Signal

GUÍA DE PRIVACIDAD

Signal a fondo

Guía completa

Esta guía describe el funcionamiento de Signal (noviembre de 2026) con el detalle suficiente para entender cada una de sus piezas: claves, protocolo, metadatos, servidores y límites.

Del cifrado de extremo a extremo a la criptografía poscuántica

Contenido

INTRODUCCIÓN Y BASES

· Cómo usar esta guía 3

· Los conceptos básicos 3

· La matemática mínima 5

CAPÍTULOS

· Notación 8

01 Visión general: qué protege Signal y qué no 9

02 Las primitivas criptográficas 11

03 Claves de identidad y prekeys 12

04 X3DH: el apretón de manos inicial 13

05 PQXDH: el apretón de manos poscuántico 16

06 El Double Ratchet 18

07 SPQR y el Triple Ratchet 20

08 Mensajes, adjuntos y varios dispositivos 22

09 Sealed Sender y metadatos 24

10 Grupos y credenciales anónimas 26

11 Llamadas 28

12 Verificación de claves 29

13 Cuenta, contactos, PIN y usernames 31

14 Copias de seguridad seguras 33

15 Modelo de confianza y límites 34

16 Buenas prácticas 35

17 Fuentes y lecturas 37

INTRODUCCIÓN

Cómo usar esta guía

Público y prerrequisitos. No hace falta saber de criptografía ni de redes: las secciones "Antes de empezar" explican desde cero los conceptos y la matemática mínima. Después la guía sube de nivel poco a poco hasta llegar a las fórmulas.

Estado del protocolo. Signal combina hoy un acuerdo de claves poscuántico (PQXDH) con un trinquete clásico (Double Ratchet) y otro poscuántico (SPQR), que juntos forman el Triple Ratchet. El despliegue de SPQR es gradual y convive con clientes antiguos, así que los detalles pueden cambiar; ante la duda, la fuente de verdad son las especificaciones publicadas por el proyecto (capítulo 17).

Cómo leerla. Si partes de cero, empieza por "Antes de empezar" y consulta la tabla de notación justo antes del capítulo 2. Cada capítulo abre con una frase que resume la idea; las fórmulas clave llevan una etiqueta, su explicación en palabras y un desglose símbolo a símbolo, y las secundarias se resumen en una idea. Los capítulos 2 a 7 son el núcleo criptográfico y deben leerse en orden. Los capítulos 8 a 11 (mensajes, metadatos, grupos, llamadas) y 12 a 14 (verificación, cuenta, copias) se pueden leer por separado. Los capítulos 15 y 16 resumen los límites y las buenas prácticas.

ANTES DE EMPEZAR

Los conceptos básicos

Para entender Signal bastan cinco ideas. Si ya las conoces, salta a la tabla de notación.

Cifrado de extremo a extremo

Cuando envías un mensaje por internet, pasa por varios ordenadores intermedios: tu operadora, los servidores de la aplicación, los de la operadora del destinatario. En un servicio sin cifrado de extremo a extremo, la empresa puede leerlo en el camino aunque lo proteja con cifrado entre tu móvil y su servidor. En uno **de extremo a extremo** el mensaje se cifra en tu dispositivo y solo se descifra en el del destinatario: los servidores en medio solo ven datos ilegibles.

La pregunta clave es cómo se ponen de acuerdo dos personas que nunca se han visto sobre la llave que van a usar, si todo lo que se dicen puede ser escuchado. Esa es la parte difícil, y a ella está dedicada la mayor parte de la guía.

Claves y firmas

Una **clave** es un número secreto que sirve para cifrar o descifrar. Hay dos tipos. En el cifrado **simétrico** la misma clave cierra y abre, como una llave de casa; es rápido, pero exige que las dos partes ya la compartan. En el cifrado **de clave pública** cada persona tiene una **clave privada**, que no sale nunca de su dispositivo, y una **clave pública** derivada de ella, que puede darse a cualquiera. Con las claves públicas dos personas pueden construir un secreto compartido sin haberse visto.

Una **firma digital** es un sello que solo puede poner quien tiene la clave privada y que cualquiera puede comprobar con la pública. Demuestra que un dato lo publicó el dueño de esa clave y que no se ha modificado.

Hashes y derivación de claves

Un **hash** convierte cualquier dato en una huella de tamaño fijo. Es determinista (los mismos datos dan la misma huella), imposible de revertir y cualquier cambio mínimo produce una huella totalmente distinta. Una **función de derivación de claves** (KDF) usa hashes para convertir un secreto en varias claves independientes: de un mismo secreto salen, por ejemplo, una clave para cifrar y otra para autenticar, sin que conocer una revele la otra. Un **MAC** es una huella con clave: sirve para comprobar que un mensaje no fue alterado y que lo creó alguien que conocía la clave.

Secreto hacia adelante y recuperación

El **secreto hacia adelante** (*forward secrecy*) significa que robar tus claves de hoy no permite leer mensajes de ayer, porque las claves antiguas se borraron. La **recuperación tras compromiso** (*post-compromise security*) es la propiedad simétrica: si alguien copia tus claves actuales, el sistema se cura solo y los mensajes futuros vuelven a ser seguros en cuanto intercambiáis material nuevo. Signal se diseñó para tener las dos.

Metadatos

Los **metadatos** son los datos sobre la comunicación y no su contenido: quién habla con quién, cuándo, desde qué dirección IP, con qué frecuencia. Cifrar el contenido no oculta automáticamente los metadatos, y una parte importante del diseño de Signal consiste en reducirlos.

Glosario rápido

TÉRMINO	QUÉ ES
Cifrado de extremo a extremo	El mensaje se cifra en el emisor y solo se descifra en el receptor
Clave privada / pública	La privada no se comparte nunca; la pública se deriva de ella y sí se comparte
Firma digital	Sello que prueba quién publicó un dato y que no cambió
Hash	Huella de tamaño fijo de unos datos, imposible de revertir
KDF	Función que deriva varias claves independientes de un secreto
MAC	Huella con clave que prueba que un mensaje no fue alterado
Secreto hacia adelante	Robar claves de hoy no abre mensajes de ayer
Recuperación tras compromiso	Tras un robo de claves, los mensajes futuros vuelven a ser seguros
Metadatos	Datos sobre la comunicación: quién, cuándo, desde dónde
Cuántico / poscuántico	Un ordenador cuántico podría romper parte de la criptografía actual; poscuántico es la que resiste

ANTES DE EMPEZAR

La matemática mínima

Toda la criptografía de Signal descansa en tres herramientas: curvas elípticas (para intercambiar claves y firmar), funciones hash (para derivar claves) y, desde hace poco, un mecanismo resistente a ordenadores cuánticos. Aquí está lo justo para seguir el resto.

Aritmética modular

En un reloj, después de las 12 viene la 1: los números "dan la vuelta". La **aritmética modular** trabaja así con un número límite, el módulo. Se escribe `a mod p` al resto de dividir `a` entre `p`. La criptografía usa módulos enormes (de unos 255 bits) y en ellos hay operaciones fáciles de calcular en un sentido e inviables de deshacer en el otro.

Curvas elípticas y Diffie-Hellman

Signal usa la curva **Curve25519**, definida sobre los números módulo el primo `2^255 - 19`. Sus puntos forman un conjunto en el que se puede "sumar" un punto consigo mismo. Existe un punto de partida fijo y público, el **punto base** `G`. Sumar `G` consigo mismo `k` veces da otro punto, que se escribe `k·G`.

Calcular `k·G` a partir de `k` es rápido; recuperar `k` a partir de `k·G` es, con los ordenadores actuales, inviable (el *problema del logaritmo discreto*). De ahí sale la pareja de claves: la **clave privada** es un número `k` de 32 bytes y la **clave pública** es el punto `K = k·G`. La variante **X25519** usa solo una coordenada del punto y es la que Signal emplea para intercambiar claves.

El **intercambio Diffie-Hellman** permite a dos personas fabricar un secreto compartido a la vista de todos. Alice tiene `a` y publica `A = a·G`; Bob tiene `b` y publica `B = b·G`. Entonces:

FÓRMULA CLAVE.

$$a \cdot B = a \cdot (b \cdot G) = (a \cdot b) \cdot G = b \cdot (a \cdot G) = b \cdot A$$

En palabras: cada uno mezcla su secreto con lo que el otro hizo público, y los dos acaban con exactamente el mismo punto; quien solo mira lo público no puede llegar a él.

Qué es cada pieza:

- `G`: el punto base, público y fijo; es el punto de partida que todo el mundo conoce.
- `a` y `b`: los secretos de Alice y de Bob. Son números y nunca salen de su dispositivo.
- `A = a·G` y `B = b·G`: las claves públicas. Se envían a la vista de todos.
- `a·B` y `b·A`: lo que calcula cada uno. Alice combina su secreto con la clave pública de Bob; Bob, el suyo con la de Alice.
- `(a·b)·G`: el resultado común, que es el secreto compartido.

Por qué funciona. Multiplicar es conmutativo: `a·b` es lo mismo que `b·a`. Alice calcula `a·(b·G)` y Bob `b·(a·G)`, y ambos son `(a·b)·G`. Un espía ve `A` y `B`, pero para llegar a `a` o a `b` tendría que resolver el logaritmo discreto, que hoy es inviable.

