رمزنگاری سرتاسری، به زبان واقعیت

آنچه ارائه‌دهندگان هنگام گفتن E2EE می‌گویند و آنچه نمی‌گویند. توضیحی آموزشی از مکانیسم و محدودیت‌های آن، بدون پوشش تبلیغاتی.

برای اینکه روشن شود: WhatsApp می‌گوید پیام‌های شما سرتاسری رمزگذاری شده‌اند. این درست است — و کافی نیست. اگر نسخه پشتیبان بدون رمزگذاری اضافی به iCloud یا Google Drive برود، رمزگذاری در گوشی خود شما شکسته می‌شود. سؤال عملی این نیست که آیا رمزگذاری شده است یا خیر، بلکه این است که کلیدها کجا قرار دارند.

معنای واقعی رمزنگاری

رمزنگاری یک پیام به معنای تبدیل آن به چیزی است که برای هر کسی که اطلاعات خاصی به نام کلید را در اختیار ندارد، مانند نویز به نظر می‌رسد. این عملیات در دستگاه فرستنده انجام می‌شود و با کلید صحیح، در دستگاه گیرنده بازگشایی می‌شود. در این میان، پیام به عنوان توالی از بایت‌ها بدون معنای ظاهری منتقل می‌شود. این یک ایده ساده است. بقیه مقاله به جزئیاتی می‌پردازد که بسته به مورد، آن را به یک ضمانت واقعی یا صرفاً یک برچسب تبلیغاتی تبدیل می‌کند.

صفت *سرتاسری* —در انگلیسی *end-to-end*، به اختصار E2EE— دقت بیشتری را اضافه می‌کند. رمزنگاری برای این انجام نمی‌شود که یک سرور واسطه بتواند آن را بخواند و تحویل دهد. بلکه به این دلیل انجام می‌شود که فقط دو طرف —دستگاه فرستنده و دستگاه گیرنده— کلید را در اختیار داشته باشند. هر سروری که پیام از آن عبور می‌کند، نویز را می‌بیند، نه پیام را. این تفاوت فنی با رمزنگاری *در حال انتقال* است، جایی که محتوا از یک سرور به سرور بعدی به صورت رمزنگاری شده منتقل می‌شود، اما هر سروری که از آن عبور می‌کند، آن را برای ارسال مجدد رمزگشایی می‌کند و متن را به طور موقت به صورت آشکار بازیابی می‌کند.

پارادوکس راز مشترک

یک مشکل واضح وجود دارد. برای اینکه دو نفر بتوانند پیام‌ها را بین خود رمزنگاری و رمزگشایی کنند، هر دو به یک کلید نیاز دارند. اما، چگونه بر سر این کلید توافق می‌کنند در حالی که هر چه برای هم می‌فرستند، طبق تعریف، از کانالی عبور می‌کند که ممکن است کسی در حال گوش دادن به آن باشد؟ توافق بر سر کلید در همان کانالی که بعداً از آن استفاده خواهند کرد، غیرممکن به نظر می‌رسد: اگر مهاجم هنگام توافق بر سر آن، آن را بشنود، می‌تواند تمام موارد بعدی را رمزگشایی کند. برای دهه‌ها، رمزنگاری کلاسیک این مشکل را از راه سخت حل می‌کرد: کلیدها به صورت حضوری، قبل از شروع استفاده، در ملاقات‌های فیزیکی تحویل داده می‌شدند. سفیران کیف‌های کلید را که در آستر کتشان دوخته شده بود، حمل می‌کردند.

در ایمیل‌های امروزی، آن راه حل مقیاس‌پذیر نیست. اگر مجبور بودیم به طور فیزیکی به خانه هر کسی که قصد داشتیم با او به صورت رمزنگاری شده ارتباط برقرار کنیم برویم، هرگز نمی‌توانستیم با کسی صحبت کنیم. سؤالی که پنجاه سال پیش توسط جامعه رمزنگاری مطرح شد این بود: آیا ممکن است دو نفر که همدیگر را نمی‌شناسند و فقط یک کانال عمومی مشترک دارند، در همان کانال عمومی بر سر رازی توافق کنند که هیچ کس که به کانال گوش می‌دهد نتواند آن را بداند؟

