End-to-end-versleuteling, echt uitgelegd

Wat providers zeggen als ze E2EE zeggen, en wat ze verzwijgen. Een didactische uitleg van het mechanisme en zijn grenzen, zonder reclameverpakking.

Laten we duidelijk zijn: WhatsApp zegt dat je berichten end-to-end versleuteld zijn. Dat is waar — en het is niet genoeg. Als de back-up zonder aanvullende versleuteling naar iCloud of Google Drive gaat, wordt de versleuteling op je eigen telefoon verbroken. De operationele vraag is niet of het versleuteld is, maar waar de sleutels zich bevinden.

Wat versleutelen echt betekent

Een bericht versleutelen betekent het veranderen in iets dat op ruis lijkt voor iedereen die niet over bepaalde informatie beschikt, een zogenaamde sleutel. De operatie wordt uitgevoerd op het apparaat van de verzender en wordt, met de juiste sleutel, ongedaan gemaakt op het apparaat van de ontvanger. Daartussen reist het bericht als een opeenvolging van bytes zonder duidelijke betekenis. Dat is het eenvoudige idee. De rest van het artikel gaat over de nuances die het, afhankelijk van het geval, veranderen in een echte garantie of in een marketinglabel.

Het bijvoeglijk naamwoord *end-to-end* — in het Engels *end-to-end*, afgekort E2EE — voegt een precisie toe. Versleuteling wordt niet gedaan zodat een tussenliggende server het kan lezen en afleveren. Het wordt gedaan zodat alleen de twee uiteinden — het apparaat van de verzender en het apparaat van de ontvanger — de sleutel bezitten. Elke server waar het bericht doorheen gaat, ziet de ruis, niet het bericht. Dat is het technische verschil met versleuteling *tijdens transport*, waarbij de inhoud versleuteld van de ene server naar de volgende reist, maar elke server waar het doorheen gaat het ontsleutelt om het door te sturen, waardoor de klare tekst tijdelijk wordt hersteld.

De paradox van het gedeelde geheim

Er is een voor de hand liggend probleem. Om twee mensen in staat te stellen berichten tussen elkaar te versleutelen und te ontsleutelen, hebben beiden dezelfde sleutel nodig. Maar hoe worden ze het eens over die sleutel als alles wat ze naar elkaar sturen per definitie via een kanaal gaat waar iemand zou kunnen meeluisteren? Over de sleutel overeenstemming bereiken in hetzelfde kanaal waar ze die later zullen gebruiken, lijkt onmogelijk: als de aancaller hem hoort bij het overeenkomen ervan, zal hij alles wat daarna komt kunnen ontsleutelen. Decennialang loste de klassieke cryptografie dit op de harde manier op: sleutels werden persoonlijk overhandigd, voordat ze in gebruik werden genomen, tijdens fysieke ontmoetingen. Ambassadeurs droegen koffers met sleutels die in de voering van hun jas waren genaaid.

In de hedendaagse e-mail is die oplossing niet schaalbaar. Als we fysiek naar het huis zouden moeten gaan van iedereen met wie we versleuteld willen communiceren, zouden we met niemand aan de praat komen. De vraag die vijftig jaar geleden door de cryptografische gemeenschap werd gesteld, was deze: is het mogelijk dat twee mensen die elkaar niet kennen en die alleen een openbaar kanaal delen, in datzelfde openbare kanaal een geheim afspreken dat niemand die het kanaal afluistert kan weten?

De elegantie van Diffie-Hellman

In 1976 toonden twee wiskundigen, genaamd Whitfield Diffie en Martin Hellman, iets schijnbaar onmogelijks aan: dat twee mensen, die alleen via een openbaar kanaal spreken — een kanaal waar iedereen alles kan horen wat ze zeggen —, een geheim wachtwoord kunnen afspreken zonder dat een luisteraar dit kan ontdekken. Het klinkt als magie. Dat is het niet: het is wiskunde. De Diffie-Hellman-sleuteluitwisseling, zoals die sindsdien bekend staat, is de basis van vrijwel alle versleutelde communicatie op internet, en een halve eeuw van intensief gebruik en wereldwijd academisch toezicht bevestigen de soliditeit ervan. Wie de visuele intuïtie of de wiskunde wil zien, kan verder lezen. Wie liever vertrouwt op het feit dat het werkt, kan ook doorgaan zonder de draad van het artikel te verliezen.