Un ejemplo con números pequeños. La curva real no se calcula a mano, pero la misma idea funciona con aritmética de reloj de 23 horas, donde "multiplicar un punto por k" se sustituye por "elevar 5 a la k":

1. Alice elige el secreto `a = 6` y publica `5^6 mod 23 = 8`.
2. Bob elige el secreto `b = 15` y publica `5^15 mod 23 = 19`.
3. Alice calcula `19^6 mod 23 = 2`.
4. Bob calcula `8^15 mod 23 = 2`.

Los dos obtienen 2 sin haber enviado nunca 6 ni 15. Con números de 255 bits, deducir 6 a partir de 8 es inviable.

En el resto de la guía, la operación completa «mi secreto con tu clave pública» se escribe `DH(a, B)`. Ese punto, pasado por un KDF (función de derivación de claves), es la clave compartida.

Firmas

Una firma de tipo EdDSA demuestra que se conoce una clave privada `k` sin revelarla. Quien firma elige un número aleatorio `r`, lo convierte en un punto `R = r·G` y mezcla ese punto, su clave pública `K` y el mensaje en un reto `h`. La firma es el par `(R, s)`, donde `s` combina el número aleatorio con el reto y la clave privada de modo que no la delata.

Quien verifica comprueba una igualdad en la que solo aparecen valores públicos:

$$s\,G = R + h\,K$$

> **En palabras:** si la firma es genuina, los dos lados dan el mismo punto; solo quien conoce `k` podía fabricar un `s` que lo cumpla.

Signal necesita que una misma clave de identidad sirva para intercambiar claves (X25519) y para firmar. Para eso usa **XEdDSA**, una forma de firmar con claves de la curva de Diffie-Hellman con el mismo esquema anterior.

Encapsulación de claves (KEM)

Un **KEM** es una forma alternativa de acordar un secreto. Tiene tres operaciones: generar un par de claves, **encapsular** con la clave pública (produce un secreto `ss` y un texto cifrado `ct`) y **desencapsular** con la clave privada (recupera `ss` a partir de `ct`). Quien tiene la clave pública puede crear un secreto y enviar `ct` en abierto; solo el dueño de la clave privada lo recupera. Los KEM poscuánticos como **ML-KEM** (antes llamado Kyber) se basan en retículos, problemas matemáticos distintos de las curvas, y por eso son mucho más grandes: del orden de un kilobyte por clave y por texto cifrado, frente a 32 bytes de X25519.

Por qué importa lo cuántico

Un ordenador cuántico suficientemente grande podría resolver el logaritmo discreto de las curvas elípticas con el algoritmo de Shor y romper X25519 y las firmas. Aún no existe, pero un adversario puede **guardar hoy** el tráfico cifrado para **descifrarlo más adelante** (*harvest now, decrypt later*). Por eso Signal añade capas poscuánticas ahora. El cifrado simétrico (AES-256) y los hashes (SHA-256) se ven afectados mucho menos: el algoritmo de Grover solo reduce a la mitad los bits de seguridad efectivos.

REFERENCIA

Notación

Esta tabla reúne los símbolos que se usan en los capítulos siguientes. Vuelve a ella cuando una fórmula te resulte opaca.

SÍMBOLO	SIGNIFICADO
`a`, `b`, `k`, `r`	Números secretos (claves privadas, aleatorios)
`A`, `B`, `K`, `R`	Puntos de la curva (claves públicas)
`G`	Punto base de Curve25519
`DH(x, Y)`	Diffie-Hellman: el punto `x·Y`, convertido en bytes
`IK_A`, `IK_B`	Clave de identidad (larga duración) de Alice y de Bob
`SPK_B`	Prekey firmada de Bob (se renueva cada cierto tiempo)
`OPK_B`	Prekey de un solo uso de Bob
`EK_A`	Clave efímera de Alice, creada para una sola sesión
`PQPK_B`, `ss`, `ct`	Clave pública KEM de Bob, secreto compartido y texto cifrado del KEM
`‖`	Concatenación de bytes
`KDF`, `HKDF`	Derivación de claves (HKDF con SHA-256 en Signal)
`SK`	Clave compartida inicial de una sesión
`RK`, `CK`, `MK`	Clave raíz, clave de cadena y clave de mensaje del Double Ratchet
`HMAC`	Código de autenticación de mensajes con hash

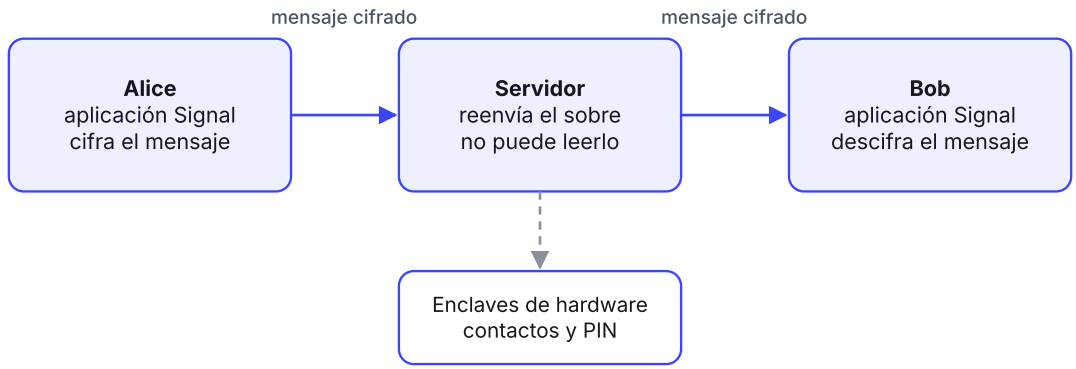
CAPÍTULO 1

Visión general: qué protege Signal y qué no

Signal protege el contenido de tus mensajes y llamadas con cifrado de extremo a extremo y se esfuerza en que su servidor sepa lo mínimo; no oculta que usas Signal ante tu operadora ni protege un dispositivo comprometido.

Las piezas del sistema

Signal es una aplicación (para Android, iOS y escritorio) más un conjunto de servicios que ejecuta la Fundación. Conviene distinguir cada pieza porque cada una ve cosas distintas.



El servidor transporta el mensaje, pero las claves para leerlo solo están en los dispositivos.

PIEZA	QUÉ HACE	QUÉ VE
Cliente (la app)	Genera las claves, cifra y descifra	Todo: es el extremo
Servidor de mensajes	Guarda las claves públicas y entrega mensajes cifrados	Destinatario, dirección IP, hora y tamaño; no el contenido
Enclaves de hardware	Descubrimiento de contactos y recuperación del PIN	Datos protegidos por hardware que ni la operadora del servidor puede leer
Servidores de llamadas	Retransmiten audio y vídeo cifrados	Quién se conecta y cuándo; no el contenido
Almacenamiento de adjuntos	Guarda archivos ya cifrados	Un bloque de bytes sin claves

Qué recorre un mensaje

1. Tu aplicación obtiene del servidor las claves públicas del destinatario y calcula con ellas un secreto compartido (caps. 4 y 5).
2. Cifra el mensaje con una clave que cambia en cada mensaje (cap. 6).
3. Lo envía al servidor, que solo ve a quién va (y, con *Sealed Sender*, no ve de quién; cap. 9).
4. El servidor lo guarda hasta que el destinatario se conecta y lo entrega. Si hay una notificación, el sistema operativo avisa de que hay algo nuevo, pero el texto no viaja por ese canal.
5. El dispositivo del destinatario lo descifra y borra la clave usada.

Qué garantiza el diseño

- **Confidencialidad:** solo los dispositivos de los participantes pueden leer el contenido.
- **Integridad y autenticidad:** cualquier alteración se detecta y los mensajes solo pueden venir de quien conoce la clave de la sesión.
- **Secreto hacia adelante:** las claves de mensajes antiguos se borran.
- **Recuperación tras compromiso:** si te copian las claves, la sesión se cura sola en cuanto hay nuevos intercambios.
- **Repudio criptográfico:** los mensajes se autentican con códigos de clave compartida y no con firmas, de modo que un tercero no puede demostrar con esos datos quién escribió qué.
- **Resistencia poscuántica híbrida:** el acuerdo de claves y el trinquete añaden un componente poscuántico además del clásico (caps. 5 y 7).

Qué no garantiza

- **Un dispositivo comprometido.** Si alguien controla tu teléfono lee los mensajes cuando los lees tú. Ninguna criptografía lo evita.

- **Anonimato.** Registrarse exige un número de teléfono y el servidor ve tu dirección IP cuando te conectas (caps. 13 y 15).
- **Metadatos mínimos, no nulos.** El servidor necesita saber a quién entregar cada mensaje. Los caps. 9 y 10 explican qué se oculta y qué no.
- **Que tu interlocutor sea quien dice ser.** Signal da herramientas para comprobarlo (cap. 12), pero hay que usarlas.
- **Que el otro no copie la conversación.** Una captura de pantalla o un reenvío no se pueden impedir.

CAPÍTULO 2

Las primitivas criptográficas

Signal no inventa algoritmos: combina piezas estándar y muy estudiadas, y su aportación está en cómo las ensambla.