ظرافت Diffie-Hellman

در سال ۱۹۷۶، دو ریاضی‌دان به نام‌های Whitfield Diffie و Martin Hellman چیزی به ظاهر غیرممکن را نشان دادند: اینکه دو نفر که فقط از طریق یک کانال عمومی صحبت می‌کنند —کانالی که هر کسی می‌تواند هر آنچه می‌گویند بشنود— می‌توانند بدون اینکه هیچ شنونده‌ای بتواند آن را کشف کند، بر سر یک رمز عبور مخفی توافق کنند. شبیه جادو به نظر می‌رسد، اما اینطور نیست: این ریاضیات است. تبادل کلید Diffie-Hellman، همانطور که از آن زمان شناخته شده است، پایه و اساس تقریباً تمام ارتباطات رمزنگاری شده اینترنت است و نیم قرن استفاده فشرده و بررسی‌های دقیق آکادمیک جهانی، استحکام آن را تأیید می‌کند. هر کسی که می‌خواهد درک شهودی بصری یا ریاضیات را ببیند، می‌تواند به خواندن ادامه دهد. هر کسی هم که ترجیح می‌دهد به کارکرد آن اعتماد کند، می‌تواند بدون از دست دادن رشته مقاله، ادامه دهد.

برای کسی که می‌خواهد آن را در یک تصویر تصور کند، یک آنالوژی شناخته شده با رنگ‌ها وجود دارد. تصور کنید که آلیس و برونو در مقابل چشم ایوا که به آن‌ها گوش می‌دهد، بر سر یک رنگ پایه —مثلاً زرد— به طور علنی توافق می‌کنند. هر کدام در خلوت یک رنگ مخفی دوم را انتخاب کرده و راز خود را با رنگ زرد مخلوط می‌کنند. آلیس یک رنگ نارنجی خاص به دست می‌آورد؛ برونو یک رنگ سبز خاص به دست می‌آورد. آن‌ها نتایج را در مقابل چشم ایوا با هم عوض می‌کنند. حالا هر کدام رنگ دریافتی را با راز خود مخلوط می‌کنند و هر دو به همان رنگ نهایی می‌رسند، زیرا ترتیب مخلوط کردن فرقی نمی‌کند. ایوا رنگ زرد و دو مخلوط واسطه را دیده است، اما رازها را ندیده؛ بدون داشتن یکی از رازها، او نمی‌تواند به رنگ نهایی برسد. ریاضیات واقعی، رنگ‌ها را با توان‌رسانی در گروه‌های پیمانه‌ای یا منحنی‌های بیضوی جایگزین می‌کند، اما ایده همان است: راز مشترک در ملاء عام ساخته می‌شود بدون اینکه کسی در کانال بتواند آن را بازسازی کند.

در علم حساب، برای کسانی که ترجیح می‌دهند مکانیسم را ببینند: آلیس یک عدد مخفی a انتخاب می‌کند، برونو b را انتخاب می‌کند. آن‌ها g^a و g^b را به صورت آشکار در کانال مبادله می‌کنند. آلیس $(g^b)^a$ را محاسبه می‌کند و برونو $(g^a)^b$ را محاسبه می‌کند؛ هر دو به همان g^{ab} می‌رسند. ایوا عبور g، g^a و g^b را از کانال می‌بیند، اما بازیابی a از g^a — که مسئله لگاریتم گسسته نامیده می‌شود— زمانی که g در یک گروه ریاضی مناسب انتخاب شود، به زمان محاسباتی نجومی، فراتر از سن جهان، نیاز دارد.

برای کسانی که می‌خواهند آن را با اعداد کوچک بررسی کنند. تبادل Diffie-Hellman را می‌توان به طور کامل با ارقامی به اندازه کافی کوچک طی کرد تا محاسبات با دست انجام شود. هر کسی که ترجیح می‌دهد وارد حساب نشود، می‌تواند از این بلوک بپرد بدون اینکه رشته مقاله را از دست بدهد؛ هر کسی که می‌خواهد مکانیسم را مرحله به مرحله در حال کار ببیند، آن را اینجا پیدا می‌کند. **قوانین عمومی**، که هر کسی می‌تواند بخواند: یک عدد اول $p = 11$ (در Diffie-Hellman واقعی حدود سیصد رقم است؛ ما از یازده استفاده می‌کنیم تا محاسبات در یک صفحه جا شوند)، یک پایه $g = 2$، و این قرارداد که تمام حساب‌ها به *پیمانه* p (مادولو p) انجام می‌شود — محاسبه می‌کنید، بر p تقسیم می‌کنید و باقیمانده را نگه می‌دارید، مانند یک ساعت یازده موقعیتی که با گذشتن از ده به صفر باز می‌گردد. **انتخاب‌های خصوصی**، هر کدام یکی و هرگز به اشتراک گذاشته نمی‌شوند: آلیس $a = 4$ را انتخاب می‌کند. برونو $b = 7$ را انتخاب می‌کند.