Voor wie het zich in een beeld wil voorstellen, is er een bekende analogie met kleuren. Stel je voor dat Alice en Bruno in het openbaar een basiskleur afspreken — zeg geel — in het bijzijn van Eva, die hen afluistert. Ieder kiest privé een tweede geheime kleur en mengt zijn geheim met het geel. Alice krijgt een bepaald oranje; Bruno krijgt een bepaald groen. Ze wisselen de resultaten uit in het bijzijn van Eva. Nu mengt ieder de ontvangen kleur met zijn eigen geheim, en beiden komen uit op dezelfde eindkleur, omdat de volgorde van de mengsels niet uitmaakt. Eva heeft het geel en de twee tussenmengsels gezien, maar niet de geheimen; zonder een van de geheimen kan ze de eindkleur niet bereiken. De werkelijke wiskunde vervangt de kleuren door machtsverheffen in modulaire groepen of elliptische curven, maar het idee is hetzelfde: het gedeelde geheim wordt in het openbaar opgebouwd zonder dat iemand in het kanaal het kan reconstrueren.

In de rekenkunde, voor wie liever het mechanisme ziet: Alice kiest een geheim getal a, Bruno kiest b. Ze wisselen g^a en g^b openlijk uit over het kanaal. Alice berekent $(g^b)^a$ en Bruno berekent $(g^a)^b$; beiden komen uit op dezelfde g^{ab}. Eva ziet g, g^a en g^b via het kanaal passeren, maar het herstellen van a uit g^a — het zogenaamde discrete logaritme-probleem — vereist een astronomische rekentijd die de leeftijd van het universum ver overstijgt wanneer g wordt gekozen in een geschikte wiskundige groep.

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, $(g^b)^a = 2^{7\times4} = 2^{28} \bmod 11$. Bruno, $(g^a)^b = 2^{4\times7} = 2^{28} \bmod 11$. 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.

Van Diffie-Hellman naar het Signal-protocol

De end-to-end-versleuteling die de huidige professionele messaging-apps gebruiken, rust bijna zonder uitzondering op een elegante en geharde versie van de Diffie-Hellman-uitwisseling. Het Signal-protocol, tussen 2013 en 2016 ontworpen door Trevor Perrin en Moxie Marlinspike, is de referentie. Het combineert twee kernideeën. De eerste is de sleuteluitwisseling in elliptische curven (X25519), die het initiële gedeelde geheim tussen twee apparaten genereert. De tweede is de zogenaamde Double Ratchet — dubbel tandwiel —, die de sleutels automatisch vernieuwt bij elk bericht, zodat het compromitteren van het apparaat vandaag het ontsleutelen van eerdere berichten niet toestaat, noch van toekomstige berichten zodra het tandwiel is gedraaid.

In Zig past de X25519-uitwisseling die het gedeelde geheim tussen twee apparaten genereert in zes regels, met gebruik van de standaardbibliotheek:

```
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)
```

Wat er gebeurt in die zes regels: De openbare sleutels reizen openlijk. De privésleutels verlaten nooit het respectieve apparaat. Elke partij leidt, op basis van zijn eigen privé- en de openbare sleutel van de andere partij, hetzelfde geheim van tweeëndertig bytes af dat niemand in het kanaal kan herstellen. Dat geheim dient later als kiem om de uitgewisselde berichten te versleutelen. De Double Ratchet van het Signal-protocol voegt een constante rotatie van dat materiaal toe, zodat het compromitteren van één moment de rest van het gesprek niet in gevaar brengt.

En wat zit er precies in `std.crypto.dh.X25519`? Geen verborgen magie. Het zijn twee korte functies die in hun geheel kunnen worden gelezen in de eigen standaardbibliotheek van Zig. De eerste leidt de openbare sleutel af uit de privésleutel — de «g^a» van de uitwisseling:

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

In de taal van het artikel: de privésleutel wordt «vermenigvuldigd» — in elliptische zin, niet in elementair rekenkundige zin — met het basispunt van de Curve25519-curve, en het resultaat wordt geserialiseerd naar tweeëndertig bytes. De operatie `clampedMul` is de geharde versie van die scalaire vermenigvuldiging: deze omvat de veiligheidsmaatregelen die de cryptografische gemeenschap in de loop der jaren heeft toegevoegd om weerstand te bieden aan bekende families van aanvallen. Twee regels functie-body.

De tweede functie combineert jouw privésleutel met de openbare sleutel die de andere partij je stuurt. Het is de «$(g^b)^a$» van de uitwisseling, die het tweeëndertig byte lange gedeelde geheim produceert dat geen van beiden ooit heeft uitgezonden:

```
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();
}
```

Nog twee regels. De ontvangen openbare sleutel wordt geïnterpreteerd als een punt op de curve en «vermenigvuldigd» met je eigen privésleutel. Door de commutativiteit van de curve-operatie — analoog aan de commutativiteit van de vermenigvuldiging van exponenten die we in het numerieke voorbeeld zagen — eindigen beide partijen met hetzelfde geserialiseerde punt: precies het gedeelde geheim waar het artikel over spreekt.

