Ende-zu-Ende-Verschlüsselung, wirklich erklärt

Was Anbieter sagen, wenn sie E2EE sagen, und was sie verschweigen. Eine didaktische Erklärung des Mechanismus und seiner Grenzen, ohne Werbehülle.

Damit wir uns verstehen: WhatsApp sagt, dass Ihre Nachrichten Ende-zu-Ende verschlüsselt sind. Das ist wahr — und nicht genug. Wenn das Backup ohne zusätzliche Verschlüsselung in iCloud oder Google Drive geht, wird die Verschlüsselung auf Ihrem eigenen Telefon gebrochen. Die operative Frage ist nicht, ob es verschlüsselt ist, sondern wo die Schlüssel liegen.

Was Verschlüsselung wirklich bedeutet

Eine Nachricht zu verschlüsseln bedeutet, sie in etwas zu verwandeln, das für jeden, der nicht über eine bestimmte Information, den sogenannten Schlüssel, verfügt, wie Rauschen aussieht. Der Vorgang erfolgt auf dem Gerät des Absenders und wird mit dem richtigen Schlüssel auf dem Gerät des Empfängers rückgängig gemacht. Dazwischen wandert die Nachricht als eine Folge von Bytes ohne offensichtliche Bedeutung. Das ist die einfache Idee. Der Rest des Artikels befasst sich mit den Nuancen, die sie je nach Fall zu einer echten Garantie oder zu einem Marketing-Etikett machen.

Das Adjektiv *Ende-zu-Ende* — auf Englisch *end-to-end*, abgekürzt E2EE — fügt eine Präzisierung hinzu. Die Verschlüsselung erfolgt nicht, damit ein Zwischenserver sie lesen und zustellen kann. Sie erfolgt so, dass nur die beiden Enden — das Gerät des Absenders und das Gerät des Empfängers — den Schlüssel besitzen. Jeder Server, den die Nachricht passiert, sieht das Rauschen, nicht die Nachricht. Dies ist der technische Unterschied zur Verschlüsselung *beim Transport*, bei der der Inhalt verschlüsselt von einem Server zum nächsten wandert, aber jeder Server, den er passiert, ihn entschlüsselt, um ihn weiterzuleiten, wodurch der Klartext vorübergehend wiederhergestellt wird.

Das Paradoxon des gemeinsamen Geheimnisses

Es gibt ein offensichtliches Problem. Damit zwei Personen Nachrichten untereinander verschlüsseln und entschlüsseln können, benötigen beide denselben Schlüssel. Aber wie einigt man sich auf diesen Schlüssel, wenn alles, was man sich schickt, per Definition über einen Kanal läuft, auf dem jemand mithören könnte? Sich auf den Schlüssel in demselben Kanal zu einigen, in dem man ihn später verwenden wird, scheint unmöglich: Wenn der Angreifer ihn bei der Einigung hört, kann er alles Folgende entschlüsseln. Jahrzehntelang löste die klassische Kryptografie dies auf die harte Tour: Die Schlüssel wurden vor Beginn der Nutzung bei physischen Treffen persönlich übergeben. Botschafter trugen Koffer mit Schlüsseln bei sich, die in das Futter ihres Mantels eingenäht waren.

In der heutigen E-Mail-Kommunikation ist diese Lösung nicht skalierbar. Wenn wir physisch zu jedem nach Hause gehen müssten, mit dem wir verschlüsselt kommunizieren wollen, würden wir mit niemandem ins Gespräch kommen. Die Frage, die sich die kryptografische Gemeinschaft vor fünfzig Jahren stellte, lautete: Ist es möglich, dass zwei Personen, die sich nicht kennen und nur einen öffentlichen Kanal teilen, in eben diesem öffentlichen Kanal ein Geheimnis vereinbaren, das niemand kennen kann, der den Kanal abhört?

Die Eleganz von Diffie-Hellman

Im Jahr 1976 demonstrierten zwei Mathematiker namens Whitfield Diffie und Martin Hellman etwas scheinbar Unmögliches: dass zwei Personen, die nur über einen öffentlichen Kanal sprechen — einen Kanal, auf dem jeder alles hören kann, was sie sagen —, sich auf ein geheimes Passwort einigen können, ohne dass ein Zuhörer es entdecken kann. Es klingt wie Magie. Ist es aber nicht: Es ist Mathematik. Der Diffie-Hellman-Schlüsselaustausch, wie er seither genannt wird, ist die Grundlage für praktisch die gesamte verschlüsselte Internetkommunikation, und ein halbes Jahrhundert intensiver Nutzung und weltweiter akademischer Prüfung belegen seine Solidität. Wer die visuelle Intuition oder die Mathematik sehen möchte, kann weiterlesen. Wer lieber darauf vertrauen möchte, dass es funktioniert, kann ebenfalls fortfahren, ohne den Faden des Artikels zu verlieren.