FUNCIÓN	ALGORITMO	DÓNDE SE USA
Intercambio de claves	X25519 (Curve25519)	Apretón de manos inicial y trinquete
Firmas	XEdDSA (sobre Curve25519)	Firma de prekeys y certificados
Acuerdo poscuántico	ML-KEM / Kyber	PQXDH y SPQR
Derivación de claves	HKDF con SHA-256	Claves de sesión y de mensaje
Autenticación	HMAC-SHA256	Cadena de claves y etiqueta de mensaje
Cifrado de mensajes y adjuntos	AES-256 en modo CBC + HMAC	Contenido de mensajes y archivos
Cifrado de campos	AES-256-GCM	Perfil y datos almacenados
Huellas	SHA-512	Números de seguridad
Estiramiento del PIN	Argon2	PIN de recuperación
Credenciales anónimas	Ristretto255	Grupos y usernames

Derivar claves con HKDF

HKDF convierte un secreto de entrada en tantas claves como se necesiten. Primero lo «extrae» en una clave maestra con una función HMAC y después la «expande» en una serie de bloques, mezclando en cada uno una etiqueta que dice para qué se usará la clave.

La salida es la concatenación de los bloques generados, recortada a la longitud pedida. Cambiar la etiqueta da claves totalmente distintas a partir del mismo secreto, y eso permite separar usos.

Cifrar y autenticar un mensaje

Cada mensaje usa una clave de mensaje `MK` de un solo uso (cap. 6). De ella se derivan con HKDF tres valores: una clave de cifrado de 32 bytes, una clave de autenticación de 32 bytes y un vector inicial de 16 bytes. El mensaje se cifra con AES-256 en modo CBC usando la clave de cifrado y el vector inicial.

Sobre el texto cifrado, las claves públicas de ambas partes y la cabecera del mensaje se calcula un HMAC-SHA256 con `k_mac`, que se recorta (en la implementación de Signal, a 8 bytes) y se añade. El receptor verifica la etiqueta antes de descifrar: "autenticar y después descifrar" evita ataques que alteran el texto cifrado. Como la clave de mensaje se usa una sola vez, la etiqueta corta es suficiente.

CAPÍTULO 3

Claves de identidad y prekeys

Cada dispositivo de Signal publica en el servidor un paquete de claves públicas (el *prekey bundle*) para que cualquiera pueda iniciar una conversación cifrada con él aunque esté apagado.

El problema

Para cifrar el primer mensaje a Bob, Alice necesita un secreto compartido con él, pero Bob puede estar sin conexión. La solución es que Bob deje de antemano en el servidor material público del que Alice pueda derivar ese secreto sin preguntarle nada.

Qué claves tiene cada dispositivo

CLAVE	DURACIÓN	FUNCIÓN
Clave de identidad `IK`	Larga (meses o años)	Identifica a la cuenta (la comparten sus dispositivos); autentica el acuerdo de claves
Prekey firmada `SPK`	Media; se renueva periódicamente	Material de intercambio, firmado con `IK`
Prekeys de un solo uso `OPK`	Una vez	Aportan secreto hacia adelante al primer mensaje
Prekey KEM firmada `PQSPK`	Media	Componente poscuántico, firmado con `IK`
Prekeys KEM de un solo uso	Una vez	Igual que las `OPK`, pero poscuánticas

Todas las claves privadas se quedan en el dispositivo. En el servidor solo están las públicas, y cada prekey firmada lleva la firma de la clave de identidad de Bob, que demuestra que le pertenece.

Cómo se reparten

Al registrarse, el dispositivo sube su clave de identidad, una prekey firmada, una prekey KEM firmada y un lote de prekeys de un solo uso, y repone el lote cuando se agota. Cuando alguien quiere hablar con Bob, el servidor le entrega la clave de identidad, las prekeys firmadas y **una** prekey de un solo uso de cada tipo, y la borra de su lista para que no se reparta dos veces.

Si el lote se agota, el acuerdo sigue funcionando sin las prekeys de un solo uso: es algo menos robusto frente a repeticiones, pero no se rompe. Para el componente poscuántico existe una prekey KEM "de último recurso" que nunca se borra, precisamente para que siempre haya una disponible.

CAPÍTULO 4

X3DH: el apretón de manos inicial

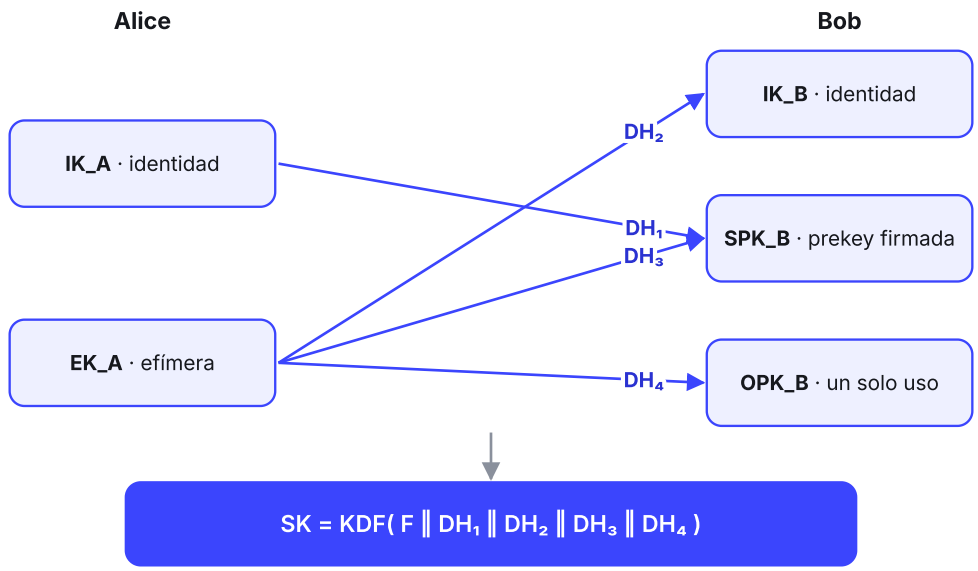
X3DH (*Extended Triple Diffie-Hellman*) combina tres o cuatro intercambios Diffie-Hellman para que Alice y Bob acuerden una clave inicial aunque Bob esté desconectado.

Los pasos

1. Alice pide al servidor el paquete de Bob: `IK_B`, `SPK_B`, su firma y, si hay, una `OPK_B`.
2. Verifica la firma de `SPK_B` con `IK_B`. Si no cuadra, aborta.
3. Crea una clave efímera `EK_A`, usada solo para esta sesión.
4. Calcula cuatro intercambios Diffie-Hellman y los junta.
5. Deriva con HKDF la clave inicial `SK` y borra la parte privada de `EK_A` y los intercambios intermedios.
6. Envía a Bob su primer mensaje cifrado, junto con `IK_A`, `EK_A` y qué prekeys usó.

Los cuatro intercambios

Alice mezcla cuatro intercambios Diffie-Hellman distintos en una sola clave inicial. Cada uno aporta una garantía diferente; la tabla de más abajo las resume.



Cada línea es un Diffie-Hellman entre una clave de Alice y una de Bob; sus resultados se juntan en la clave inicial.

FÓRMULA CLAVE.

$$DH_1 = DH(IK_A,\ SPK_B)$$
$$DH_2 = DH(EK_A,\ IK_B)$$
$$DH_3 = DH(EK_A,\ SPK_B)$$
$$DH_4 = DH(EK_A,\ OPK_B)$$

En palabras: Alice hace cuatro mezclas de «un secreto mío con una clave pública tuya», combinando sus claves de larga duración con las de Bob y con una clave efímera suya.

Qué es cada pieza:

- `IK_A`, `IK_B`: claves de identidad de Alice y de Bob (larga duración).
- `SPK_B`: la prekey firmada de Bob, que él dejó en el servidor.
- `OPK_B`: una prekey de un solo uso de Bob, si quedaba alguna.
- `EK_A`: la clave efímera de Alice, creada solo para esta sesión.
- `DH(x, Y)`: el Diffie-Hellman de la sección anterior: mi secreto `x` con tu pública `Y`.

$$SK = \mathrm{KDF}\big(F \,\|\, DH_1 \,\|\, DH_2 \,\|\, DH_3 \,\|\, DH_4\big)$$

En palabras: se pegan los cuatro resultados, se añade una constante para separar este uso de otros y todo pasa por una función de derivación de claves que devuelve la clave inicial de la sesión.

Qué es cada pieza:

- `DH_1 … DH_4`: los cuatro secretos compartidos de arriba.
- `F`: una constante de 32 bytes con todos los bits a uno; separa este uso de otros.
- `||`: concatenar, pegar un valor tras otro.
- `KDF`: la función que mezcla todo y da una clave uniforme de la longitud pedida.
- `SK`: la clave compartida inicial de la sesión, que alimentará el Double Ratchet.

Por qué funciona. Cada `DH_i` es un Diffie-Hellman, así que Bob puede calcular el mismo valor con sus claves privadas (por ejemplo, `DH(IK_A, SPK_B)` lo obtiene él como `DH(spk_B, IK_A)`) y llega al mismo `SK`. Ninguno de los cuatro secretos viaja por la red, y un atacante necesitaría claves privadas de ambos para reconstruirlos.

