エンドツーエンド暗号化、その真の解説

プロバイダーがE2EEと言うときに語ること、そして語らないこと。広告的な飾りを排した、その仕組みと限界についての教育的解説。

はっきりさせておきましょう： WhatsApp はあなたのメッセージがエンドツーエンドで暗号化されていると言います。それは事実ですが、それだけでは十分ではありません。もしバックアップが追加の暗号化なしで iCloud や Google Drive に行くなら、暗号化はあなた自身の電話で破られます。実務的な問いは、暗号化されているかどうかではなく、鍵がどこにあるかです。

暗号化が真に意味するもの

メッセージを暗号化するとは、鍵と呼ばれる特定の情報を持たない者にとって、それをノイズのように見えるものに変換することを意味します。この操作は送信者のデバイスで行われ、正しい鍵を使えば、受信者のデバイスで元の状態に戻されます。その間、メッセージは明らかな意味を持たないバイトの連続として送られます。これがシンプルな考え方です。本記事の残りの部分では、場合によって、それが真の保証になるか、あるいは単なるマーケティング上のラベルになるかを分けるニュアンスについて解説します。

エンドツーエンド — 英語の *end-to-end*、略して E2EE — という形容詞は、ある種の正確さを加えます。暗号化は、中間のサーバーがそれを読み取って配信するために行われるのではありません。送信者のデバイスと受信者のデバイスという両端だけが鍵を持つように行われます。メッセージが通過するいかなるサーバーもノイズを見るだけで、メッセージを見ることはありません。これが、コンテンツがあるサーバーから次のサーバーへと暗号化されて移動するものの、通過する各サーバーが転送のために復号し、一時的に平文を復元する *中継時（in transit）* 暗号化との技術的な違いです。

共有秘密のパラドックス

明らかな問題があります。二人の人間が互いにメッセージを暗号化および復号できるようにするには、両者が同じ鍵を必要とします。しかし、お互いに送るすべてのものが、定義上、誰かが盗聴している可能性のあるチャネルを通過する場合、どうやってその鍵に合意するのでしょうか。後にそれを使用するのと同じチャネルで鍵を合意することは不可能に思えます。もし攻撃者が合意の際にそれを聞いてしまえば、その後のすべてを復号できてしまうからです。何十年もの間、古典的な暗号学はこれを力技で解決してきました。鍵は、使用を開始する前に、物理的な面会で直接手渡されていました。大使たちは、コートの裏地に縫い付けられた鍵のケースを持ち歩いていました。

現代の電子メールにおいて、その解決策はスケールしません。暗号化通信を行いたい相手全員の家へ物理的に行かなければならないとしたら、誰とも話すことができなくなるでしょう。50年前に暗号学コミュニティによって提示された問いはこれでした。「互いに面識がなく、公開チャネルのみを共有している二人が、その同じ公開チャネル上で、チャネルを聞いている誰もが知ることのできない秘密に合意することは可能か」。

Diffie-Hellmanの優雅さ

1976年、Whitfield Diffie と Martin Hellman という二人の数学者が、一見不可能と思えることを証明しました。公開チャネル（誰もが話していることをすべて聞くことができるチャネル）だけで話している二人が、いかなる傍聴者にも知られることなく秘密のパスワードに合意できるというものです。魔法のように聞こえますが、そうではありません。数学です。それ以来知られている Diffie-Hellman 鍵交換は、インターネットの事実上すべての暗号化通信の基礎となっており、半世紀にわたる集中的な使用と世界的な学術的精査がその堅牢さを裏付けています。視覚的な直感や数学を見たい方は、読み進めてください。それが機能することを信頼する方は、記事の文脈を失うことなく次に進むこともできます。

イメージで理解したい方のために、色を使った有名な比喩があります。アリスとボブが、彼らを聞いているイヴの目の前で、ベースとなる色（例えば黄色）に公開で合意すると想像してください。各自がプライベートで二つ目の秘密の色を選び、自分の秘密を黄色と混ぜます。アリスは特定のオレンジ色を得、ボブは特定の緑色を得ます。彼らはイヴの目の前でその結果を交換します。今度は各自が受け取った色と自分の秘密を混ぜ合わせます。混ぜる順番は関係ないので、二人は同じ最終的な色に到達します。イヴは黄色と二つの中間混合色を見ましたが、秘密の色は見ていません。秘密の色のいずれかがなければ、彼女は最終的な色に到達できません。実際の数学では、色をモジュラ群や楕円曲線におけるべき乗に置き換えますが、考え方は同じです。チャネル内の誰もが再構築することなく、共有秘密が公開の場で構築されるのです。