Für diejenigen, die es sich bildlich vorstellen wollen, gibt es eine bekannte Analogie mit Farben. Stellen Sie sich vor, dass sich Alice und Bruno vor den Augen von Eva, die sie belauscht, öffentlich auf eine Grundfarbe einigen — sagen wir Gelb. Jeder wählt privat eine zweite Geheimfarbe und mischt sein Geheimnis mit dem Gelb. Alice erhält ein bestimmtes Orange; Bruno erhält ein bestimmtes Grün. Sie tauschen die Ergebnisse vor den Augen von Eva aus. Nun mischt jeder die erhaltene Farbe mit seinem eigenen Geheimnis, und beide gelangen zur gleichen Endfarbe, da die Reihenfolge der Mischungen keine Rolle spielt. Eva hat das Gelb und die beiden Zwischenmischungen gesehen, aber nicht die Geheimnisse; ohne eines der Geheimnisse kann sie nicht zur Endfarbe gelangen. Die reale Mathematik ersetzt die Farben durch Exponentiationen in modularen Gruppen oder elliptischen Kurven, aber die Idee ist die gleiche: Das gemeinsame Geheimnis wird in aller Öffentlichkeit aufgebaut, ohne dass jemand im Kanal es rekonstruieren kann.

In der Arithmetik, für diejenigen, die den Mechanismus lieber sehen wollen: Alice wählt eine Geheimzahl a, Bruno wählt b. Sie tauschen g^a und g^b offen über den Kanal aus. Alice berechnet $(g^b)^a$ und Bruno berechnet $(g^a)^b$; beide gelangen zum gleichen g^{ab}. Eva sieht g, g^a und g^b über

den Kanal wandern, aber a aus g^a wiederherzustellen — das sogenannte Problem des diskreten Logarithmus — erfordert eine astronomische Rechenzeit, die das Alter des Universums bei weitem übersteigt, wenn g in einer geeigneten mathematischen Gruppe gewählt wird.

Für diejenigen, die es mit kleinen Zahlen überprüfen wollen. Der Diffie-Hellman-Austausch kann vollständig mit Zahlen durchgespielt werden, die klein genug sind, um die Berechnungen von Hand durchzuführen. Wer sich nicht mit Arithmetik befassen möchte, kann diesen Block überspringen, ohne den Faden des Artikels zu verlieren; wer den Mechanismus Schritt für Schritt in Aktion sehen möchte, findet ihn hier. **Die öffentlichen Regeln**, die jeder lesen kann: eine Primzahl $p = 11$ (beim echten Diffie-Hellman hat sie etwa dreihundert Stellen; wir verwenden elf, damit die Rechnungen auf eine Seite passen), eine Basis $g = 2$, und die Konvention, dass die gesamte Arithmetik *modulo p* durchgeführt wird — man rechnet, teilt durch p und behält den Rest, wie eine Uhr mit elf Positionen, die nach der Zehn wieder auf Null springt. **Die privaten Wahlen**, je eine für jeden und niemals geteilt: Alice wählt $a = 4$. Bruno wählt $b = 7$.

Schritt 1. Alice berechnet $2^4 = 16$, dann 16 mod 11 = **5**. Sie sendet die Fünf. Eva notiert sie sich.

Schritt 2. Bruno berechnet $2^7 = 128$, dann 128 mod 11 = **7**. Er sendet die Sieben. Eva notiert sie sich ebenfalls. Nach den beiden Übertragungen enthält Evas Notizbuch vier Daten: $p = 11$, $g = 2$, A = 5, B = 7. Ihr fehlt die gemeinsame Zahl, die Alice und Bruno gleich ableiten werden — und die Eva nicht rekonstruieren können wird.

Schritt 3. Alice nimmt die Sieben, die Bruno ihr geschickt hat, und potenziert sie mit ihrem privaten Exponenten $a = 4$. Um nicht mit $7^4 = 2401$ hantieren zu müssen, wird stückweise gerechnet, indem bei jedem Schritt Modulo angewendet wird:

$7^2 = 49$

49 mod 11 = 5

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

25 mod 11 = 3

Alice erhält die Zahl **3**.

