Enkripsi end-to-end, dijelaskan dengan sungguh-sungguh

Apa yang dikatakan penyedia saat mereka menyebut E2EE, और apa yang tidak mereka katakan. Penjelasan didaktik tentang mekanisme dan batasannya, tanpa bungkus pemasaran.

Biar jelas: WhatsApp mengatakan pesan Anda dienkripsi end-to-end. Itu benar — dan itu tidak cukup. Jika cadangan masuk ke iCloud atau Google Drive tanpa enkripsi tambahan, enkripsi tersebut akan rusak di ponsel Anda sendiri. Pertanyaan operasionalnya bukanlah apakah ia dienkripsi, melainkan di mana kunci tersebut berada.

Arti enkripsi yang sebenarnya

Mengenkripsi pesan berarti mengubahnya menjadi sesuatu yang tampak seperti derau bagi siapa pun yang tidak memiliki informasi tertentu yang disebut kunci. Operasi dilakukan pada perangkat pengirim dan, dengan kunci yang benar, dibatalkan pada perangkat penerima. Di antaranya, pesan berpindah sebagai urutan bita tanpa makna yang jelas. Itulah ide sederhananya. Sisa artikel ini membahas nuansa yang mengubahnya, tergantung kasusnya, menjadi jaminan nyata atau sekadar label pemasaran.

Kata sifat *end-to-end* — dalam bahasa Inggris *end-to-end*, disingkat E2EE — menambahkan ketepatan. Enkripsi tidak dilakukan agar server perantara dapat membaca dan mengirimkannya. Ini dilakukan agar hanya kedua ujung — perangkat pengirim dan perangkat penerima — yang memiliki kuncinya. Server mana pun yang dilewati pesan hanya melihat derau, bukan pesan. Itulah perbedaan teknis dengan enkripsi *dalam transit*, di mana konten berpindah terenkripsi dari satu server ke server berikutnya, tetapi setiap server yang dilewatinya mendekripsinya untuk meneruskannya, memulihkan teks asli secara sementara.

Paradoks rahasia bersama

Ada masalah yang jelas. Agar dua orang dapat mengenkripsi dan mendekripsi pesan di antara mereka sendiri, keduanya memerlukan kunci yang sama. Namun, bagaimana mereka menyepakati kunci tersebut jika semua yang mereka kirimkan satu sama lain, menurut definisi, melewati saluran di mana seseorang mungkin mendengarkan? Menyepakati kunci di saluran yang sama yang nantinya akan mereka gunakan tampak mustahil: jika penyerang mendengarnya saat menyepakatinya, mereka akan dapat mendekripsi semua hal berikutnya. Selama beberapa dekade, kriptografi klasik memecahkan masalah ini dengan cara yang sulit: kunci dikirimkan secara langsung, sebelum mulai digunakan, dalam pertemuan fisik. Duta besar membawa koper berisi kunci yang dijahit ke dalam lapisan mantel mereka.

Dalam email kontemporer, solusi tersebut tidak berskala. Jika kita harus pergi secara fisik ke rumah setiap orang yang ingin kita ajak berkomunikasi secara terenkripsi, kita tidak akan sempat berbicara dengan siapa pun. Pertanyaan yang diajukan lima puluh tahun lalu oleh komunitas kriptografi adalah ini: apakah mungkin bagi dua orang yang tidak saling mengenal dan hanya berbagi saluran publik untuk menyepakati, di saluran publik yang sama, sebuah rahasia yang tidak dapat diketahui oleh siapa pun yang mendengarkan saluran tersebut?

