

Signal Messenger Prä-Quanten Kryptographie

Präsentation im Modul „C179 - Kryptologie“

2026-01-19

Eric Behrendt, Daniel Kretschmer, Mose Schmiedel

HTWK Leipzig
University of Applied Sciences Leipzig

Überblick

1. Schutzmaßnahmen
2. Sessionverwaltung
3. Verschlüsselung
4. Gruppenchats
5. Anwendungsbeispiel



[krypto-ws25/signal-pre-quantum](#)

Schutzmaßnahmen

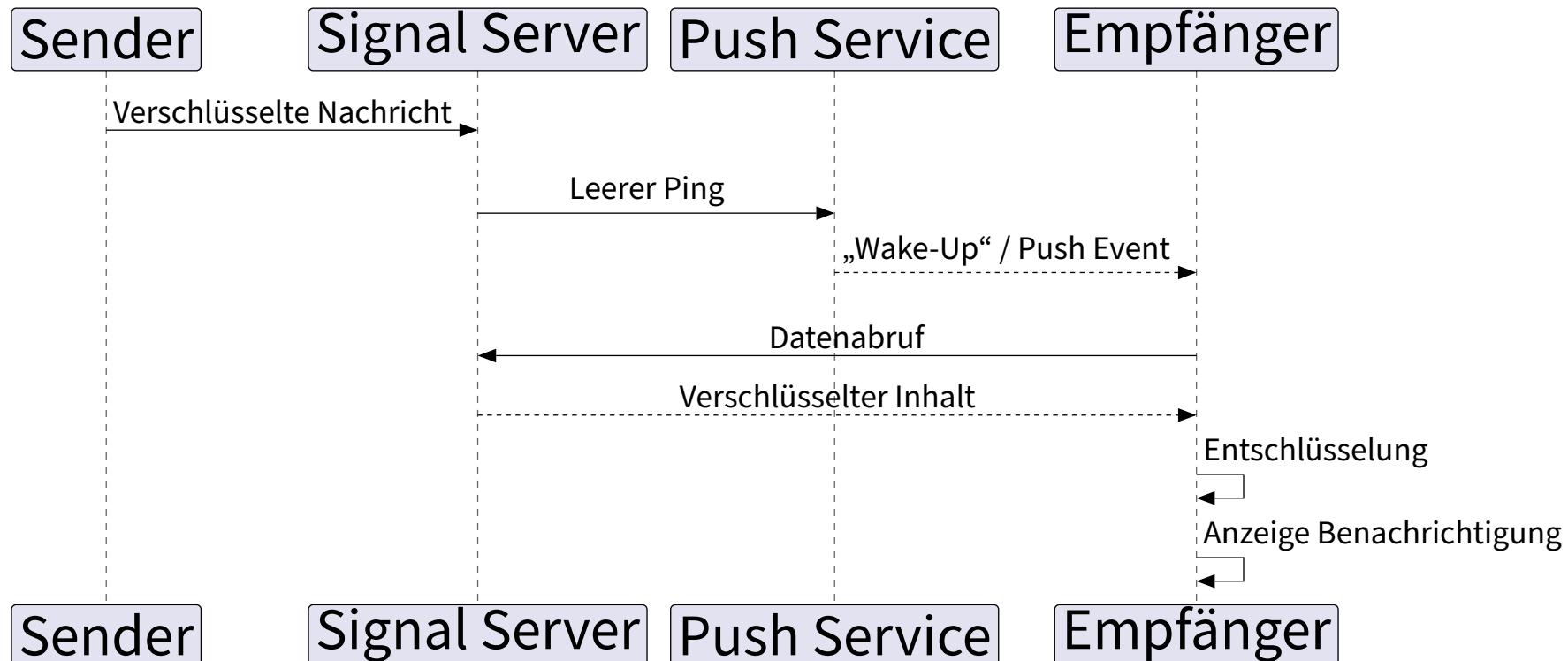
Routing

Sealed Sender

- Absender wird in der Nachricht verschlüsselt
- Minimierung der sichtbaren Metadaten
- Spamvermeidung durch Delivery Token