مرحله ۱. آلیس $2^4 = 16$ را محاسبه می‌کند، سپس $16 \bmod 11 = 5$. او پنج را ارسال می‌کند. ایوا آن را یادداشت می‌کند.

مرحله ۲. برونو $2^7 = 128$ را محاسبه می‌کند، سپس $128 \bmod 11 = 7$. او هفت را ارسال می‌کند. ایوا نیز آن را یادداشت می‌کند. پس از دو ارسال، دفترچه ایوا شامل چهار داده است: $p = 11$، $g = 2$، A = 5، B = 7. او عدد مشترکی را که آلیس و برونو در حال استخراج آن هستند — و ایوا نمی‌تواند آن را بازسازی کند، کم دارد.

مرحله ۳. آلیس هفتی را که برونو برای او فرستاده می‌گیرد و آن را به توان مخفی خود $a = 4$ می‌رساند. برای جلوگیری از کار با $7^4 = 2401$، محاسبه به صورت بخش به بخش با اعمال پیمانه در هر مرحله انجام می‌شود:

$$7^2 = 49$$

$$49 \bmod 11 = 5$$

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

mod 11 = 3 25

آلیس عدد **3** را به دست می‌آورد.

مرحله ۴. برونو پنجی را که آلیس برای او فرستاده می‌گیرد و آن را به توان مخفی خود $b = 7$ می‌رساند. دوباره به صورت بخش به بخش:

$$25 = 5^2 \mod 11 = 3$$

$$9 = 3^2 = {}^2(5^2) = 5^4 \mod 11 = 9$$

$$27 = 3 \times 9 = 5^2 \times 5^4 = 5^6 \mod 11 = 5$$

در نهایت $25 = 5 \times 5 = 5 \times 5^6 = 5^7 \mod 11 = \mathbf{3}$.

برونو نیز **3** را به دست می‌آورد.

هر دو با کار موازی به یک عدد، یعنی 3 رسیده‌اند. هیچ کدام توان مخفی خود را در هیچ زمانی ارسال نکردند. آلیس نمی‌داند که $b = 7$ است؛ برونو نمی‌داند که $a = 4$ است. هر یک از مقدار عمومی که دیگری فرستاده همراه با توان مخفی خود استفاده کرد و آن‌ها در یک مقصد با هم روبرو شدند. **چرا آن‌ها به یک عدد می‌رسند؟** آنچه هر کدام محاسبه کردند: آلیس، $(g^b) \mod 11 = 2^{28} = 2^{7x4} = {}^a$. برونو، $(g^a) \mod 11 = 2^{28} = 2^{4x7} = {}^b$. این همان مقدار است زیرا ترتیب ضرب توان‌ها مهم نیست ($7 \times 4 = 4 \times 7$). هر یک از راهی متفاوت به یک مقصد رسیدند.

و ایوا؟ او در دفترچه خود $p = 11$، $g = 2$، $A = 5$، $B = 7$ را دارد و 3 را می‌خواهد. برای محاسبه آن باید a یا b را بداند — اما هیچ کدام از طریق کانال منتقل نشده‌اند. تنها راه او این است که از خود بپرسد: «برای چه توانی a، معادله $2^a \mod 11 = 5$ برقرار است؟». با p به این کوچکی او می‌تواند 0، 1، 2، 3، 4... را امتحان کند و در کمتر از یک دقیقه آن را پیدا کند. اما وقتی 11 را با یک عدد اول سیصد رقمی جایگزین می‌کنیم، فضای توان‌های ممکن عناصر بیشتری از تعداد اتم‌های جهان قابل مشاهده دارد. **در حال حاضر هیچ الگوریتمی برای بشریت شناخته نشده است که بتواند این فضا را در کمتر از میلیاردها سال طی کند.** این همان *مسئله لگاریتم گسسته* است: رو به جلو آسان، رو به عقب از نظر محاسباتی غیرممکن. و این دلیلی است که رمزنگاری مقاومت می‌کند حتی اگر ایوا تمام مکالمه را حرف به حرف دنبال کرده باشد.