Schritt 4. Bruno nimmt die Fünf, die Alice ihm geschickt hat, und potenziert sie mit seinem privaten Exponenten $b = 7$. Wieder stückweise:

$5^2 = 25$ 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

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

Bruno erhält ebenfalls **3**.

Beide sind parallel arbeitend zur gleichen Zahl, 3, gelangt. Keiner von beiden hat zu irgendeinem Zeitpunkt seinen privaten Exponenten gesendet. Alice weiß nicht, dass $b = 7$ ist; Bruno weiß nicht, dass $a = 4$ ist. Jeder nutzte den öffentlichen Wert, den der andere gesendet hat, in Kombination mit seinem eigenen privaten Exponenten, und sie trafen sich am selben Ziel. **Warum kommen sie zur gleichen Zahl?** Was jeder berechnet hat: Alice, $(g^b)^a = 2^{7\times4} = 2^{28}$ mod 11. Bruno, $(g^a)^b = 2^{4\times7} = 2^{28}$ mod 11. Es ist dieselbe Menge, weil die Reihenfolge bei der Multiplikation der Exponenten keine Rolle spielt ($7 \times 4 = 4 \times 7$). Jeder ist auf einem anderen Weg zum selben Ziel gelangt.

Und was ist mit Eva? Sie hat in ihrem Notizbuch $p = 11$, $g = 2$, A = 5, B = 7, und hätte gerne die 3. Um sie zu berechnen, müsste sie a oder b kennen — aber keines von beiden ist über den Kanal gereist. Ihr einziger Weg ist, sich zu fragen: «Für welchen Exponenten a gilt 2^a mod 11 = 5?». Bei einem so kleinen p kann sie 0, 1, 2, 3, 4... ausprobieren und es in weniger als einer Minute finden. Ersetzt man die 11 jedoch durch eine Primzahl mit dreihundert Stellen, hat der Raum der möglichen Exponenten mehr Elemente als es Atome im beobachtbaren Universum gibt. **Bis heute gibt es keinen der Menschheit bekannten Algorithmus, der diesen Raum in weniger als Milliarden von Jahren durchsuchen könnte.** Dies ist das sogenannte *Problem des diskreten Logarithmus*: vorwärts einfach, rückwärts rechnerisch unmöglich. Und das ist der Grund, warum die Verschlüsselung standhält, selbst wenn Eva das gesamte Gespräch Buchstabe für Buchstabe verfolgt hat.

Drei einfache Zutaten — Uhren-Arithmetik, Exponentiation und die Kommutativität der Multiplikation ($a \cdot b = b \cdot a$) — ergeben in Kombination ein Protokoll, auf das die halbe Menschheit jeden Tag für ihre private Kommunikation angewiesen ist. Keine der drei Komponenten für sich allein erscheint besonders. Entscheidend ist der Zusammenbau.

Von Diffie-Hellman zum Signal-Protokoll

Die Ende-zu-Ende-Verschlüsselung, die heute in professionellen Messaging-Apps verwendet wird, beruht fast ausnahmslos auf einer eleganten und gehärteten Version des Diffie-Hellman-Austauschs. Das Signal-Protokoll, das von Trevor Perrin und Moxie Marlinspike zwischen 2013 und 2016 entwickelt wurde, ist die Referenz. Es kombiniert zwei Schlüsselideen. Die erste ist der Schlüsselaustausch in elliptischen Kurven (X25519), der das ursprüngliche gemeinsame Geheimnis zwischen zwei Geräten erzeugt. Die zweite ist das sogenannte Double Ratchet — das Doppelgetriebe —, das die Schlüssel automatisch mit jeder Nachricht erneuert, so dass die Kompromittierung des Geräts heute keine Entschlüsselung vergangener Nachrichten ermöglicht, und auch keine zukünftiger Nachrichten, sobald das Getriebe gedreht wurde.