Keanggunan Diffie-Hellman

Pada tahun 1976, dua matematikawan bernama Whitfield Diffie dan Martin Hellman mendemonstrasikan sesuatu yang tampak mustahil: bahwa dua orang, yang hanya berbicara melalui saluran publik — saluran di mana siapa pun dapat mendengar semua yang mereka katakan — dapat menyepakati kata sandi rahasia tanpa ada pendengar yang dapat menemukannya. Kedengarannya seperti sihir. Padahal bukan: itu matematika. Pertukaran kunci Diffie-Hellman, sebagaimana dikenal sejak saat itu, adalah dasar dari hampir semua komunikasi terenkripsi di internet, dan penggunaan intensif selama setengah abad serta pengawasan akademis global mengonfirmasi kekuatannya. Siapa pun yang ingin melihat intuisi visual atau matematika dapat terus membaca. Siapa pun yang lebih suka percaya bahwa itu berhasil juga dapat melanjutkan tanpa kehilangan alur artikel.

Bagi siapa pun yang ingin merasakannya dalam sebuah gambar, ada analogi yang dikenal dengan warna. Bayangkan Alice dan Bruno menyepakati warna dasar secara terbuka — katakanlah kuning — di depan Eva, yang mendengarkan mereka. Masing-masing memilih warna rahasia kedua secara pribadi dan mencampur rahasia mereka dengan warna kuning. Alice mendapatkan oranye tertentu; Bruno mendapatkan hijau tertentu. Mereka menukar hasilnya di depan Eva. Sekarang masing-masing mencampur warna yang diterima dengan rahasia mereka sendiri, dan keduanya sampai pada warna akhir yang sama, karena urutan pencampuran tidak masalah. Eva telah melihat warna kuning dan dua campuran perantara, tetapi bukan rahasianya; tanpa salah satu rahasia tersebut dia tidak dapat mencapai warna akhir. Matematika yang sebenarnya menggantikan warna untuk eksponensiasi dalam grup modular atau kurva elips, tetapi idenya sama: rahasia bersama dibangun di depan umum tanpa ada orang di saluran tersebut yang dapat merekonstruksinya.

Dalam aritmatika, bagi siapa pun yang lebih suka melihat mekanismenya: Alice memilih nomor rahasia a, Bruno memilih b. Mereka menukar g^a dan g^b secara terbuka di saluran tersebut. Alice menghitung $(g^b)^a$ dan Bruno menghitung $(g^a)^b$; keduanya sampai pada g^{ab} yang sama.

Eva melihat g, g^a, dan g^b melewati saluran tersebut, tetapi memulihkan a dari g^a — yang disebut masalah logaritma diskrit — memerlukan waktu komputasi yang secara astronomis lebih tinggi daripada usia alam semesta ketika g dipilih dalam grup matematika yang sesuai.

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.

Dari Diffie-Hellman ke protokol Signal

Enkripsi end-to-end yang digunakan oleh aplikasi perpesanan profesional saat ini bertumpu, hampir tanpa kecuali, pada versi pertukaran Diffie-Hellman yang elegan dan diperkuat. Protokol Signal, yang dirancang oleh Trevor Perrin dan Moxie Marlinspike antara tahun 2013 dan 2016, adalah referensinya. Ini menggabungkan dua ide kunci. Pertama, pertukaran kunci dalam kurva elips (X25519), yang menghasilkan rahasia bersama awal antara dua perangkat. Kedua, yang disebut Double Ratchet — roda gigi ganda — yang memperbarui kunci secara otomatis dengan setiap pesan, sehingga mengompromikan perangkat hari ini tidak memungkinkan dekripsi pesan masa lalu, atau pesan masa depan setelah roda gigi diputar.

Di Zig, pertukaran X25519 yang menghasilkan rahasia bersama antara dua perangkat muat dalam enam baris, menggunakan pustaka standar:

```
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)
```

Apa yang terjadi dalam enam baris tersebut: Kunci publik berpindah secara terbuka. Kunci privat tidak pernah meninggalkan perangkat masing-masing. Setiap pihak menurunkan, dari kunci privatnya dan kunci publik pihak lain, rahasia tiga puluh dua bita yang sama yang tidak dapat dipulihkan oleh siapa pun di saluran tersebut. Rahasia itu kemudian berfungsi sebagai benih untuk mengenkripsi pesan yang dipertukarkan. Double Ratchet dari protokol Signal menambahkan rotasi konstan materi tersebut sehingga kompromi sesaat tidak mengompromikan sisa percakapan.