Si no quedaba ninguna `OPK_B`, se omite `DH_4`.

Qué aporta cada intercambio

INTERCAMBIO	QUÉ APORTA
`DH_1`	Autentica a Alice: solo quien tiene la clave privada de `IK_A` puede calcularlo
`DH_2`	Autentica a Bob: solo quien tiene la clave privada de `IK_B` puede calcularlo
`DH_3`	Secreto hacia adelante: mezcla una clave efímera de Alice con la prekey de Bob
`DH_4`	Añade un valor que se usa una sola vez y evita repeticiones del primer mensaje

Datos asociados y repudio

Las dos claves de identidad forman unos *datos asociados* `AD = IK_A || IK_B` que se incorporan a la etiqueta de autenticación de los mensajes, de manera que un mensaje queda ligado a esta pareja concreta. Como la autenticación se hace con Diffie-Hellman (algo que cualquiera de los dos puede calcular) y no con una firma sobre el mensaje, ninguno de los dos puede probar ante un tercero que el otro escribió algo: es el *repudio criptográfico*.

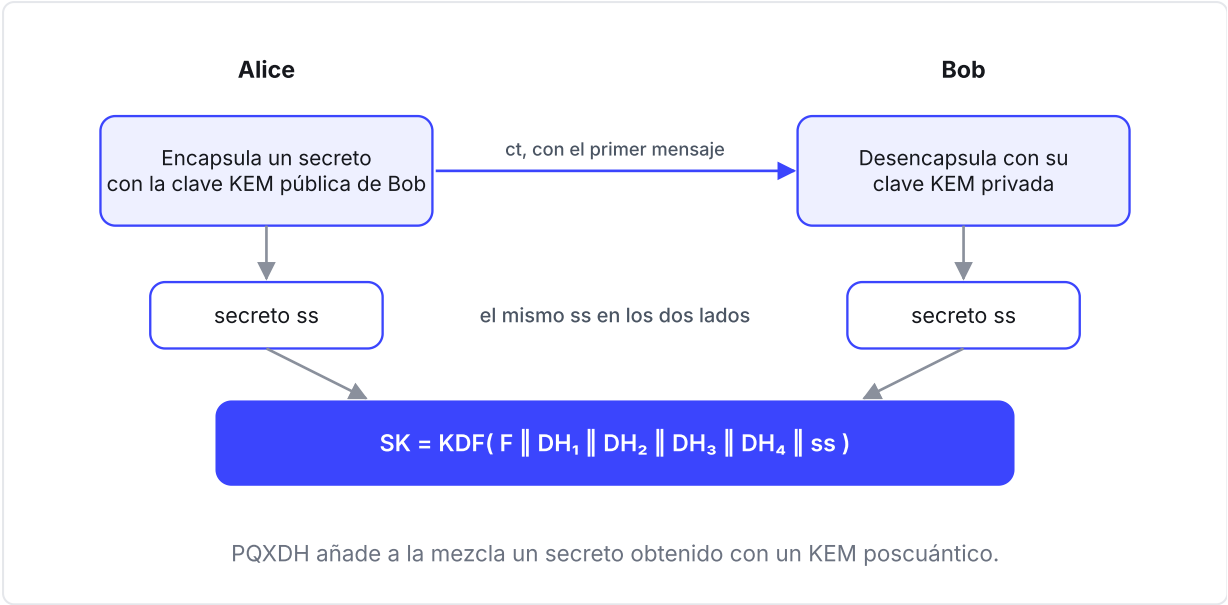
CAPÍTULO 5

PQXDH: el apretón de manos poscuántico

PQXDH añade a X3DH un intercambio de claves poscuántico para que un adversario que guarde hoy el tráfico no pueda descifrarlo cuando exista un ordenador cuántico.

Qué cambia respecto a X3DH

Signal introdujo PQXDH en 2023 como primer paso hacia la resistencia cuántica. Bob publica además de las prekeys clásicas una prekey KEM firmada (y opcionalmente otras de un solo uso). Alice la usa para **encapsular** un secreto `ss` que solo Bob podrá recuperar, y envía el texto cifrado `ct` junto al primer mensaje. Después añade ese secreto a la mezcla:



PQXDH añade a la mezcla un secreto obtenido con un KEM poscuántico.

$$SK = \mathrm{KDF}\big(F \,\|\, DH_1 \,\|\, DH_2 \,\|\, DH_3 \,\|\, DH_4 \,\|\, ss\big)$$

En palabras: es la misma mezcla de X3DH con un ingrediente más, el secreto del KEM; para recuperar `SK` hay que conocer a la vez los secretos clásicos y el poscuántico.

Bob recupera `ss = Decaps(pq_B, ct)` con su clave privada y obtiene el mismo `SK`. El KEM que especifica el protocolo es CRYSTALS-Kyber en su variante de mayor nivel (1024), el que NIST estandarizó después con el nombre de ML-KEM.

Por qué es híbrido

`SK` depende a la vez del valor de las curvas y del valor del KEM. Para recuperarlo hay que romper **las dos** cosas: si alguien descubre un fallo en Kyber, los cuatro intercambios de curva siguen protegiendo; si un ordenador cuántico rompe las curvas, `ss` sigue protegiendo. Es una apuesta prudente porque los esquemas poscuánticos son mucho más jóvenes que las curvas.

Qué protege y qué no

- **Protege la confidencialidad del secreto inicial** frente a quien grabe tráfico hoy y tenga un ordenador cuántico en el futuro.
- **No protege la autenticación frente a un atacante cuántico activo.** Las claves de identidad y las firmas siguen siendo de curva elíptica; un adversario que ya tuviera ese ordenador podría hacerse pasar por alguien en una sesión nueva. El riesgo que cubre PQXDH es el de "guardar ahora, descifrar después".
- **No cubre la sesión entera.** PQXDH solo protege el secreto inicial; los intercambios posteriores del Double Ratchet seguían siendo clásicos. Esa parte la cierra el capítulo 7.

CAPÍTULO 6

El Double Ratchet

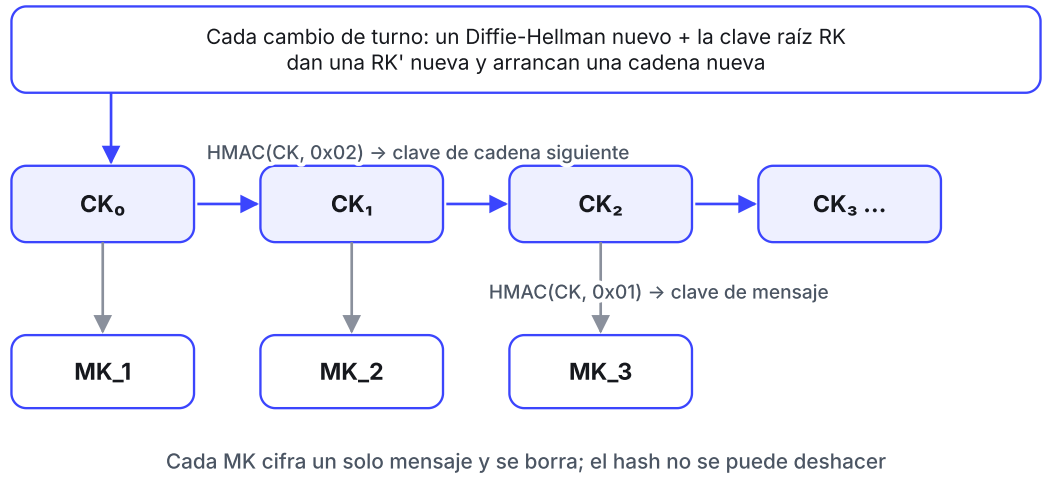
El Double Ratchet ("doble trinquete") renueva la clave de cifrado en cada mensaje y en cada cambio de turno, de modo que robar una clave solo da acceso a un trocito de la conversación.

La idea

Un trinquete es un mecanismo que gira en un solo sentido. Aquí son dos. El **trinquete simétrico** hace avanzar una clave con una función hash tras cada mensaje; como el hash no se puede revertir, quien robe la clave actual no puede calcular las anteriores (secreto hacia adelante). El **trinquete Diffie-Hellman** mezcla material nuevo cada vez que la conversación cambia de dirección; como el material nuevo es secreto para el atacante, este queda fuera aunque haya copiado las claves de antes (recuperación tras compromiso).

Las tres cadenas

Cada parte mantiene una **clave raíz** `RK` (que sale del apretón de manos inicial, capítulos 4 y 5) y dos cadenas de claves: una para enviar y otra para recibir, cada una con su **clave de cadena** `CK`. Cada vez que cambia el turno se hace un paso del trinquete Diffie-Hellman: quien va a hablar genera un par de claves nuevo, hace un Diffie-Hellman con la última clave pública de la otra parte y lo usa para actualizar la clave raíz y arrancar una cadena nueva:



Una cadena que solo avanza: de cada clave de cadena sale la clave de un mensaje y la clave siguiente.

FÓRMULA CLAVE.

$$(RK',\ CK) = \mathrm{KDF}_{RK}\big(RK,\ DH(dh_s,\ DH_r)\big)$$

En palabras: cuando cambia el turno, la parte que va a hablar crea un par de claves nuevo, lo mezcla con la clave pública más reciente de la otra parte y, con ese secreto nuevo y la clave raíz anterior, obtiene una clave raíz nueva y una cadena nueva.