Push Notifications



Attachments

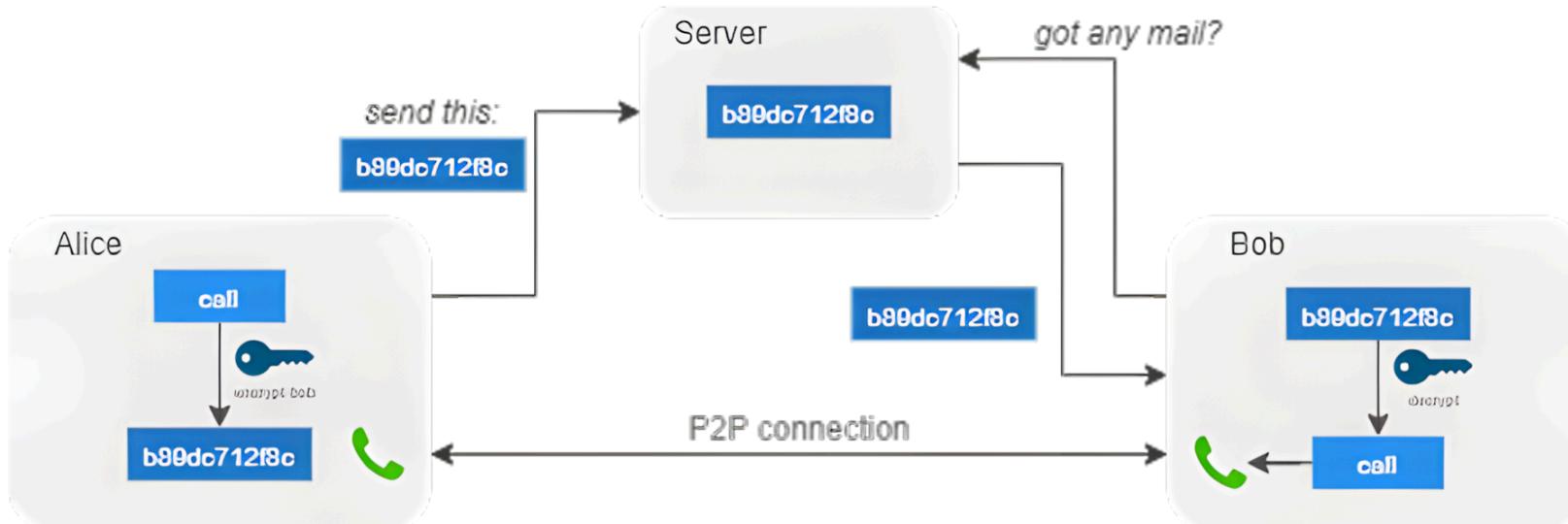
Blob Store

- Anhänge werden nicht über die Nachrichten-Warteschlange übertragen
- Übertragung als verschlüsseltes Objekt mittels weiterem Server
- Schlüsselübertragung über Nachrichtenweg

Calling Infrastructure

RingRTC

- Erweiterung von WebRTC
- TURN-Relay auf Wunsch oder bei Bedarf



Private Contact Discovery

SGX Enclaves

- Kontaktabgleich auf Server
- Nutzung Intel SGX Software Guard Extensions

Secure Backups

- Privacy-First „Zero-Knowledge“ Architecture
 - Unlinked Data
 - No Tracking
- User-Controlled 64-Character Recovery Key
- Layered Encryption & Padding
- Handling of Disappearing Messages
- Opt-In
- Cross-Platform Wiederherstellung

Sessionverwaltung - Sesame

Kernfunktionen

Multi-Device Management

- UserRecords und DeviceRecords
 - Hierarchie zum Speichern der Encryption-Sessions
- Self-Sending
 - Nachrichten werden auch eigenen Geräte geschickt

Session Convergence

- Abstimmung über aktive Session
 - Aktive Session für das Senden von Nachrichten
- Receiving & Switching