Dat is alles. Wat er in een applicatie als magie uitziet, zijn in werkelijkheid twee functies van elk drie regels. De technische complexiteit is geconcentreerd in een enkele operatie, `clampedMul`, die verderop in dezelfde standaardbibliotheek is geschreven, al tientallen jaren wordt beoordeeld door de internationale cryptografische gemeenschap, en beschikbaar is voor iedereen die het letter voor letter wil lezen. Er is geen zwarte doos, niet in onze applicatie en niet in de standaardbibliotheek van Zig. Er is open source code die een mens kan begrijpen, waarbij hij of zij zelf het tempo bepaalt om zich erin te verdiepen.

Wat end-to-end-versleuteling beschermt

Wat E2EE goed beschermt, uitgaande van een correcte implementatie, is de inhoud van het bericht tijdens transport. Een tussenliggende server die de versleutelde gegevens ontvangt en doorstuurt, ziet een opeenvolging van onbegrijpelijke bytes. Een aanvaller met toegang tot de kabel, de router, het wifi-toegangspunt ziet hetzelfde. Een dienstverlener die kopieën van het verkeer bewaart, zal dit achteraf niet kunnen lezen. Een regering die de operator van de dienst beveelt de inhoud te overhandigen, ontvangt dezelfde onbegrijpelijke bytes die de server in eerste instantie had.

Dat is in praktische termen veel. Het is het verschil tussen het schrijven van een brief in een ondoorzichtige envelop en het schrijven op een briefkaart. Beiden komen aan. Slechts één bewaart de inhoud voor de postbode.

Wat end-to-end-versleuteling niet beschermt

Het is de moeite waard dat net zo goed te weten. E2EE beschermt geen metadata: de server weet nog steeds dat gebruiker A gegevens naar gebruiker B stuurt, hoe laat, met welke frequentie en van waaruit, ook al weet hij niet wat ze zeggen. Deze metadata zijn, zoals we al hebben betoogd in *Versleutelen is niet privé zijn*, vaak onhullender dan de inhoud. Weten dat iemand op een vrijdag om 22:00 uur dertig minuten lang een advocatenkantoor belde dat gespecialiseerd is in echtscheidingen, vertelt een verhaal dat de inhoud van het gesprek nooit vertelde. Het is dezelfde situatie als iemand meerdere keren een oncologische kliniek zien binnengaan en verlaten: je hoeft niets te horen van wat er binnen wordt besproken om je voor te stellen wat er aan de hand is. Eén enkel los metagegeven hoeft niets te betekenen; meerdere kruislings gekoppelde gegevens tekenen iets dat te veel op de waarheid lijkt. E2EE beschermt de uiteinden niet: als het apparaat van de ontvanger is gecompromitteerd door een kwaadaardig programma, wordt het bericht normaal gesproken ontsleuteld voor die ontvanger en leest het kwaadaardige programma het. E2EE beschermt niet tegen de identiteit van de gesprekspartner op zich: als Alice gelooft dat ze met Bruno praat, maar een aanvaller zich in het begin heeft tussengevoegd (een *man in the middle*) und het protocol geen onafhankelijke verificatie bevat, eindigen beide partijen door met de indringer te praten terwijl ze denken dat ze met elkaar praten.

Er is een vierde ding dat de moeite waard is om zonder ambiguïteit te formuleren. E2EE belet niet dat een provider die beweert het aan te bieden, bovendien een kopie van het niet-versleutelde bericht in zijn eigen systemen bewaart. De uitspraak «mijn berichten zijn end-to-end versleuteld» en de uitspraak «de provider bewaart mijn inhoud niet» zijn niet hetzelfde. Een applicatie kan aan de eerste voldoen terwijl hij de tweede

schendt; we hebben dit sinds 2018 herhaaldelijk in krantenkoppen gezien. De gebruiker heeft, tenzij de code van de client verifieerbaar is, geen technische manier om het ene geval van het andere te onderscheiden zonder deskundig onderzoek. Het bekendste geval bij het grote publiek: WhatsApp versleutelt berichten end-to-end tijdens transport, maar als de gebruiker de back-up in iCloud of Google Drive activeert zonder aanvullende versleuteling, wordt die kopie leesbaar opgeslagen in de infrastructura van een derde partij, en wordt de versleuteling aan het uiteinde van de gebruiker zelf verbroken.

De vraag die de operator niet wil horen

Een applicatie die beweert end-to-end te versleutelen kan technisch gezien een van drie dingen doen met betrekking tot de sleutels:

1. **De sleutels bevinden zich alleen op de apparaten.** Ze worden uitsluitend op de apparaten van de gebruikers gegenereerd en bevinden zich daar; de operator kent ze niet en slaat ze niet op. Dit is het optimale geval.
2. **De operator heeft toegang als hij dat wil.** De operator bezit de sleutels van de gebruikers (of kan deze naar eigen believen genereren) en bewaart deze in zijn databases. Als hij dat wil of daartoe gedwongen wordt, kan hij de inhoud lezen. Dit is het geval bij de meeste «cloud»-diensten.
3. **De operator heeft ontwerpmatig geen toegang, maar controleert de toegang.** De operator heeft de sleutels niet, maar heeft controle over de applicatie die ze genereert. Indien daartoe gedwongen, kan hij een kwaadaardige update sturen die de sleutels of de inhoud onderschept vóór de versleuteling. Dit is het geval bij veel commerciële E2EE-diensten.