Qué es cada pieza:

- `RK` : la clave raíz actual, que sale del apretón de manos inicial.
- `dh_s` : la clave privada Diffie-Hellman nueva de quien envía.
- `DH_r` : la última clave pública Diffie-Hellman recibida de la otra parte.
- `DH(dh_s, DH_r)` : el secreto fresco que ninguno tenía antes.
- `KDF_RK` : la función que mezcla la raíz vieja con el secreto fresco.
- `RK'` y `CK` : la clave raíz nueva y la clave de la primera cadena nueva.

Por qué funciona. El secreto fresco depende de una clave privada nueva que un atacante que copió el estado anterior no tiene, así que la raíz nueva queda fuera de su alcance: eso es la recuperación tras compromiso.

Dentro de una cadena, cada mensaje consume una clave de cadena y produce una clave de mensaje de un solo uso y la clave de cadena siguiente:

FÓRMULA CLAVE.

$$MK = \mathrm{HMAC}(CK,\ 0x01), \qquad CK' = \mathrm{HMAC}(CK,\ 0x02)$$

En palabras: de la clave de cadena actual salen, por separado, la clave de un solo mensaje y la clave de cadena siguiente; la anterior se borra.

Qué es cada pieza:

- `CK` : la clave de cadena de este paso.
- `HMAC(CK, 0x01)` : una huella con clave de `CK` y la constante 1; da la clave del mensaje `MK` .
- `HMAC(CK, 0x02)` : lo mismo con la constante 2; da la clave de cadena siguiente `CK'` .

Un ejemplo. Es una cinta que solo avanza: de `CK_0` salen `MK_1` y `CK_1` ; de `CK_1` salen `MK_2` y `CK_2` ; de `CK_2` salen `MK_3` y `CK_3` . Cada `MK_n` cifra un mensaje y se borra.

Por qué funciona. HMAC no se puede revertir: con `CK_2` en la mano no hay forma de calcular `CK_1` ni `MK_2` . Quien robe la clave actual solo lee los mensajes siguientes, no los anteriores.

Tras cifrar o descifrar, se borran `MK` y la `CK` anterior. `MK` pasa a HKDF para obtener la clave de cifrado, la de autenticación y el vector inicial (capítulo 2).

La cabecera de cada mensaje

Cada mensaje viaja con una cabecera que incluye la clave pública Diffie-Hellman actual del emisor, el número del mensaje dentro de su cadena y la longitud de la cadena anterior. Con eso el receptor sabe si debe dar un paso del trinquete Diffie-Hellman y cuántos mensajes se han saltado.

Un ejemplo

TURNO	QUÉ OCURRE	CLAVE NUEVA
1. Alice envía 3 mensajes	Paso simétrico tras cada uno	`MK_1`, `MK_2`, `MK_3` de su cadena
2. Bob responde	Bob genera un par nuevo y da un paso Diffie-Hellman	Cadena nueva de Bob
3. Alice contesta	Alice genera un par nuevo y da un paso	Cadena nueva de Alice

Si un atacante copia el estado de Alice tras el turno 1, puede leer los mensajes posteriores de esa cadena; pero en cuanto Bob responde con una clave que el atacante no tiene, la raíz cambia y el atacante queda fuera.

Mensajes desordenados

La red puede entregar los mensajes desordenados o perder alguno. Si llega el mensaje 5 antes que el 4, el receptor avanza la cadena hasta el 5 y guarda la clave de mensaje del 4 por si llega después. Para que esto no sea una fuga de seguridad, el número de claves guardadas está limitado y se borran con el tiempo.

CAPÍTULO 7

SPQR y el Triple Ratchet

El Sparse Post-Quantum Ratchet (SPQR) añade a la conversación un tercer trinquete poscuántico que corre en paralelo al Double Ratchet y completa lo que PQXDH dejó a medias.

El problema

PQXDH protege el secreto inicial, pero a partir de ahí el trinquete Diffie-Hellman usa curvas elípticas. Un adversario cuántico que hubiera guardado el tráfico podría romper esos pasos más tarde. La solución natural sería sustituir los pasos de curva por un KEM poscuántico, pero ML-KEM tiene claves y textos cifrados de alrededor de un kilobyte cada uno (en la variante de 768, 1184 y 1088 bytes), mientras que una clave X25519 ocupa 32 bytes. Meter eso en cada mensaje haría crecer el tráfico y el coste de cada conversación.

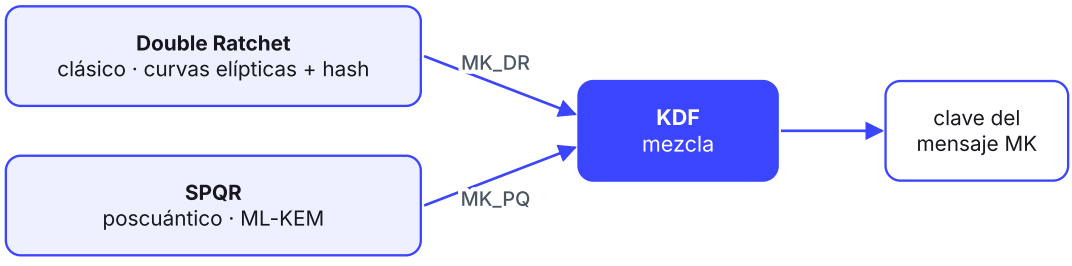
La solución: un trinquete disperso

Signal publicó SPQR en octubre de 2025, desarrollado junto a investigadores académicos y con apoyo de empresas de criptografía para su análisis formal. Sus ideas principales son:

- **Un KEM poscuántico (ML-KEM 768)** en lugar de Diffie-Hellman para hacer avanzar el trinquete.
- **Códigos de borrado:** la clave pública y el texto cifrado, que son grandes, se trocean y reparten en muchos mensajes normales, y el receptor puede reconstruirlos aunque falten algunos trozos. Así cada mensaje solo carga una fracción pequeña.
- **Avance por épocas:** el trinquete no avanza en cada mensaje sino cada cierto tiempo, cuando se ha completado un intercambio. Esto lo hace "disperso" (*sparse*).

El Triple Ratchet

SPQR no sustituye al Double Ratchet: corre a su lado. Cada mensaje se cifra con una clave que mezcla las dos salidas mediante un KDF:



Para recuperar la clave hay que romper los dos trinquetes a la vez

El trinquete poscuántico corre en paralelo al clásico y sus salidas se mezclan.

$$MK = \mathrm{KDF}\left(MK_{DR} \,\|\, MK_{PQ}\right)$$

En palabras: la clave de cada mensaje mezcla la que sale del trinquete clásico y la que sale del poscuántico.

Como con PQXDH, la mezcla es **híbrida**: hay que romper a la vez el trinquete clásico y el poscuántico para recuperar la clave. Mantener el Double Ratchet probado conserva las garantías que ya se conocían y añade las nuevas.

Despliegue y compatibilidad

Para no interrumpir el servicio, las sesiones pueden degradarse temporalmente al Double Ratchet cuando el otro extremo aún no soporta SPQR. Signal ha anunciado que, cuando todos los clientes lo admitan, archivará las sesiones que no lo usen para que toda conversación tenga protección poscuántica. Mientras tanto la protección depende de que ambos extremos estén actualizados. Su fortaleza real depende también de una implementación correcta y de que ML-KEM siga resistiendo el análisis, de ahí el diseño híbrido.

CAPÍTULO 8

Mensajes, adjuntos y varios dispositivos

El Double Ratchet cifra mensajes sueltos; este capítulo explica cómo se empaquetan, qué pasa con los archivos y cómo funciona una misma cuenta en varios dispositivos.

El sobre del mensaje

Cada mensaje cifrado lleva una cabecera y un cuerpo. El primer mensaje de una sesión incluye además los datos del apretón de manos (clave de identidad y efímera de Alice, prekeys usadas y, desde PQXDH, el texto cifrado del KEM); los siguientes solo llevan la cabecera del trinquete. Antes de cifrar, el contenido se rellena (*padding*) hasta tamaños fijos para ocultar la longitud exacta del texto.

Adjuntos

Los archivos grandes no viajan dentro del mensaje. La aplicación genera una clave aleatoria de 64 bytes (32 para cifrar y 32 para autenticar), cifra el archivo con AES-256-CBC y le calcula un HMAC-SHA256, y sube el resultado cifrado a un almacenamiento externo. El enlace de descarga, la clave y la huella del archivo viajan **dentro** del mensaje cifrado de extremo a extremo. El servidor de almacenamiento solo ve un bloque de bytes sin clave.

Una cuenta, varios dispositivos

Una cuenta puede tener un teléfono principal y varios dispositivos vinculados (escritorio, tableta). Cada dispositivo tiene su propia identidad de dispositivo y una sesión independiente con cada dispositivo de tus contactos. Cuando envías un mensaje, tu aplicación lo cifra **una vez por cada dispositivo** del destinatario y de tus propios dispositivos, y envía todas las copias; la gestión de estas sesiones múltiples sigue un protocolo llamado *Sesame*.