Kernfunktionen

Handling Stale Devices und Messages

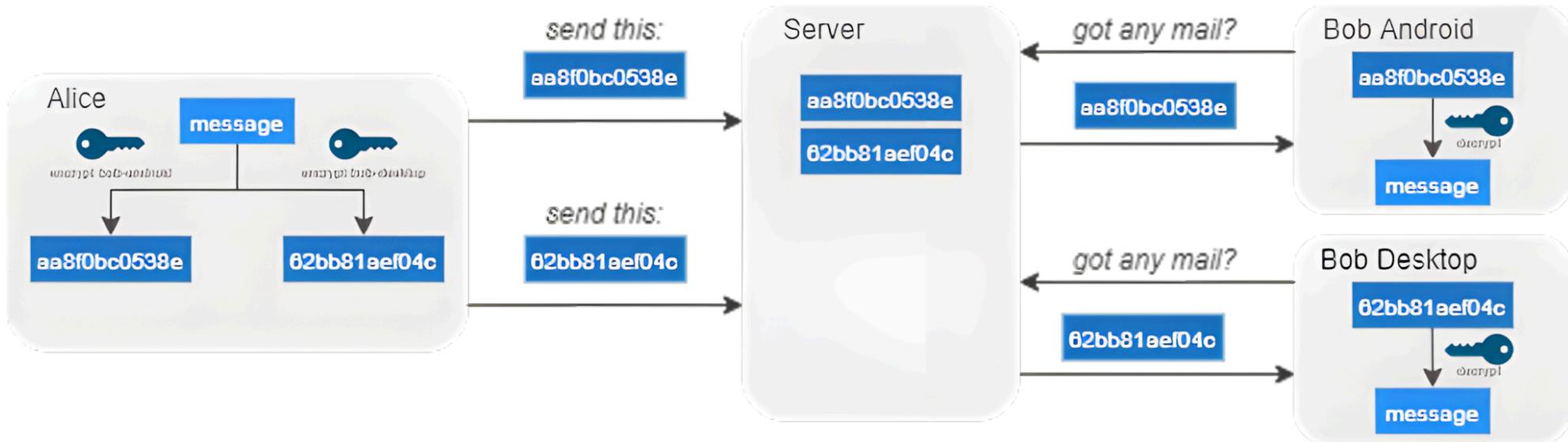
- Stale Records
 - Behandlung von abgelaufenen Records