算術において、仕組みを見たい方のために： アリスは秘密の数字 a を選び、ボブは b を選びます。彼らはチャネル上で g^a と g^b を公開で交換します。アリスは $(g^b)^a$ を計算し、ボブは $(g^a)^b$ を計算します。二人は同じ g^{ab} に到達します。イヴは g、g^a、g^b がチャネルを通過するのを見ますが、g^a から a を復元すること（いわゆる離散対数問題）は、g が適切な数学的群から選ばれている場合、宇宙の年齢をはるかに超える天文学的な計算時間を必要とします。

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.

Diffie-HellmanからSignalプロトコルへ

今日のプロフェッショナルなメッセージング・アプリケーションが使用しているエンドツーエンド暗号化は、ほぼ例外なく、Diffie-Hellman 鍵交換のエレガントで強化されたバージョンに基づいています。2013年から2016年の間に Trevor Perrin と Moxie Marlinspike によって設計された Signal プロトコルがその参照基準です。二つの主要なアイデアを組み合わせています。第一は、二つのデバイス間で初期の共有秘密を生成する楕円曲線暗号（X25519）における鍵交換です。第二は、Double Ratchet（ダブル・ラチェット）と呼ばれるもので、メッセージごとに鍵を自動的に更新します。これにより、今日デバイスが侵害されたとしても、過去のメッセージを復号することはできず、ラチェットが回転した後の将来のメッセージも復号できません。

Zig では、二つのデバイス間で共有秘密を生成する X25519 鍵交換は、標準ライブラリを使用して6行に収まります。

```
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)
```

その6行で起こっていること： 公開鍵は公開されて移動します。秘密鍵はそれぞれのデバイスから決して出ません。各当事者は、自分の秘密鍵と相手の公開鍵から、チャネル内の誰も復元できない32バイトの同じ秘密を導出します。その秘密は後に、交換されるメッセージを暗号化するためのシードとして機能します。Signal プロトコルの Double Ratchet は、その材料を常に回転させることで、一瞬の侵害が残りの会話を侵害しないようにします。

そして、`std.crypto.dh.X25519` の中には正確に何があるのでしょうか？ 隠された魔法はありません。Zig 自身の標準ライブラリで完全に読むことができる2つの短い関数です。最初の関数は、秘密鍵から公開鍵 — 交換の「g^a」 — を導き出します:

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

この記事の言葉で言えば、秘密鍵は Curve25519 曲線のベースポイントによって「掛け合わされ」（初等算術的ではなく、楕円的な意味で）、その結果が32バイトにシリアライズされます。`clampedMul` 操作は、そのスカラー乗算を強化したバージョンです。暗号化コミュニティが既知の攻撃ファミリーに抵抗するために長年にわたって追加してきた保護措置が組み込まれています。関数本体は2行です。

2つ目の関数は、あなたの秘密鍵を相手が送ってきた公開鍵と組み合わせます。これは交換の「$(g^b)^a$」であり、二人とも一度も送信しなかった32バイトの共有秘密を生成します:

```
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();
}
```

さらに2行です。受信した公開鍵は曲線上の点として解釈され、自身の秘密鍵と「掛け合わせ」られます。曲線の操作の交換法則 — 数値の例で見た指数の乗算の交換法則と類似 — により、双方が同じシリアライズされた点に行き着きます。まさに記事が語る共有秘密です。

これで全部です。 アプリケーションの中で魔法のように見えるものは、実際にはそれぞれ3行の2つの関数にすぎません。技術的な複雑さは単一の操作 `clampedMul` に集中しています。この操作は同じ標準ライブラリのさらに下に書かれており、国際的な暗号化コミュニティによって何十年にもわたって見直され、一字一句読みたいと願う人なら誰でもアクセスできるようになっています。私たちのアプリケーションにも、Zig の標準ライブラリにも、ブラックボックスはありません。人間が理解できるオープンソースコードがあり、それをどの程度のペースで深く掘り下げるかは自由に選ぶことができます。