Para vincular un dispositivo se escanea un código QR en el teléfono principal. Ese código contiene una clave efímera de la nueva aplicación; el teléfono la usa para entregarle, cifrado, el material de la cuenta (claves de perfil y de identidad). **Consecuencia importante:** un dispositivo vinculado forma parte de tu cuenta y puede leer tus mensajes; escanear un QR de vinculación que no hayas generado tú equivale a darle tu cuenta a otra persona.

Mensajes temporales y recibos

Los mensajes que desaparecen llevan un temporizador dentro del propio mensaje cifrado, y cada aplicación los borra al cumplirse. Es un acuerdo entre aplicaciones y no una garantía técnica: un interlocutor con un cliente modificado o con una captura de pantalla puede conservarlos. Los recibos de entrega y lectura son también mensajes cifrados y se pueden desactivar.

Qué guarda el servidor

Mientras el destinatario está desconectado, el servidor guarda el mensaje cifrado en una cola y lo borra cuando se entrega. No conserva el historial de conversaciones: este vive en los dispositivos y, si lo activas, en una copia de seguridad cifrada que el servidor no puede leer (capítulo 14).

CAPÍTULO 9

Sealed Sender y metadatos

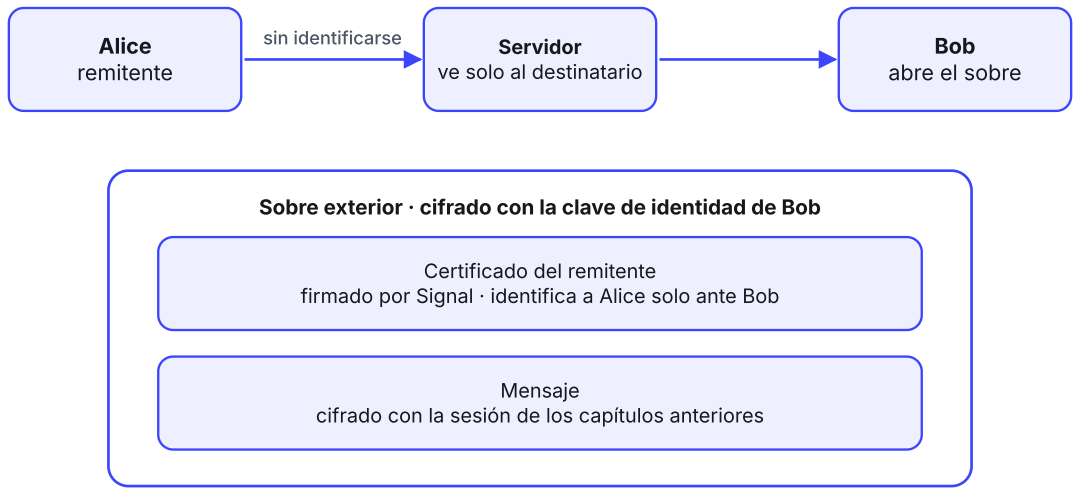
Sealed Sender ("remitente sellado") esconde al servidor quién envía cada mensaje: el servidor sabe a quién se lo entrega, pero no de quién viene.

El problema

En un servicio normal, cada mensaje que subes va acompañado de tus credenciales para que el servidor sepa que eres tú y pueda limitar abusos. Eso significa que el servidor conoce el remitente y el destinatario de cada mensaje, que es precisamente un grafo social. Cifrar el contenido no lo evita.

La solución

El remitente guarda su identidad **dentro** del sobre cifrado, de forma que solo el destinatario puede leerla:



La identidad del remitente viaja dentro del sobre: el servidor no la ve, el destinatario sí.

1. El remitente obtiene del servidor un **certificado de remitente**, firmado por el propio servicio, con validez corta, que atestigua que su clave de identidad pertenece a su cuenta.

2. Cifra el mensaje con la sesión de los capítulos anteriores y mete el resultado, junto con el certificado, en un segundo sobre cifrado con la clave de identidad del destinatario y una clave efímera.
3. Sube ese sobre al servidor **sin autenticarse como remitente**. En su lugar presenta una prueba de que tiene permiso para escribir al destinatario, una clave de acceso derivada de la clave de perfil de este.
4. El servidor comprueba esa clave de acceso, no sabe quién es el remitente y entrega el sobre.
5. El destinatario abre el sobre exterior, lee el certificado, verifica la firma y después descifra el mensaje con la sesión.

La clave de acceso se deriva de la clave de perfil del destinatario, que normalmente solo conocen sus contactos. Quien no la tiene no puede enviar mensajes sellados; el destinatario puede permitir mensajes sellados "de cualquiera" con un ajuste, a costa de aceptar más posible spam.

Qué sigue sabiendo el servidor

Sealed Sender mejora la situación pero no la resuelve del todo. La siguiente tabla resume qué ve el servicio.

DATO	¿LO VE EL SERVIDOR?
Contenido de mensajes y llamadas	No
Remitente de un mensaje sellado	No (el destinatario sí)
Destinatario y hora de entrega	Sí
Tamaño aproximado del mensaje	Sí
Tu dirección IP al conectarte	Sí
Tus contactos	No en claro (descubrimiento privado, capítulo 13)
Miembros y título de tus grupos	No en claro (capítulo 10)
Nombre y foto de perfil	Solo cifrados con la clave de perfil
Tu número de teléfono	Sí (se usa para registrarse)
Fecha de creación de la cuenta y última conexión	Sí

La última fila tiene valor práctico: en las citaciones judiciales que Signal ha hecho públicas, esos fueron los únicos datos de usuario que pudo entregar, junto al número de teléfono.

Ataques de correlación

Un servidor que vea quién entrega y quién recibe a la vez, junto a las IP y las horas, puede intentar emparejar remitentes y destinatarios por simple coincidencia temporal: si Alice se conecta y envía un sobre justo antes de que Bob reciba uno y conteste, la correlación es fuerte aunque el sobre esté sellado. Trabajos académicos han mostrado que con suficientes observaciones se pueden inferir conversaciones. Sealed Sender eleva el coste de esa vigilancia, pero hay que confiar en que el operador no la haga; ocultar la IP exige además una VPN o una red como Tor.

CAPÍTULO 10

Grupos y credenciales anónimas

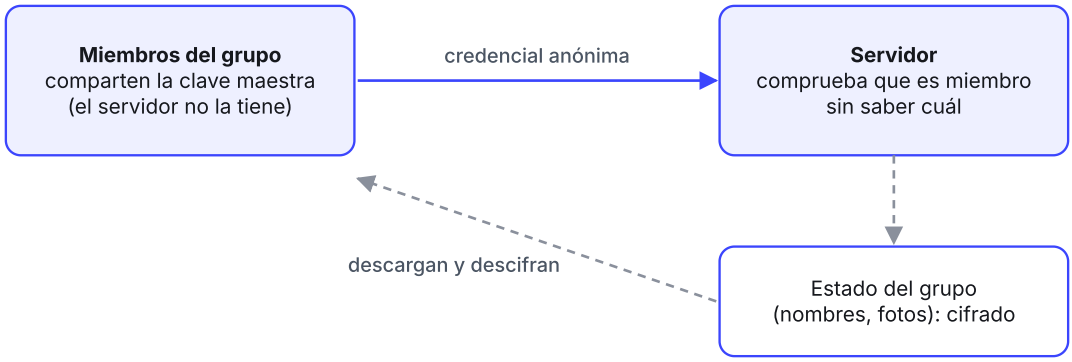
Signal organiza los grupos para que el servidor guarde su estado cifrado y compruebe quién puede actuar sobre él sin saber quién es.

El problema

Un grupo tiene un nombre, una foto y una lista de miembros, y el servidor tiene que decidir quién puede leerlos o modificarlos. Si el servidor viera esa lista, conocería las relaciones entre personas. La meta es que no la vea y que, aun así, pueda negar el acceso a quien no pertenece al grupo.

Clave maestra y estado cifrado

Al crear un grupo, la aplicación genera una **clave maestra de grupo** de 32 bytes aleatorios, que reparte a los miembros mediante mensajes normales cifrados de extremo a extremo. De ella se derivan las claves con que se cifran el título, la foto, la descripción y la lista de miembros, y un identificador público del grupo. El servidor guarda ese estado cifrado y versionado, y los miembros se identifican en él mediante un cifrado determinista de su identificador: el servidor puede ver que dos entradas son distintas, pero no a quién corresponde cada una.



El servidor admite a los miembros sin saber quién es cada uno y guarda un estado que no puede leer.

Credenciales anónimas

Signal necesita que el servidor acepte solo a miembros sin que se identifiquen. Lo logra con **credenciales anónimas de verificación con clave** (*keyed-verification anonymous credentials*), la construcción llamada *zkgroup*, trabajo de Chase, Perrin y Zaverucha y basada en el grupo Ristretto255:

1. El servidor emite a cada cuenta, cada día, una credencial de autenticación que firma su identificador y la fecha.
2. Para actuar sobre un grupo, el miembro presenta una **prueba de conocimiento cero**: demuestra que tiene una credencial válida para un identificador que figura (cifrado) en la lista del grupo, sin revelar cuál.
3. El servidor verifica la prueba y autoriza la acción, pero no aprende qué miembro la hizo.