سه عنصر ساده — حساب ساعت، توان‌رسانی، و خاصیت جابه‌جایی ضرب ($a \cdot b = b \cdot a$) — با هم ترکیب می‌شوند تا پروتکلی را تولید کنند که نیمی از بشریت هر روز برای ارتباطات خصوصی خود به آن وابسته‌اند. هیچ یک از این سه قطعه به تنهایی خاص به نظر نمی‌رسد. آنچه تعیین‌کننده است، مونتاژ آن‌هاست.

از Diffie-Hellman تا پروتکل Signal

رمزنگاری سرتاسری که امروزه برنامه‌های پیام‌رسان حرفه‌ای از آن استفاده می‌کنند، تقریباً بدون استثنا، بر پایه نسخه‌ای ظریف و سخت‌گیرانه از تبادل Diffie-Hellman استوار است. پروتکل Signal که توسط Trevor Perrin و Moxie Marlinspike بین سال‌های ۲۰۱۳ تا ۲۰۱۶ طراحی شده، مرجع اصلی است. این پروتکل دو ایده کلیدی را با هم ترکیب می‌کند. اول، تبادل کلید در منحنی‌های بیضوی (X25519) که راز مشترک اولیه را بین دو دستگاه ایجاد می‌کند. دوم، آنچه Double Ratchet —چرخ‌دنده دوگانه— نامیده می‌شود که کلیدها را با هر پیام به طور خودکار نو می‌کند، به طوری که لو رفتن دستگاه در امروز، اجازه رمزگشایی پیام‌های گذشته و همچنین پیام‌های آینده را پس از چرخش چرخ‌دنده نمی‌دهد.

در زبان Zig، تبادل X25519 که راز مشترک بین دو دستگاه را ایجاد می‌کند، با استفاده از کتابخانه استاندارد، در شش سطر جای می‌گیرد:

```
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 //
reto_alicia = X25519.scalarmult(par_alicia.secret_key, par_bruno.public_key) catch unreachable
eto_bruno  = X25519.scalarmult(par_bruno.secret_key,  par_alicia.public_key) catch unreachable
                                                secreto_alicia == secreto_bruno  (32 bytes) //
```

آنچه در آن شش سطر اتفاق می‌افتد: کلیدهای عمومی به صورت آشکار منتقل می‌شوند. کلیدهای خصوصی هرگز از دستگاه مربوطه خارج نمی‌شوند. هر طرف، بر اساس کلید خصوصی خود و کلید عمومی طرف مقابل، همان راز سی و دو بایتی را استخراج می‌کند که هیچ‌کس در کانال نمی‌تواند آن را بازیابی کند. آن راز بعداً به عنوان بذر برای رمزنگاری پیام‌های مبادله شده عمل می‌کند. چرخ‌دنده دوگانه (Double Ratchet) در پروتکل Signal، یک چرخش مداوم به آن متریال اضافه می‌کند تا لو رفتن یک لحظه، بقیه گفتگو را به خطر نیندازد.

و دقیقاً در داخل `std.crypto.dh.X25519` چه چیزی وجود دارد؟ هیچ جادوی پنهانی نیست. آن‌ها دو تابع کوتاه هستند که می‌توان به طور کامل در خود کتابخانه استاندارد Zig خواند. اولی کلید عمومی را از کلید خصوصی استخراج می‌کند — همان «g^a» در تبادل:

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

به زبان مقاله: کلید خصوصی —در مفهوم بیضوی، نه در مفهوم حساب مقدماتی— در نقطه پایه منحنی Curve25519 «ضرب» می‌شود و نتیجه به صورت سی و دو بایت سریال‌سازی (serialized) می‌شود. عملیات `clampedMul` نسخه سخت‌گیرانه آن ضرب اسکالر است: این عملیات شامل محافظ‌هایی است که جامعه رمزنگاری طی سال‌ها برای مقاومت در برابر خانواده‌های شناخته شده حملات به آن اضافه کرده است. دو خط بدنه تابع.