Identity Key Models

- Per-User Identity Keys
- Per-Device Identity Keys

Server Interaction

- Mailboxes
- Unreliable Network Resilience



Verschlüsselung – X3DH

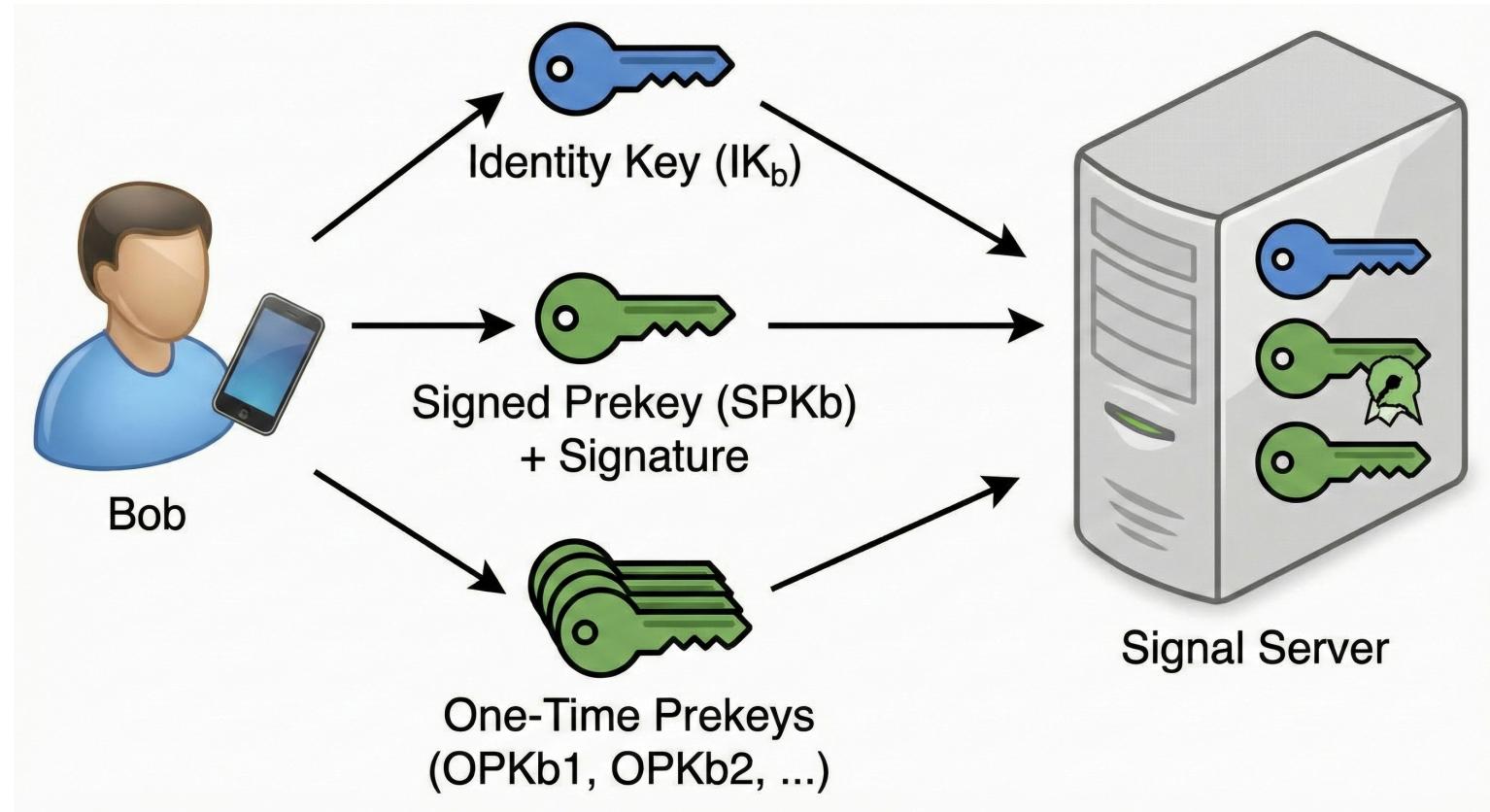
Allgemeine Informationen

- **Überblick**
 - ▶ „Extended Triple Diffie-Hellman“
 - ▶ Etabliert einen gemeinsamen geheimen Schlüssel zwischen zwei Parteien
 - ▶ Bietet gegenseitige Authentifizierung mittels öffentlicher Schlüssel
 - ▶ Entwickelt für asynchrone Umgebungen (z. B. wenn eine Partei offline ist)
 - ▶ Bietet Forward Secrecy (Folgenlosigkeit) und kryptografische Abstreitbarkeit