In Zig passt der X25519-Austausch, der das gemeinsame Geheimnis zwischen zwei Geräten erzeugt, unter Verwendung der Standardbibliothek in sechs Zeilen:

```
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)
```

Was in diesen sechs Zeilen passiert: Die öffentlichen Schlüssel wandern offen. Die privaten Schlüssel verlassen niemals das jeweilige Gerät. Jede Seite leitet aus ihrem privaten und dem öffentlichen Schlüssel der anderen Seite das gleiche zweiunddreißig Byte lange Geheimnis ab, das niemand im Kanal wiederherstellen kann. Dieses Geheimnis dient später als Keim für die Verschlüsselung der ausgetauschten Nachrichten. Das Double Ratchet des Signal-Protokolls fügt eine ständige Rotation dieses Materials hinzu, so dass die Kompromittierung eines Augenblicks nicht den Rest des Gesprächs gefährdet.

Und was genau befindet sich in `std.crypto.dh.X25519`? Keine verborgene Magie. Es sind zwei kurze Funktionen, die man vollständig in Zigs eigener Standardbibliothek nachlesen kann. Die erste leitet den öffentlichen Schlüssel aus dem privaten ab — das «g^a» des Austauschs:

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

In der Sprache des Artikels: Der private Schlüssel wird mit dem Basispunkt der Curve25519-Kurve «multipliziert» — im elliptischen, nicht im elementar-arithmetischen Sinne — und das Ergebnis wird in zweiunddreißig Bytes serialisiert. Die Operation `clampedMul` ist die gehärtete Version dieser skalaren Multiplikation: Sie integriert die Schutzmaßnahmen, die die kryptografische Gemeinschaft im Laufe der Jahre hinzugefügt hat, um bekannten Angriffsfamilien zu widerstehen. Zwei Zeilen Funktionskörper.

Die zweite Funktion kombiniert Ihren privaten Schlüssel mit dem öffentlichen Schlüssel, den Ihnen die andere Partei sendet. Es ist das «$(g^b)^a$» des Austauschs, das das zweiunddreißig Byte lange gemeinsame Geheimnis erzeugt, das keiner von Ihnen jemals übertragen hat:

```
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();
}
```

Zwei weitere Zeilen. Der empfangene öffentliche Schlüssel wird als Punkt auf der Kurve interpretiert und mit dem eigenen privaten Schlüssel «multipliziert». Aufgrund der Kommutativität der Kurvenoperation — analog zur Kommutativität der Exponenten-Multiplikation, die wir im numerischen Beispiel gesehen haben — enden beide Parteien mit demselben serialisierten Punkt: genau dem gemeinsamen Geheimnis, von dem der Artikel spricht.

Das ist alles. Was in einer Anwendung wie Magie aussieht, sind in Wirklichkeit zwei Funktionen von jeweils drei Zeilen. Die technische Komplexität konzentriert sich auf eine einzige Operation, `clampedMul`, die weiter unten in derselben Standardbibliothek steht, seit Jahrzehnten von der internationalen kryptografischen Gemeinschaft überprüft wird und für jeden zugänglich ist, der sie Buchstabe für Buchstabe lesen möchte. Es gibt keine Blackbox, weder in unserer Anwendung noch in Zigs Standardbibliothek. Es gibt Open Source Code, den ein Mensch verstehen kann, wobei er das Tempo wählt, mit dem er sich darauf einlassen möchte.

Was Ende-zu-Ende-Verschlüsselung schützt

Was E2EE gut schützt, eine korrekte Implementierung vorausgesetzt, ist der Inhalt der Nachricht während des Transports. Ein Zwischenserver, der die verschlüsselten Daten empfängt und weiterleitet, sieht eine Folge unverständlicher Bytes. Ein Angreifer mit Zugriff auf das Kabel, den Router, den WLAN-Zugangspunkt sieht dasselbe. Ein Dienstanbieter, der Kopien des Datenverkehrs aufbewahrt, kann diesen nachträglich nicht lesen. Eine Regierung, die den Dienstbetreiber anweist, den Inhalt herauszugeben, erhält dieselben unverständlichen Bytes, die der Server ursprünglich hatte.

Das ist in praktischer Hinsicht viel. Es ist der Unterschied zwischen dem Schreiben eines Briefes in einem undurchsichtigen Umschlag und dem Schreiben auf einer Postkarte. Beide kommen an. Nur einer bewahrt den Inhalt vor dem Postboten.