تابع دوم کلید خصوصی شما را با کلید عمومی که طرف مقابل برای شما می‌فرستد ترکیب می‌کند. این همان «$(g^b)^a$» در تبادل است که راز مشترک سی و دو بایتی را تولید می‌کند که هیچ یک از شما هرگز آن را منتقل نکرده‌اید:

```
_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
                                                                            {
```

دو خط دیگر. کلید عمومی دریافت شده به عنوان نقطه‌ای روی منحنی تفسیر می‌شود و در کلید خصوصی خود شخص «ضرب» می‌شود. به دلیل خاصیت جابه‌جایی عملیات منحنی —مشابه خاصیت جابه‌جایی ضرب توان‌ها که در مثال عددی دیدیم— هر دو طرف در نهایت به همان نقطه سریال‌سازی شده می‌رسند: دقیقاً همان راز مشترکی که مقاله در مورد آن صحبت می‌کند.

همین و بس. آنچه در یک اپلیکیشن شبیه جادو به نظر می‌رسد، در واقعیت دو تابع با سه خط کد است. پیچیدگی فنی در یک عملیات واحد متمرکز شده است، `clampedMul`، که در همان کتابخانه استاندارد در پایین‌تر نوشته شده است، ده‌ها سال توسط جامعه بین‌المللی رمزنگاری مورد بازبینی قرار گرفته و در دسترس هر کسی است که بخواهد آن را حرف به حرف بخواند. هیچ جعبه سیاهی نه در اپلیکیشن ما و نه در کتابخانه استاندارد Zig وجود ندارد. کد منبع بازی وجود دارد که یک انسان می‌تواند آن را درک کند و سرعت ورود به آن را خودش انتخاب کند.

آنچه رمزنگاری سرتاسری از آن محافظت می‌کند

آنچه E2EE به خوبی از آن محافظت می‌کند، با فرض اجرای صحیح، محتوای پیام در حال انتقال است. یک سرور واسطه که داده‌های رمزنگاری شده را دریافت و ارسال می‌کند، توالی از بایت‌های غیرقابل فهم را خواهد دید. مهاجمی که به کابل، روتر یا نقطه دسترسی وای‌فای دسترسی دارد، همان را خواهد دید. ارائه‌دهنده سرویسی که کپی‌هایی از ترافیک را نگه می‌دارد، نمی‌تواند بعداً آن را بخواند. دولتی که به اپراتور سرویس دستور تحویل محتوا را می‌دهد، همان بایت‌های غیرقابل فهمی را دریافت خواهد کرد که سرور در ابتدا داشت.

این، در اصطلاحات کاربردی، بسیار زیاد است. این تفاوت بین نوشتن یک نامه در یک پاکت کدر و نوشتن آن روی یک کارت پستال است. هر دو می‌رسند. فقط یکی محتوا را در برابر نامه‌رسان حفظ می‌کند.

آنچه رمزنگاری سرتاسری از آن محافظت نمی‌کند

دانستن این موضوع نیز به همان اندازه ضروری است. E2EE از متاداده‌ها (فراداده‌ها) محافظت نمی‌کند: سرور همچنان می‌داند که کاربر الف داده‌هایی را به کاربر ب می‌فرستد، در چه ساعتی، با چه فرکانسی و از کجا، هرچند نمی‌داند چه می‌گوید. این متاداده‌ها، همانطور که قبلاً در *رمزنگاری به معنای خصوصی بودن نیست* استدلال کرده‌ایم، اغلب افشاگرانه‌تر از خود محتوا هستند. دانستن اینکه کسی با یک دفتر وکالت متخصص در طلاق در یک جمعه ساعت ۲۲:۰۰ به مدت سی دقیقه تماس گرفته است، داستانی را روایت می‌کند که محتوای تماس هرگز روایت نکرده است. این همان موقعیتی است که فردی را در حال ورود و خروج چندین باره از یک کلینیک سرطان‌شناسی ببینیم: نیازی به شنیدن هیچ یک از صحبت‌های داخل نیست تا تصور کنیم چه اتفاقی در حال رخ دادن است. یک متاداده تنها ممکن است معنایی نداشته باشد؛ اما چندین متاداده متقاطع، چیزی بسیار شبیه به حقیقت را ترسیم می‌کنند. E2EE از نقاط انتهایی محافظت نمی‌کند: اگر دستگاه گیرنده توسط یک برنامه مخرب آلوده شده باشد، پیام به طور معمول برای آن گیرنده رمزگشایی می‌شود و برنامه مخرب آن را می‌خواند. E2EE از هویت خود مخاطب محافظت نمی‌کند: اگر آلیس فکر کند که با برونو صحبت می‌کند اما مهاجمی در ابتدا خود را واسطه کرده باشد (یک *man in the middle*) و پروتکل شامل تأیید هویت مستقل نباشد، هر دو طرف در نهایت با فرد نفوذی صحبت می‌کنند در حالی که فکر می‌کنند با هم صحبت می‌کنند.