Dan apa sebenarnya yang ada di dalam `std.crypto.dh.X25519`? Tidak ada keajaiban yang tersembunyi. Itu adalah dua fungsi pendek yang dapat dibaca secara keseluruhan di pustaka standar Zig itu sendiri. Yang pertama menurunkan kunci publik dari kunci privat — «g^a» dari pertukaran tersebut:

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

Dalam bahasa artikel ini: kunci privat «dikalikan» — dalam pengertian elips, bukan aritmatika dasar — oleh titik dasar kurva Curve25519, dan hasilnya diserialisasi menjadi tiga puluh dua bita. Operasi `clampedMul` adalah versi yang diperkuat dari perkalian skalar tersebut: ia menggabungkan perlindungan yang ditambahkan komunitas kriptografi selama bertahun-tahun untuk menahan keluarga serangan yang diketahui. Dua baris badan fungsi.

Fungsi kedua menggabungkan kunci privat Anda dengan kunci publik yang dikirimkan pihak lain kepada Anda. Ini adalah «$(g^b)^a$» dari pertukaran, yang menghasilkan rahasia bersama tiga puluh dua bita yang tidak pernah ditransmisikan oleh kalian berdua:

```
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();
}
```

Dua baris lagi. Kunci publik yang diterima ditafsirkan sebagai titik pada kurva, dan «dikalikan» dengan kunci privat sendiri. Dengan komutativitas operasi kurva — yang analog dengan komutativitas perkalian eksponen yang kita lihat dalam contoh numerik — kedua belah pihak berakhir dengan titik serial yang sama: persis rahasia bersama yang dibicarakan dalam artikel.

Itu saja. Apa yang dalam sebuah aplikasi terlihat seperti keajaiban, pada kenyataannya, adalah dua fungsi yang masing-masing terdiri dari tiga baris. Kompleksitas teknis terkonsentrasi dalam satu operasi, `clampedMul`, yang ditulis lebih jauh di bawah dalam pustaka standar yang sama, ditinjau selama beberapa dekade oleh komunitas kriptografi internasional, dan tersedia bagi siapa saja yang ingin membacanya huruf demi huruf. Tidak ada kotak hitam baik di aplikasi kami maupun di pustaka standar Zig. Ada kode sumber terbuka yang dapat dipahami manusia, sambil memilih kecepatan yang mereka inginkan untuk mendalaminya.

Apa yang dilindungi oleh enkripsi end-to-end

Apa yang dilindungi E2EE dengan baik, dengan asumsi implementasi yang benar, adalah konten pesan dalam transit. Server perantara yang menerima dan meneruskan data terenkripsi akan melihat urutan bita yang tidak dapat dipahami. Penyerang dengan akses ke kabel, router, titik akses wifi, akan melihat hal yang sama. Penyedia layanan yang menyimpan salinan lalu lintas tidak akan dapat membacanya di kemudian hari. Pemerintah yang memerintahkan operator layanan untuk menyerahkan konten akan menerima bita tidak dapat dipahami yang sama dengan yang dimiliki server pada awalnya.

Ini, dalam istilah praktis, sangat banyak. Ini adalah perbedaan antara menulis surat di dalam amplop buram dan menulisnya di kartu pos. Keduanya sampai. Hanya satu yang menjaga konten dari tukang pos.