De operationele vraag is daarom niet of iets versleuteld is, maar wie de controle heeft over het apparaat en de software die de sleutels beheert. Bij Solo2 bevinden de sleutels zich uitsluitend in je Vault (met je wachtwoord versleutelde IndexedDB) en is de software verifieerbare open source.

Voor de professionele lezer

End-to-end-versleuteling is een hulpmiddel voor digitale soevereiniteit. Maar zoals elk hulpmiddel hangt de effectiviteit ervan af van de hand die het hanteert en de grond waarop het rust.

1. Waar worden de cryptografische sleutels gegenereerd en waar bevinden ze zich fysiek? Als de operator er toegang toe heeft (zelfs tijdelijk, zelfs onder het voorwendsel van herstel), is de E2EE slechts nominaal.
2. Is er onafhankelijke verificatie van de gesprekspartner (beveiligingsnummers, QR-codes, out-of-band vergelijking) die een man-in-the-middle-aanval voorkomt tijdens het tot stand brengen van het gesprek?
3. Is de code van de client controleerbaar — open, gepubliceerd, reproduceerbaar — of vereist het vertrouwen in het woord van de aanbieder over wat de client daadwerkelijk doet?
4. Welke metadata genereert en bewaart de dienst, en voor hoe lang? Zelfs als de inhoud ondoorzichtig is, kunnen metadata een groot deel van de gevoelige informatie reconstrueren.

Deze vier vragen vragen niet om geavanceerde technische informatie; ze vragen om informatie die elke eerlijke operator in zijn openbare documentatie kan beantwoorden. De kwaliteit en precisie van het antwoord zeggen evenveel over het product als het antwoord zelf.

End-to-end-versleuteling, mits goed uitgevoerd, is een van de fraaiste constructies die de hedendaagse cryptografie aan de dagelijkse praktijk heeft geleverd. Het oorspronkelijke idee — twee mensen kunnen via een openbaar kanaal een geheim afspreken — is van Whitfield Diffie en Martin Hellman, 1976; een halve eeuw later leven we nog steeds in de gevolgen daarvan. Maar zoals bij elke technische belofte hangt de waarde af van de werkelijke vervulling, niet van het label. De vraag van de eerlijke professional is niet «is het versleuteld?», maar «wie heeft de sleutels?». De antwoorden hebben verschillende gevolgen. Het is de moeite waard die te kennen.

Bronnen und verdere lectuur

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, november 1976. Fundamenteel artikel over public-key cryptografie.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, openbare specificatie door Open Whisper Systems, revisie 2016. Basis van het Signal-protocol en zijn industriële afgeleiden.
- RFC 7748 — Elliptic Curves for Security (IETF, januari 2016). Normatieve specificatie van de X25519- en X448-curven die worden gebruikt in moderne sleuteluitwisselingen.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Hoofdstukken over sleuteluitwisseling en geauthenticeerde versleutelingsprotocollen.
- Verordening (EU) 2024/1183 betreffende een kader voor een Europese digitale identiteit (eIDAS 2) — stelt kaders vast waarbinnen onafhankelijke verificatie van de gesprekspartner institutionele steun krijgt, en waar het onderscheid tussen nominale en echte versleuteling verschillende juridische gevolgen heeft.

← VorigeKill switch en institutionele inbeslagnameVolgende →Het bedrijfsmodel als signaal van vertrouwen

Recente artikelen

- Analyse · 18 mei 2026 Echte vs. schijnbare privacy: de vragen die men zich moet stellen
- Analyse · 18 mei 2026 Self-hosting als professionele praktijk
- Concept · 18 mei 2026 De 24 woorden: wat een cryptografische identiteit is

Neem dit artikel mee naar waar u het nodig heeft.

↓ Markdown ↓ Platte tekst ↓ PDF

Het bestand wordt gedownload naar uw apparaat. Van daaruit kunt u het opslaan, importeren in Solo2 of delen waar u maar wilt. Cuadernos beslist niet voor u over de bestemming.

Lakzegel · SHA-256 f016c3fbf729c2d4059e92e264da4115b78aece2d977d744184c7184e59fb26e

Cuadernos Lacre · Een uitgave van Menzuri Gestión S.L. ·
geschreven door R.Eugenio · geredigeerd door het team van Solo2.