfte afhænger dets værdi af reel overholdelse, ikke af mærkaten. Den ærlige fagpersons spørgsmål er ikke "er det krypteret?", men "hvem har nøglerne?". Svarene har forskellige konsekvenser. Det er værd at kende dem.

Kilder og yderligere læsning

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, november 1976. Grundlæggende artikel om public key-kryptografi.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, offentlig specifikation fra Open Whisper Systems, revision 2016. Grundlaget for Signal-protokollen og dens industrielle derivater.
- RFC 7748 — Elliptic Curves for Security (IETF, januar 2016). Normativ specifikation af X25519- og X448-kurverne, der bruges i moderne nøgleudvekslinger.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Kapitler om nøgleudveksling og godkendte krypteringsprotokoller.
- Forordning (EU) 2024/1183 om en ramme for en europæisk digital identitet (eIDAS 2) — etablerer rammer, hvor uafhængig verifikation af samtalepartneren får institutionel støtte, og hvor skelnen mellem nominel og reel kryptering har forskellige juridiske konsekvenser.

← ForrigeKill switch og institutionel indfangningNæste →Forretningsmodellen som et signal om tillid

Seneste læsning

- Analyse · 18. maj 2026 Reelt vs. tilsyneladende privatliv: De spørgsmål, man bør stille sig selv
- Analyse · 18. maj 2026 Self-hosting som professionel praksis
- Koncept · 18. maj 2026 De 24 ord: hvad en kryptografisk identitet er

Tag denne artikel med dig, hvor du har brug for den.

↓ Markdown ↓ Almindelig tekst ↓ PDF

Filen downloades til din enhed. Derfra kan du gemme den, importere den til Solo2 eller dele den, hvor du vil. Cuadernos beslutter ikke destinationen for dig.

Laksegl · SHA-256 11c4837208bab3f8105629e30e158eb4244b76f5ed18bb07996d23d240c8d254

Cuadernos Lacre · En udgivelse fra Menzuri Gestión S.L. ·
skrevet af R.Eugenio · redigeret af holdet bag Solo2.