Apa yang tidak dilindungi oleh enkripsi end-to-end

Penting untuk mengetahuinya dengan baik. E2EE tidak melindungi metadata: server masih mengetahui bahwa pengguna A mengirimkan data ke pengguna B, jam berapa, dengan frekuensi berapa, dan dari mana, meskipun tidak tahu apa yang dikatakannya. Metadata ini, seperti yang telah kami argumenkan dalam *Enkripsi bukan berarti pribadi*, seringkali lebih mengungkapkan daripada kontennya. Mengetahui bahwa seseorang menelepon firma hukum spesialis perceraian pada hari Jumat pukul 22:00 selama tiga puluh menit menceritakan sebuah kisah yang tidak pernah diceritakan oleh konten panggilan tersebut. Ini adalah situasi yang sama dengan melihat seseorang masuk dan keluar beberapa kali dari klinik onkologi: tidak perlu mendengar apa pun yang dibicarakan di dalam untuk membayangkan apa yang sedang terjadi. Satu metadata yang terisolasi mungkin tidak berarti apa-apa; beberapa yang saling terkait menggambar sesuatu yang terlalu mirip dengan kebenaran. E2EE tidak melindungi ujung-ujungnya: jika perangkat penerima dikompromikan oleh program jahat, pesan didekripsi secara normal untuk penerima tersebut dan program jahat membacanya. E2EE tidak melindungi terhadap identitas lawan bicara itu sendiri: jika Alice yakin dia berbicara dengan Bruno tetapi penyerang telah menyisipkan diri di awal (seorang *man in the middle*) dan protokol tidak menyertakan verifikasi independen, kedua pihak akhirnya berbicara dengan penyusup sambil mengira mereka sedang berbicara satu sama lain.

Ada hal keempat yang layak dirumuskan tanpa ambiguitas. E2EE tidak mencegah penyedia yang mengklaim menawarkannya untuk juga menyimpan salinan pesan yang tidak terenkripsi di sistemnya sendiri. Pernyataan «pesan saya terenkripsi end-to-end» dan pernyataan «penyedia tidak menyimpan konten saya» tidaklah sama. Sebuah aplikasi dapat memenuhi pernyataan pertama sambil melanggar pernyataan kedua; kita telah melihatnya di tajuk berita berulang kali sejak 2018. Pengguna, kecuali kode klien dapat diverifikasi, tidak memiliki cara teknis untuk membedakan satu kasus dari kasus lainnya tanpa investigasi ahli. Kasus yang paling dikenal publik: WhatsApp mengenkripsi pesan end-to-end dalam transit, tetapi jika pengguna mengaktifkan cadangan di iCloud atau Google Drive tanpa enkripsi tambahan, salinan itu disimpan secara terbaca di infrastruktur pihak ketiga, dan enkripsi rusak di ujung pengguna itu sendiri.

Pertanyaan yang tidak ingin didengar oleh operator

Aplikasi yang mengklaim mengenkripsi end-to-end dapat, secara teknis, melakukan satu dari tiga hal terkait kunci:

1. **Kunci hanya berada di perangkat.** Kunci dibuat dan berada secara eksklusif di perangkat pengguna; operator tidak mengetahuinya atau menyimpannya. Ini adalah kasus yang optimal.
2. **Operator dapat mengakses jika mereka mau.** Operator memiliki kunci pengguna (atau dapat membuatnya sesuka hati) dan menyimpannya di basis data mereka. Jika mereka mau atau dipaksa, mereka dapat membaca kontennya. Ini adalah kasus untuk sebagian besar layanan 'cloud'.
3. **Operator tidak dapat mengakses berdasarkan desain, tetapi mengontrol akses.** Operator tidak memiliki kunci, tetapi memiliki kendali atas aplikasi yang membuatnya. Jika dipaksa, mereka dapat mengirim pembaruan berbahaya yang menangkap kunci atau konten sebelum dienkripsi. Ini adalah kasus bagi banyak layanan E2EE komersial.

Oleh karena itu, pertanyaan operasionalnya bukanlah apakah sesuatu itu dienkripsi, melainkan siapa yang memiliki kendali atas perangkat dan perangkat lunak yang mengelola kunci. Di Solo2, kunci hanya berada di Brankas Anda (IndexedDB yang dienkripsi dengan kata sandi Anda) dan perangkat lunaknya adalah kode sumber terbuka yang dapat diverifikasi.

Untuk pembaca profesional

Enkripsi end-to-end adalah alat untuk kedaulatan digital. Namun seperti setiap alat, efektivitasnya bergantung pada tangan yang memegangnya dan landasan tempat ia berpijak.