エンドツーエンド暗号化が守るもの

E2EEが正しく実装されていると仮定して、それが十分に保護するのは中継中のメッセージの内容です。暗号化されたデータを受信して転送する中間サーバーには、理解不能なバイトの連続が見えます。ケーブル、ルーター、Wi-Fiアクセスポイントにアクセスできる攻撃者も同じものを見ます。通信のコピーを保存しているサービス・プロバイダーも、後からそれを読み取ることはできません。サービス運営者にコンテンツの提出を命じる政府も、サーバーが最初に持っていたのと同じ理解不能なバイトを受け取ることになります。

これは実務的な観点から見て、非常に大きなことです。不透明な封筒の中に手紙を書くのと、ハガキに書くのとの違いです。どちらも届きますが、郵便配達人に対して内容を保護できるのは一方だけです。

エンドツーエンド暗号化が守らないもの

同様に知っておくべきことがあります。E2EEはメタデータを保護しません。サーバーは、ユーザーAがユーザーBに何時に、どのくらいの頻度で、どこからデータを送っているかを、何を言っているかは知らなくても、依然として知っています。これらのメタデータは、以前に *「暗号化する」ことは「プライバシーを守る」ことではない* で論じたように、コンテンツそのものよりも多くのことを明らかにすることがよくあります。金曜日の22:00に離婚専門の弁護士事務所に30分間電話したという事実を知ることは、通話の内容が決して語らなかったストーリーを物語ります。これは、ある人が腫瘍内科のクリニックを数回出入りするのを見るのと同じ状況です。中で何を話しているかを聞かなくても、何が起こっているかを想像するのに十分です。単一の孤立したメタデータは何の意味も持たないかもしれませんが、複数のデータが交差することで、真実にあまりにも近い何かが描き出されます。E2EEはエンドポイントを保護しません。もし受信者のデバイスが悪意のあるプログラムに侵害されていれば、メッセージはその受信者に対して通常通り復号され、悪意のあるプログラムがそれを読み取ります。E2EEは対話者自身のアイデンティティを保護しません。もしアリスがボブと話していると信じていても、攻撃者が最初に介在していた場合（*中間者（man-in-the-middle）攻撃*）、そしてプロトコルに独立した検証が含まれていない場合、両者は互いに話していると思い込みながら侵入者と話すことになります。

曖昧さなく定式化しておくべき4つ目のことがあります。E2EEは、それを提供していると主張するプロバイダーが、さらに暗号化されていないメッセージのコピーを自社のシステムに保存することを防ぐものではありません。「私のメッセージはエンドツーエンドで暗号化されている」という主張と「プロバイダーは私のコンテンツを保存していない」という主張は同じではありません。アプリケーションは、2番目の主張に違反しながら最初の主張を満たすことができます。2018年以来、ニュースの見出しでこれを繰り返し目にしてきました。ユーザーは、クライアントのコードが検証可能でない限り、専門家の調査なしに一方のケースと他方を技術的に区別する方法がありません。一般大衆に最もよく知られている例：WhatsAppは中継中にメッセージをエンドツーエンドで暗号化しますが、ユーザーが追加の暗号化なしに iCloud や Google ドライブでバックアップを有効にすると、そのコピーは第三者のインフラストラクチャに読み取り可能な状態で保存され、ユーザー自身の端で暗号化が破られます。

オペレーターが聞きたくない質問

エンドツーエンドで暗号化すると主張するアプリケーションは、鍵に関して技術的に3つのことのいずれかを行うことができます。

1. **鍵はデバイスにのみ存在する。** ユーザーのデバイス上でのみ生成され、そこにのみ存在します。運営者は鍵を知らず、保存もしていません。これが最適なケースです。
2. **運営者は望めばアクセスできる。**運営者がユーザーの鍵を保持している（あるいは自由に生成できる）状態で、データベースに保存しています。運営者が望むか、あるいは強制されれば、内容を読むことができます。ほとんどの「クラウド」サービスがこのケースに該当します。
3. **設計上はアクセスできないが、アクセス権を制御している。**運営者は鍵を持っていませんが、鍵を生成するアプリ自体を制御しています。強制されれば、暗号化される前に鍵や内容を奪取する悪意のあるアップデートを送信することが可能です。多くの商用E2EEサービスがこのケースに該当します。