Was Ende-zu-Ende-Verschlüsselung nicht schützt

Man sollte es genauso gut wissen. E2EE schützt keine Metadaten: Der Server weiß immer noch, dass Benutzer A Daten an Benutzer B sendet, zu welcher Zeit, mit welcher Häufigkeit und von wo aus, auch wenn er nicht weiß, was gesagt wird. Diese Metadaten, wie wir bereits in *Verschlüsseln bedeutet nicht privat zu sein* dargelegt haben, sind oft aussagekräftiger als der Inhalt. Zu wissen, dass jemand an einem Freitag um 22:00 Uhr dreißig Minuten lang eine auf Scheidungen spezialisierte Anwaltskanzlei angerufen hat, erzählt eine Geschichte, die der Inhalt des Anrufs nie erzählt hat. Es ist dieselbe Situation wie die Beobachtung einer Person, die mehrmals eine Onkologie-Klinik betritt und verlässt: Man muss nichts von dem hören, was drinnen gesprochen wird, um sich vorzustellen, was passiert. Ein einzelnes isoliertes Metadatum mag nichts bedeuten; mehrere miteinander verknüpfte zeichnen ein Bild, das der Wahrheit allzu nahe kommt. E2EE schützt die Endpunkte nicht: Wenn das Gerät des Empfängers durch ein Schadprogramm kompromittiert ist, wird die Nachricht für diesen Empfänger normal entschlüsselt und das Schadprogramm liest sie. E2EE schützt nicht vor der Identität des Gesprächspartners an sich: Wenn Alice glaubt, mit Bruno zu sprechen, sich

aber ein Angreifer am Anfang eingeschaltet hat (ein *Man-in-the-Middle*) und das Protokoll keine unabhängige Verifizierung beinhaltet, sprechen am Ende beide Parteien mit dem Eindringling und glauben, sie sprächen miteinander.

Es gibt eine vierte Sache, die unmissverständlich formuliert werden sollte. E2EE verhindert nicht, dass ein Anbieter, der behauptet, es anzubieten, zusätzlich eine Kopie der unverschlüsselten Nachricht in seinen eigenen Systemen speichert. Die Behauptung „meine Nachrichten sind Ende-zu-Ende verschlüsselt“ und die Behauptung „der Anbieter speichert meinen Inhalt nicht“ sind nicht dasselbe. Eine Anwendung kann die erste Behauptung erfüllen, während sie die zweite verletzt; wir haben dies seit 2018 wiederholt in Schlagzeilen der Presse gesehen. Der Benutzer hat, sofern der Code des Clients nicht verifizierbar ist, keine technische Möglichkeit, den einen Fall ohne Expertenuntersuchung vom anderen zu unterscheiden. Der bekannteste Fall in der breiten Öffentlichkeit: WhatsApp verschlüsselt Nachrichten beim Transport Ende-zu-Ende, aber wenn der Benutzer das Backup in iCloud oder Google Drive ohne zusätzliche Verschlüsselung aktiviert, wird diese Kopie lesbar in der Infrastruktur eines Dritten gespeichert, und die Verschlüsselung wird am Ende des Benutzers selbst aufgebrochen.

Die Frage, die der Betreiber nicht hören will

Eine Anwendung, die behauptet, Ende-zu-Ende zu verschlüsseln, kann technisch gesehen eine von drei Dingen in Bezug auf die Schlüssel tun:

1. **Die Schlüssel befinden sich nur auf den Geräten.** Sie werden ausschließlich auf den Geräten der Benutzer generiert und befinden sich dort; der Betreiber kennt und speichert sie nicht. Dies ist der optimale Fall.
2. **Der Betreiber kann zugreifen, wenn er will.** Der Betreiber besitzt die Schlüssel der Nutzer (oder kann sie nach Belieben generieren) und speichert sie in seinen Datenbanken. Wenn er will oder dazu gezwungen wird, kann er den Inhalt lesen. Dies ist bei den meisten „Cloud“-Diensten der Fall.
3. **Der Betreiber kann bauartbedingt nicht zugreifen, kontrolliert aber den Zugriff.** Der Betreiber hat die Schlüssel nicht, hat aber die Kontrolle über die Anwendung, die sie generiert. Wenn er dazu gezwungen wird, kann er ein bösartiges Update senden, das die Schlüssel oder den Inhalt vor der Verschlüsselung abfängt. Dies ist bei vielen kommerziellen E2EE-Diensten der Fall.

Die operative Frage lautet daher nicht, ob etwas verschlüsselt ist, sondern wer die Kontrolle über das Gerät und die Software hat, die die Schlüssel verwaltet. Bei Solo2 befinden sich die Schlüssel ausschließlich in Ihrem Tresor (mit Ihrem Passwort verschlüsselte IndexedDB) und die Software ist verifizierbarer Open Source Code.