Para que nuevos miembros puedan ver los perfiles de los demás, existe un esquema similar con **credenciales de clave de perfil**. De este modo el servidor puede aplicar permisos de administrador o de escritura sin ver la lista.

Cómo se envían los mensajes del grupo

Cifrar un mensaje por separado para cada miembro es caro en grupos grandes. Signal usa **claves de emisor** (*sender keys*): cada miembro crea una cadena de claves propia y la entrega a los demás por sus sesiones individuales. Después, cada mensaje se cifra una sola vez con la cadena del emisor y se reparte a todos. Para enviar esos mensajes de forma sellada sin revelar la pertenencia al grupo, Signal ha añadido pruebas de pertenencia anónimas llamadas *group send endorsements*.

Qué queda expuesto

- Los miembros del grupo ven la lista y, según ajustes, los números de teléfono de los demás; los usernames permiten invitar o unirse sin compartir el número.
- El servidor sabe qué cuentas se conectan, desde qué IP y a qué horas, y cuántos sobres entrega, pero no a qué grupo pertenecen.
- Un miembro con un cliente modificado puede copiar los mensajes: en un grupo todos son extremos.

CAPÍTULO 11

Llamadas

Las llamadas de voz y vídeo de Signal usan tecnología WebRTC con las claves acordadas dentro de mensajes cifrados de extremo a extremo, y esconden tu IP si se lo pides.

Llamadas de uno a uno

Para iniciar una llamada, las dos aplicaciones intercambian mensajes de señalización (la oferta, la respuesta y las direcciones de red posibles) y lo hacen como mensajes normales de Signal, es decir, cifrados con la sesión de los capítulos anteriores. Dentro de esa señalización se acuerdan las claves con que se cifra el audio y el vídeo (SRTP). El servidor que ayuda a conectar nunca recibe esas claves.

La llamada puede ir **directa** entre los dos dispositivos o **por un servidor de retransmisión** (TURN) de Signal, que reenvía paquetes ya cifrados. En una llamada directa, cada parte ve la dirección IP de la otra. Existe un ajuste, "Retransmitir siempre las llamadas", que obliga a pasar por el servidor: a cambio de algo de latencia, tu contacto no ve tu IP, y Signal tampoco ve el contenido. Las llamadas de alguien que no está en tus contactos se retransmiten por defecto.

Llamadas de grupo

En una llamada con varias personas, cada una envía su audio y vídeo a un servidor, llamado **SFU**, que lo reenvía a las demás sin descifrarlo. El cifrado de extremo a extremo se mantiene así:

1. Un participante crea una clave de medios y la reparte a los demás mediante mensajes individuales cifrados con sus sesiones.

2. Cada paquete de audio y vídeo se cifra con esa clave antes de salir del dispositivo, de modo que el servidor reenvía bytes opacos.
3. Cuando alguien entra o sale, se genera una clave nueva y se reparte, para que quien se fue no pueda seguir leyendo y quien entra no pueda leer lo anterior.

En los enlaces de llamada, la clave viaja en la parte del enlace que el navegador no envía al servidor.

Qué ve el servidor

El servidor de llamadas ve las direcciones IP de quienes se conectan, cuándo entran y salen y cuánto tráfico generan; no ve el contenido. Si la IP te preocupa, activa la retransmisión y, para más protección, usa una VPN o Tor.

CAPÍTULO 12

Verificación de claves

Todo el cifrado de extremo a extremo depende de que la clave pública que recibes sea realmente la de tu interlocutor; los números de seguridad y la verificación automática sirven para comprobarlo.

El problema de la confianza inicial

Signal reparte las claves públicas a través de su servidor. Si ese servidor (o alguien que lo controle) entregara a Alice una clave que no es la de Bob, sino la de un atacante, Alice cifraría para el atacante, que podría reenviar el mensaje a Bob y leerlo en medio. Es un ataque de intermediario (*man in the middle*). El cifrado no lo detecta por sí solo: hay que comparar las claves por un canal independiente.

Números de seguridad

Cada conversación tiene un **número de seguridad** de 60 dígitos que depende de las dos claves de identidad. Cada parte aporta una huella de 30 dígitos, que se calcula aplicando SHA-512 miles de veces (más de cinco mil iteraciones, para encarecer la fuerza bruta) a su clave de identidad y su identificador. Las dos huellas se ordenan y se concatenan para que ambas partes vean el mismo número.

Si Alice y Bob ven el mismo número (en persona, por llamada o escaneando el código QR del otro), saben que cada uno tiene la clave genuina del otro: un atacante en medio habría producido un número distinto. Cuando la clave de identidad de un contacto cambia, Signal avisa de que "el número de seguridad ha cambiado". Suele ser algo inofensivo (reinstaló la aplicación, cambió de teléfono) pero es también lo que verías ante un ataque.

Verificación automática de claves

Comparar números a mano casi nadie lo hace. Por eso Signal añadió en 2026 la **verificación automática de claves**, basada en un registro de transparencia:

- Signal anota los cambios de número de teléfono y de username en un **registro de solo añadir** (un árbol de hashes). Los identificadores se ocultan con funciones pseudoaleatorias verificables para que los auditores no vean datos de usuarios.
- **Auditores independientes** (Cloudflare y Trail of Bits) comprueban que el registro es coherente y que todo el mundo ve lo mismo. Tu aplicación también comprueba periódicamente tus propios identificadores.
- En la pantalla de la conversación, "Ver número de seguridad" ofrece "Verificar automáticamente"; si todo cuadra, aparece "Cifrado verificado".

Tiene límites: necesita el número de teléfono del contacto (no sirve si solo lo conoces por username), falla si ese contacto cambió de número hasta que lo confirmes por otra vía, y demuestra que *todos ven la misma clave para ese identificador*, no *quién controla* la cuenta: una cuenta robada no se detecta. Se puede desactivar en Privacidad > Avanzado.

MÉTODO	QUÉ DEMUESTRA	QUÉ REQUIERE
Número de seguridad	La clave que ves es la que tiene tu contacto	Comparar el número o el QR por otro canal
Verificación automática	Todos ven la misma clave para ese número	Tener el número de teléfono del contacto

CAPÍTULO 13

Cuenta, contactos, PIN y usernames

Este capítulo recorre cómo se crea una cuenta, cómo encuentra Signal a tus contactos sin conocerlos, cómo se recupera un PIN sin que el servidor lo adivine y cómo ocultar tu número con un username.

Registro y bloqueo de registro

Para crear una cuenta, Signal envía un código por SMS o llamada al número indicado; quien lo recibe prueba que controla ese número. Este paso existe sobre todo contra el spam: los números cuestan dinero y escasean. Su consecuencia es que Signal siempre ha estado ligado a tu número, y que quien secuestre tu número en la operadora podría registrarlo en otro teléfono. El **bloqueo de registro** exige además tu PIN para registrar el número en un dispositivo nuevo, y cierra esa vía.

Descubrimiento privado de contactos

Para decirte cuáles de tus contactos usan Signal sin que el servidor vea tu agenda, tu aplicación envía los números cifrados hacia un **enclave de hardware seguro** (Intel SGX) del servicio. La aplicación comprueba mediante atestación remota que el enclave ejecuta el código publicado, y el enclave consulta su lista de forma que ni siquiera el operador del servidor puede ver qué números se consultaron ni cuáles coinciden.

PIN y recuperación segura de valores (SVR)

Un PIN de 4 a 6 dígitos es fácil de adivinar. SVR permite usarlo igualmente sin dejar que nadie lo ataque por fuerza bruta:

1. El cliente pasa el PIN por **Argon2**, una función lenta que encarece cada intento, y obtiene un valor de 32 bytes del que deriva una ficha de autenticación `auth` y un secreto `c1`.
2. Genera además 256 bits aleatorios `c2`, que forman junto a `c1` la clave maestra de la que salen las claves con que se cifran en la nube tus datos de cuenta.
3. `c2` se guarda en un enclave del servidor, asociado a `auth`. El enclave solo lo entrega ante la `auth` correcta y **cuenta los intentos fallidos**: tras unos pocos (cinco en el ejemplo de Signal) deja de responder.

4. El estado vive solo en memoria cifrada y se replica entre enclaves con el protocolo de consenso Raft, de modo que nadie pueda reiniciar el contador reiniciando el almacenamiento.

Como `c2` es aleatorio, la clave maestra no se puede atacar a partir del PIN solo, y como el enclave limita los intentos, un atacante con acceso al servidor tampoco tiene intentos ilimitados. Versiones posteriores (SVR3, descrita en un artículo académico de 2024) reparten el secreto entre varios entornos de hardware protegido, de manera que comprometer uno solo no basta. Elegir un PIN largo, de letras y números, refuerza todo el sistema.

Usernames y privacidad del número

Desde 2024 puedes crear un **username** (`nombre.42`: un apodo más un número obligatorio que evita que alguien acapare nombres) y compartirlo en lugar del número de teléfono. Los dos ajustes que lo acompañan son ¿quién ve mi número? y ¿quién puede encontrarme por mi número?; en las cuentas nuevas el número ya no se muestra por defecto. Los usernames no se guardan en claro en el servidor: se almacenan en forma de hash y se resuelven con pruebas criptográficas, y el enlace o código QR del username contiene un secreto que impide recorrerlos. Puedes cambiarlo o borrarlo en cualquier momento para romper los enlaces que se hayan compartido.