مورد چهارمی هم هست که شایسته است بدون ابهام بیان شود. E2EE مانع از آن نمی‌شود که ارائه‌دهنده‌ای که ادعای ارائه آن را دارد، علاوه بر این، کپی از پیام رمزنگاری نشده را در سیستم‌های خود نگه دارد. ادعای «پیام‌های من سرتاسری رمزنگاری شده‌اند» و ادعای «ارائه‌دهنده محتوای من را نگه نمی‌دارد» یکی نیستند. یک برنامه می‌تواند اولی را رعایت کند در حالی که دومی را نقض می‌کند؛ ما این را از سال ۲۰۱۸ بارها در عناوین اخبار دیده‌ایم. کاربر، مگر اینکه کد کلاینت قابل تأیید باشد، راه فنی برای تشخیص یک مورد از دیگری بدون تحقیق تخصصی ندارد. معروف‌ترین مورد در بین عموم مردم: واتس‌اپ پیام‌ها را در حال انتقال به صورت سرتاسری رمزنگاری می‌کند، اما اگر کاربر نسخه پشتیبان را در iCloud یا Google Drive بدون رمزنگاری اضافی فعال کند، آن کپی به صورت خوانا در زیرساخت یک شخص ثالث ذخیره می‌شود و رمزنگاری در نقطه انتهایی خود کاربر شکسته می‌شود.

سؤالی که اپراتور نمی‌خواهد بشنود

یک برنامه که ادعا می‌کند رمزنگاری سرتاسری دارد، از نظر فنی می‌تواند در مورد کلیدها یکی از این سه کار را انجام دهد:

1. **کلیدها فقط در دستگاه‌ها قرار دارند.** آن‌ها منحصراً در دستگاه‌های کاربران تولید شده و قرار می‌گیرند؛ اپراتور آن‌ها را نمی‌شناسد و ذخیره نمی‌کند. این حالت بهینه است.
2. **اپراتور اگر بخواهد می‌تواند دسترسی داشته باشد.** اپراتور کلیدهای کاربران را دارد (یا می‌تواند هر زمان بخواهد آن‌ها را تولید کند) و در پایگاه‌های داده خود ذخیره می‌کند. اگر بخواهد یا مجبور شود، می‌تواند محتوا را بخواند. این وضعیت اکثر سرویس‌های «ابری» است.
3. **اپراتور طبق طراحی نمی‌تواند دسترسی داشته باشد، اما دسترسی را کنترل می‌کند.** اپراتور کلیدها را ندارد، اما کنترل اپلیکیشنی را که آن‌ها را تولید می‌کند در دست دارد. اگر مجبور شود، می‌تواند یک به‌روزرسانی مخرب ارسال کند که کلیدها یا محتوا را قبل از رمزنگاری ضبط کند. این وضعیت بسیاری از سرویس‌های E2EE تجاری است.

بنابراین، سؤال عملی این نیست که آیا چیزی رمزگذاری شده است یا خیر، بلکه این است که چه کسی کنترل دستگاه و نرم‌افزاری را که کلیدها را مدیریت می‌کند، در دست دارد. در Solo2، کلیدها فقط در «صندوق» شما (IndexedDB رمزگذاری شده با رمز عبور شما) قرار دارند و نرم‌افزار کد منبع باز قابل تأیید است.

برای خواننده متخصص

رمزنگاری سرتاسری ابزاری برای حاکمیت دیجیتال است. اما مانند هر ابزار دیگری، کارایی آن به دستی که آن را نگه می‌دارد و زمینی که بر آن تکیه دارد، بستگی دارد.