Für den professionellen Leser

Ende-zu-Ende-Verschlüsselung ist ein Werkzeug für digitale Souveränität. Aber wie jedes Werkzeug hängt seine Wirksamkeit von der Hand ab, die es führt, und dem Boden, auf dem es steht.

1. Wo werden die kryptografischen Schlüssel generiert und wo befinden sie sich physisch? Wenn der Betreiber auf sie zugreifen kann (auch nur vorübergehend, auch unter dem Vorwand der Wiederherstellung), ist E2EE nur nominell.
2. Gibt es eine unabhängige Verifizierung des Gesprächspartners (Sicherheitsnummern, QR-Codes, Out-of-Band-Vergleich), die einen Man-in-the-Middle-Angriff beim Aufbau des Gesprächs verhindert?
3. Ist der Code des Clients überprüfbar — offen, veröffentlicht, reproduzierbar — oder verlangt er, dass man sich auf das Wort des Anbieters verlässt, was der Client tatsächlich tut?
4. Welche Metadaten generiert und speichert der Dienst, und wie lange? Selbst wenn der Inhalt undurchsichtig ist, können Metadaten einen Großteil der sensiblen Informationen rekonstruieren.

Diese vier Fragen verlangen keine fortgeschrittenen technischen Informationen; sie verlangen nach Informationen, die jeder ehrliche Betreiber in seiner öffentlichen Dokumentation beantworten kann. Die Qualität und Präzision der Antwort sagt ebenso viel über das Produkt aus wie die Antwort selbst.

Ende-zu-Ende-Verschlüsselung, richtig gemacht, ist eine der feinsten Konstruktionen, die die zeitgenössische Kryptografie für die tägliche Praxis geliefert hat. Die ursprüngliche Idee — zwei Personen können sich über einen öffentlichen Kanal auf ein Geheimnis einigen — stammt von Whitfield Diffie und Martin Hellman, 1976; ein halbes Jahrhundert später leben wir immer noch in deren Konsequenz. Aber wie bei jedem technischen Versprechen hängt sein Wert von der tatsächlichen Einhaltung ab, nicht vom Etikett. Die Frage des ehrlichen Fachmanns lautet nicht „Ist es verschlüsselt?“, sondern „Wer hat die Schlüssel?“. Die Antworten haben unterschiedliche Konsequenzen. Man sollte sie kennen.

Quellen und weiterführende Literatur

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, November 1976. Grundlegender Artikel zur Public-Key-Kryptografie.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, öffentliche Spezifikation von Open Whisper Systems, Revision 2016. Basis des Signal-Protokolls und seiner industriellen Ableitungen.
- RFC 7748 — Elliptic Curves for Security (IETF, Januar 2016). Normative Spezifikation der Kurven X25519 und X448, die in modernen Schlüsselaustauschen verwendet werden.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Kapitel über Schlüsselaustausch und authentifizierte Verschlüsselungsprotokolle.
- Verordnung (EU) 2024/1183 über einen Rahmen für eine europäische digitale Identität (eIDAS 2) — etabliert Rahmenbedingungen, in denen die unabhängige Verifizierung des Gesprächspartners institutionelle Unterstützung erhält und in denen die Unterscheidung zwischen nomineller und echter Verschlüsselung unterschiedliche rechtliche Konsequenzen hat.

← ZurückKill Switch und institutionelle VereinnahmungWeiter →Das Geschäftsmodell als Vertrauenssignal

Aktuelle Lektüre

- Analyse · 18. Mai 2026 Echte vs. scheinbare Privatsphäre: Die Fragen, die man sich stellen sollte
- Analyse · 18. Mai 2026 Self-Hosting als berufliche Praxis
- Konzept · 18. Mai 2026 Die 24 Wörter: Was eine kryptografische Identität ist

Nehmen Sie diesen Artikel mit, wohin Sie ihn brauchen.

↓ Markdown ↓ Klartext ↓ PDF

Die Datei wird auf Ihr Gerät heruntergeladen. Von dort aus können Sie sie speichern, in Solo2 importieren oder teilen, wo immer Sie möchten. Cuadernos entscheidet nicht über den Zielort für Sie.

Siegellack-Siegel · SHA-256 617245ec4b88d72d2efca08c4b9544c2ab5de702408f74c009955f20d8e16402

Cuadernos Lacre · Eine Publikation von Menzuri Gestión S.L. ·
geschrieben von R.Eugenio · herausgegeben vom Team von Solo2.