したがって、実務的な問いは「暗号化されているか」ではなく、「デバイスと鍵を管理するソフトウェアを誰が制御しているか」です。Solo2では、鍵はあなたの「Bóveda（ボベダ：金庫）」（あなたのパスワードで暗号化されたIndexedDB）にのみ存在し、ソフトウェアは検証可能なオープンソースコードです。

プロフェッショナルな読者のために

エンドツーエンド暗号化は、デジタル主権のための道具です。しかし、あらゆる道具と同様に、その有効性はそれを使う手と、それを支える土壌にかかっています。

1. 暗号鍵はどこで生成され、物理的にどこに存在していますか？もし運営者がそれらにアクセスできるなら（たとえ一時的であれ、リカバリの名目であれ）、E2EEは名ばかりのものです。
2. 会話の確立時に中間者攻撃を防ぐための、対話者の独立した検証（セキュリティ番号、QRコード、アウトオブバンド比較）は存在しますか？
3. クライアントのコードは監査可能（公開され、出版され、再現可能）ですか、それともクライアントが実際に何をしているかについてプロバイダーの言葉を信用する必要がありますか？
4. サービスはどのようなメタデータを生成および保持し、それはどのくらいの期間ですか？たとえ内容が不透明であっても、メタデータは機密情報の大部分を再構築することができます。

これら4つの質問は、高度な技術情報を求めているわけではありません。誠実な運営者なら公開ドキュメントで答えられる情報を求めています。回答の質と正確さは、回答そのものと同じくらい製品について多くを語ります。

エンドツーエンド暗号化は、正しく実装されれば、現代の暗号学が日常の実践にもたらした最も洗練された構造の一つです。「二人が公開チャネル上で秘密に合意できる」という独創的なアイデアは、1976年のWhitfield DiffieとMartin Hellmanによるものです。半世紀を経た今も、私たちはその恩恵の中に生きています。しかし、いかなる技術的約束もそうであるように、その価値はラベルではなく実際の履行にかかっています。誠実な専門家の問いは「暗号化されているか」ではなく、「誰が鍵を持っているか」です。その答えによって、もたらされる結果は全く異なります。それを知っておくことは重要です。

参考文献および関連資料

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, 1976年11月。公開鍵暗号の基礎となる論文。
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, Open Whisper Systems による公開仕様、2016年改訂。Signal プロトコルおよびその産業用派生物の基礎。
- RFC 7748 — Elliptic Curves for Security (IETF, 2016年1月)。現代の鍵交換で使用される X25519 および X448 曲線の規範的仕様。
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010)。鍵交換と認証付き暗号プロトコルに関する章。
- 欧州デジタルアイデンティティ枠組みに関する規則 (EU) 2024/1183 (eIDAS 2) — 対話者の独立した検証が制度的支援を得る枠組みを確立し、名目上の暗号化と実際の暗号化の区別が異なる法的結果をもたらす場所。

← 前へキルスイッチと制度的キャプチャ次へ →信頼のシグナルとしてのビジネスモデル

最近の記事

- 分析 · 2026年5月18日 真のプライバシー vs 表向きのプライバシー：問い直すべきこと
- 分析 · 2026年5月18日 専門的実践としてのセルフホスティング
- コンセプト · 2026年5月18日 24個の単語：暗号学的アイデンティティとは何か

この記事をダウンロードして、必要な場所で活用してください。

↓ Markdown ↓ テキスト形式 ↓ PDF

ファイルはお使いのデバイスにダウンロードされます。そこから保存、Solo2 へのインポート、または任意の場所での共有が可能です。Cuadernos が送信先を決定することはありません。

封蝋 · SHA-256 2727b1de4bee50288b048580731cbf089703a8f97d753c5127fb29e612be8f4c

Cuadernos Lacre · Menzuri Gestión S.L. による刊行物 ·
著者：R.Eugenio · 編集：Solo2 チーム

当サイトはクッキーを使用せず、第三者のリソースも読み込みません。自社運用の匿名訪問者カウンター（欧州サーバー上のUmami）を使用し、ライト/ダークテーマの切り替えに必要な最小限の JavaScript のみで動作します。トラッカー、プロファイリング、データ共有は一切ありません。更新情報を受け取るには：RSS。