Matiz importante. El username no hace anónima la cuenta: Signal sigue conociendo el número y, ante una orden judicial sobre un username, podría entregar ese número junto con la fecha de creación y la última conexión.

Cuentas sin número: en desarrollo

La dependencia del número es la crítica más repetida. El código para registrarse sin número de teléfono ya aparece en el código de Signal, pero aún no es utilizable y el calendario es provisional. La dificultad es el spam: una posibilidad que se ha mencionado es cobrar una pequeña cuota por cuenta sin número para que crear miles resulte caro. Hasta que exista, la vinculación con un número sigue siendo un límite real de la privacidad de Signal.

CAPÍTULO 14

Copias de seguridad seguras

Las copias seguras de Signal guardan tu historial en la nube de forma que solo tu clave de recuperación puede abrirlas.

El problema

Durante años Signal no ofreció copias en la nube porque son un riesgo: lo que está guardado en un servidor puede filtrarse o ser exigido por una autoridad. A cambio, perder o romper el teléfono significaba perder el historial. Las copias seguras, que se activan voluntariamente, intentan evitar esa disyuntiva.

Cómo funcionan

1. Al activarlas, el dispositivo genera una **clave de recuperación de 64 caracteres**. Esa clave nunca se comparte con Signal.
2. De ella se derivan con un KDF las claves con que se cifra la copia y el identificador bajo el que se guarda.
3. Cada día el dispositivo crea un archivo nuevo con tus mensajes, lo cifra y lo sube, y este sustituye al del día anterior.
4. Para restaurar en un teléfono nuevo se introduce la clave de recuperación, que permite descargar y descifrar la copia.

El servidor solo ve un bloque de bytes cifrado cuyo tamaño y fecha de actualización conoce; ni Signal ni nadie sin la clave pueden leerlo. La consecuencia inversa es dura: **si pierdes la clave de recuperación, nadie puede recuperar la copia**, ni siquiera Signal.

Planes

Las copias llegaron primero a Android y después a iOS. Hay un nivel gratuito, que cubre el texto y un espacio limitado de archivos recientes (en torno a 100 MiB), y uno de pago de hasta 100 GB por unos dos dólares al mes, que además ayuda a financiar a la Fundación. Signal ha dicho que la misma tecnología servirá de base para más opciones, entre ellas transferir el historial entre Android, iPhone y escritorio.

Qué riesgos añaden

Una copia es siempre una segunda superficie de ataque: quien consiga tu clave de recuperación y el archivo puede leer tu historial pasado. Por eso conviene guardarla fuera del teléfono (en un gestor de contraseñas o en papel) y valorar si la comodidad de poder restaurar compensa el riesgo. Si tu modelo de amenaza es alto, puedes dejar las copias desactivadas y usar mensajes temporales.

CAPÍTULO 15

Modelo de confianza y límites

Signal reduce mucho lo que hay que creer, pero no a cero: este capítulo recoge en quién y en qué sigues confiando al usarlo.

Quién está detrás

Signal lo mantiene la **Signal Foundation**, una organización sin ánimo de lucro fundada en 2018 con una aportación inicial de 50 millones de dólares de Brian Acton, cofundador de WhatsApp. Se financia con donaciones y, ahora, con el nivel de pago de las copias de seguridad. Moxie Marlinspike, creador del protocolo junto con Trevor Perrin, dejó la dirección en 2022; la preside Meredith Whittaker. El código de las aplicaciones y del servidor es abierto, y los protocolos están documentados en detalle.

A quién le confias qué

PARTE	QUÉ PUEDE HACER SI FALLA O ES HOSTIL
Tu dispositivo	Leerlo todo: es uno de los extremos
Dispositivos de tu contacto	Leer, copiar o filtrar lo que le envíes
Sistema operativo y fabricante	Tocar la pantalla, el teclado y la memoria de las apps
Servidor de Signal	Entregar una clave falsa (se detecta verificando), ver metadatos, denegar el servicio
Enclaves de hardware	Un fallo del fabricante podría debilitar el descubrimiento de contactos y el PIN
Tu operadora	Ver que usas Signal y secuestrar el número (el bloqueo de registro lo frena)

Qué puedes comprobar y qué no

Puedes leer el protocolo y su código, y el protocolo ha sido analizado en publicaciones académicas y verificado formalmente en parte. Lo que no puedes comprobar por tu cuenta es que el servidor que usas ejecuta exactamente el código publicado; para los enclaves, la atestación remota cubre buena parte de ese hueco, pero para el resto es un acto de confianza. La verificación de claves (capítulo 12) es lo que reduce la confianza en el servidor.

Los límites, uno a uno

- **El número de teléfono.** Sigue siendo el identificador de la cuenta, aunque los usernames lo oculten a otros usuarios. Las cuentas sin número están en desarrollo.
- **Un servidor central.** Signal no es federado a propósito (permite actualizar el protocolo rápido), pero implica un punto único de fallo y de presión legal, y que la disponibilidad depende de un solo operador.
- **La dirección IP y la correlación.** El servidor ve desde dónde te conectas; Sealed Sender no lo evita (capítulo 9).
- **Los extremos comprometidos.** Spyware, un teléfono desbloqueado en manos ajenas, una vista previa de notificaciones en pantalla bloqueada, una copia del sistema en la nube que incluya datos de la app. El cifrado de extremo a extremo no cubre nada de esto.
- **Los dispositivos vinculados.** Cada uno amplía la superficie de ataque, y una vinculación por QR engañosa da acceso a la cuenta.
- **Las notificaciones push.** Pasan por los servicios de Apple y Google. No incluyen el texto, pero revelan el momento en que llega algo.
- **El contexto legal.** El servicio se opera desde Estados Unidos. Signal ha declarado públicamente que prefiere dejar de operar en un país antes que debilitar el cifrado, lo que muestra que las leyes que obligan a escanear los mensajes son un riesgo real para el servicio, no una hipótesis.

CAPÍTULO 16

Buenas prácticas

La mayor parte de las ganancias de privacidad con Signal vienen de unos pocos ajustes y

hábitos, ordenados aquí de mayor a menor impacto.

1. **Mantén actualizados la aplicación y el sistema operativo.** La protección poscuántica y las correcciones de seguridad llegan por actualización; instala siempre desde la fuente oficial.
2. **Pon un PIN largo y activa el bloqueo de registro.** Con letras y números si es posible. Protege tu cuenta ante el robo del número (capítulo 13).
3. **Bloquea la pantalla y la aplicación,** y desactiva el contenido en las notificaciones.
4. **Crea un username y oculta tu número** en los ajustes de privacidad, y limita quién puede encontrarte por número.
5. **Verifica las claves de los contactos importantes,** con número de seguridad o con la verificación automática.
6. **Usa mensajes que desaparecen** en las conversaciones sensibles, sabiendo que es un acuerdo entre aplicaciones.
7. **Revisa tus dispositivos vinculados** de vez en cuando y no escanees nunca un QR de vinculación que no hayas generado tú.
8. **Si te preocupa tu IP,** activa "Retransmitir siempre las llamadas" y usa una VPN o Tor en la red desde la que te conectas.
9. **Decide con calma sobre las copias.** Si las activas, guarda la clave de recuperación fuera del teléfono.
10. **Desconfía de quien te pida el código de registro o tu PIN.** Signal no los pide nunca.

Qué combinar según el adversario

QUIERES PROTEGERTE DE...	QUÉ HACER
Quien intercepta la red	Nada especial: el cifrado ya lo cubre
Que alguien tome tu número	PIN largo y bloqueo de registro
Que compartan tu número con desconocidos	Username y ajustes de número
Un servidor que sustituya claves	Verificar números de seguridad
Que se sepa desde dónde te conectas	VPN o Tor, retransmisión de llamadas
Un teléfono robado o desbloqueado	Bloqueo de pantalla, mensajes temporales, sin vistas previas
Spyware dirigido	Ningún ajuste de Signal basta: actualiza, minimiza lo que guardas y busca ayuda especializada

CAPÍTULO 17

Fuentes y lecturas

La guía se apoya en las especificaciones y artículos oficiales de Signal y en las publicaciones que se citan a continuación. Para los detalles que cambian con el tiempo (despliegue de SPQR, cuentas sin número, precios), consulta siempre la fuente original.

Especificaciones y artículos de Signal

- Documentación y especificaciones de Signal: X3DH, Double Ratchet y Sesame.
- La especificación de PQXDH
- Signal Protocol and Post-Quantum Ratchets (SPQR)
- Technology preview for secure value recovery
- Introducing Automatic Key Verification

Artículos académicos y técnicos

- Secret Key Recovery in a Global-Scale End-to-End Encryption System (SVR3)
- PQShield: Diving into Signal's New Post-Quantum Protocol
- Migrating Protocols to the Post-Quantum Setting: The Case of Signal's Double Ratchet

Prensa especializada

- TechCrunch: Signal now lets you keep your phone number private
- The Intercept: usernames y privacidad del número
- BleepingComputer: copias de seguridad seguras
- Daring Fireball: copias seguras en iOS
- CyberInsider: Triple Ratchet
- Privacy Guides: cuentas sin número en la beta de Android