1. Di mana kunci kriptografi dibuat dan di mana kunci tersebut secara fisik berada? Jika operator dapat mengaksesnya (meskipun sementara, bahkan dengan kedok pemulihan), E2EE tersebut hanya nominal.
2. Apakah ada verifikasi independen dari lawan bicara (nomor keamanan, kode QR, perbandingan di luar pita) yang mencegah serangan man-in-the-middle selama penetapan percakapan?
3. Apakah kode klien dapat diaudit — terbuka, dipublikasikan, dapat diproduksi ulang — atau apakah itu mengharuskan untuk memercayai kata-kata penyedia tentang apa yang sebenarnya dilakukan klien?
4. Metadata apa yang dihasilkan dan disimpan layanan, dan untuk berapa lama? Meskipun kontennya tidak transparan, metadata dapat merekonstruksi sebagian besar informasi sensitif.

Keempat pertanyaan ini tidak meminta informasi teknis lanjutan; mereka meminta informasi yang dapat dijawab oleh operator yang jujur mana pun dalam dokumentasi publik mereka. Kualitas dan presisi jawaban mengatakan tentang produk sama halnya dengan jawaban itu sendiri.

Enkripsi end-to-end, jika dilakukan dengan benar, adalah salah satu konstruksi terbaik yang diberikan kriptografi kontemporer kepada praktik sehari-hari. Ide aslinya — dua orang dapat menyetujui sebuah rahasia di saluran publik — milik Whitfield Diffie dan Martin Hellman, 1976; setengah abad kemudian kita terus hidup dalam konsekuensinya. Namun, seperti halnya janji teknis lainnya, nilainya bergantung pada pemenuhan nyata, bukan pada labelnya. Pertanyaan profesional yang jujur bukanlah «apakah ini dienkripsi?», melainkan «siapa yang memegang kuncinya?». Jawabannya memiliki konsekuensi yang berbeda. Ada baiknya mengetahuinya.

Sumber dan bacaan lebih lanjut

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, November 1976. Artikel dasar tentang kriptografi kunci publik.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, spesifikasi publik oleh Open Whisper Systems, revisi 2016. Basis dari protokol Signal dan turunannya di industri.
- RFC 7748 — Elliptic Curves for Security (IETF, Januari 2016). Spesifikasi normatif dari kurva X25519 dan X448 yang digunakan dalam pertukaran kunci modern.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Bab-bab tentang pertukaran kunci dan protokol enkripsi yang diautentikasi.
- Regulasi (UE) 2024/1183 tentang kerangka identitas digital Eropa (eIDAS 2) — menetapkan kerangka kerja di mana verifikasi independen lawan bicara memperoleh dukungan institusional, dan di mana perbedaan antara enkripsi nominal dan nyata memiliki konsekuensi hukum yang berbeda.

← SebelumnyaKill switch dan penangkapan institusionalBerikutnya →Model bisnis sebagai sinyal kepercayaan

Bacaan terbaru

- Analisis · 18 Mei 2026 Privasi nyata vs semu: pertanyaan yang perlu Anda ajukan

- Analisis · 18 Mei 2026 Self-hosting sebagai praktik profesional
- Konsep · 18 Mei 2026 24 kata: apa itu identitas kriptografi

Bawa artikel ini bersama Anda ke mana pun Anda membutuhkannya.

↓ Markdown ↓ Teks murni ↓ PDF

File akan diunduh ke perangkat Anda. Dari sana Anda dapat menyimpannya, mengimpornya ke Solo2, atau membagikannya di mana pun Anda mau. Cuadernos tidak memutuskan tujuan untuk Anda.

Segel lilin · SHA-256 82a51f766ca2e1d322456df8fffd678a0147dc45902814f0c45e81c90c838e43

Cuadernos Lacre · Publikasi dari Menzuri Gestión S.L. ·
ditulis oleh R.Eugenio · disunting oleh tim Solo2.