1. کلیدهای رمزنگاری در کجا تولید می‌شوند و از نظر فیزیکی در کجا قرار دارند؟ اگر اپراتور بتواند به آن‌ها دسترسی داشته باشد (حتی به طور موقت، حتی تحت عنوان بازیابی)، E2EE فقط اسمی است.
2. آیا تأیید هویت مستقلی از مخاطب وجود دارد (شماره‌های امنیتی، کدهای QR، مقایسه خارج از باند) که از حمله مرد میانی در طول برقراری مکالمه جلوگیری کند؟
3. آیا کد کلاینت قابل حسابرسی است — باز، منتشر شده، قابل بازتولید — یا نیازمند اعتماد به حرف ارائه‌دهنده در مورد کاری است که کلاینت در واقع انجام می‌دهد؟
4. سرویس چه متاداده‌هایی را تولید و نگهداری می‌کند و برای چه مدت؟ حتی اگر محتوا مات باشد، متاداده‌ها می‌توانند بخش عمده‌ای از اطلاعات حساس را بازسازی کنند.

این چهار سؤال اطلاعات فنی پیشرفته‌ای نمی‌خواهند؛ آن‌ها اطلاعاتی را می‌خواهند که هر اپراتور صادقی می‌تواند در مستندات عمومی خود به آن‌ها پاسخ دهد. کیفیت و دقت پاسخ به اندازه خود پاسخ در مورد محصول حرف می‌زند.

رمزنگاری سرتاسری، اگر درست اجرا شود، یکی از ظریف‌ترین سازه‌هایی است که رمزنگاری معاصر به دنیای عمل آورده است. ایده اصلی —اینکه دو نفر بتوانند در یک کانال عمومی بر سر یک راز توافق کنند— متعلق به Whitfield Diffie و Martin Hellman در سال ۱۹۷۶ است؛ نیم قرن بعد ما همچنان در پیامدهای آن زندگی می‌کنیم. اما، مانند هر وعده فنی دیگری، ارزش آن به پایبندی واقعی بستگی دارد، نه به برچسب تبلیغاتی. سؤال یک متخصص صادق این نیست که «آیا رمزگذاری شده است؟»، بلکه این است که «کلیدها نزد کیست؟». پاسخ‌ها عواقب متفاوتی دارند. دانستن آن‌ها ضروری است.

منابع و مطالعه بیشتر

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*، IEEE Transactions on Information Theory، نوامبر ۱۹۷۶. مقاله بنیادی رمزنگاری کلید عمومی.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*، مشخصات عمومی Open Whisper Systems، بازبینی ۲۰۱۶. پایه پروتکل Signal و مشتقات صنعتی آن.
- IETF) RFC 7748 — Elliptic Curves for Security، ژانویه ۲۰۱۶). مشخصات هنجاری منحنی‌های X25519 و X448 که در تبادلات کلید مدرن استفاده می‌شوند.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley، ۲۰۱۰). فصل‌هایی در مورد تبادل کلید و پروتکل‌های رمزگذاری تأیید شده.
- مقررات (اتحادیه اروپا) 2024/1183 در مورد چارچوب هویت دیجیتال اروپایی (eIDAS 2) — چارچوب‌هایی را ایجاد می‌کند که در آن‌ها تأیید مستقل مخاطب پشتیبانی سازمانی پیدا می‌کند و تمایز بین رمزگذاری اسمی و واقعی پیامدهای حقوقی متفاوتی دارد.

← السابقKill switch و تصاحب نهادیالتالي → مدل کسب‌وکار به عنوان سیگنال اعتماد

قراءات حديثة

- تحلیل ۱۸۰ مه ۲۰۲۶ حریم خصوصی واقعی در مقابل ظاهری: سؤالاتی که باید از خود بپرسید
- تحلیل ۱۸۰ مه ۲۰۲۶ Self-hosting به عنوان یک فعالیت حرفه‌ای
- مفهوم ۱۸۰ مه ۲۰۲۶ ۲۴ کلمه: هویت رمزنگاری شده چیست

این مقاله را هر کجا که نیاز دارید همراه خود ببرید.

↓ مارک‌داون ↓ متن ساده ↓ PDF

فایل در دستگاه شما دانلود خواهد شد. از آنجا می‌توانید آن را ذخیره کنید، به Solo2 وارد کنید یا در هر کجا که می‌خواهید به اشتراک بگذارید. Cuadernos مقصد را برای شما تعیین نمی‌کند.

ختم شمعي · SHA-256 74dca832356c0370a3d8cea85ad618804f06828f74556a2281db601fd77c5648

Cuadernos Lacre · نشريّة من Menzuri Gestión S.L. ·
كتبها R.Eugenio · حرّرها فريق Solo2.