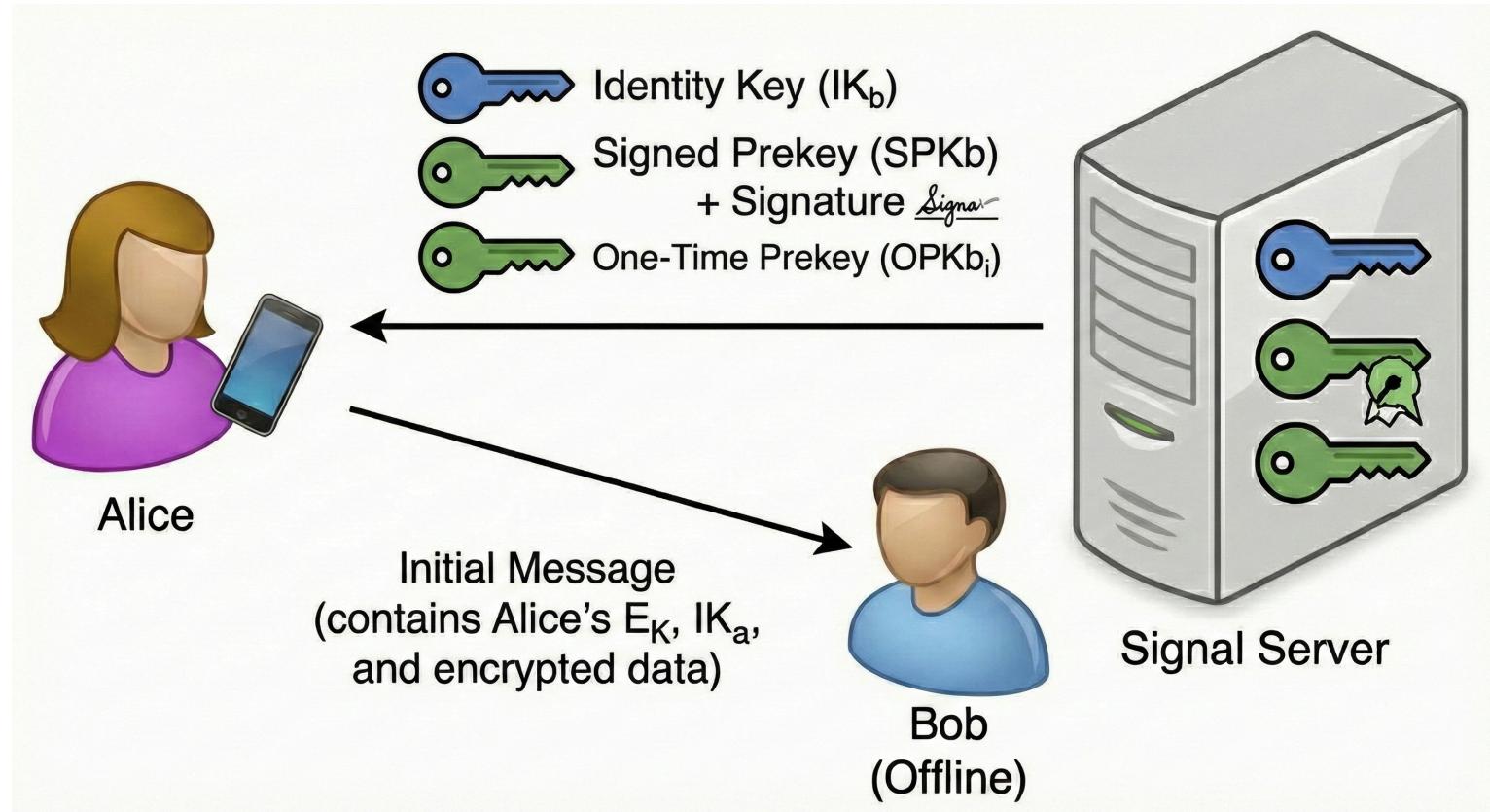
- **Schlüsselarten**

Bezeichnung	Symbol	Beschreibung
Identitätsschlüssel	IK_A, IK_B	Langzeitschlüssel
Ephemerer Schlüssel	EK_A	Wird für einen einzelnen Durchlauf generiert
Signierter Prekey	SPK_B	Wird von Bob periodisch aktualisiert
Einmal-Prekey	OPK_B	Wird einmalig verwendet und dann gelöscht

Veröffentlichen der Schlüssel



Abrufen des Schlüsselbündels



Verarbeiten des Schlüsselbundes

- **Schlüsselberechnung (Alice)**
 - Alice generiert ein ephemeres Schlüsselpaar (EK_A)
 - Alice berechnet die DH-Ausgaben:
 - $DH1 = DH(IK_A, SPK_B)$
 - $DH2 = DH(EK_A, IK_B)$
 - $DH3 = DH(EK_A, SPK_B)$
 - Optional: $DH4 = DH(EK_A, OPK_B)$
- **Ableitung des gemeinsamen Geheimnisses (SK)**
 - $SK = KDF(DH1 \parallel DH2 \parallel DH3 \parallel DH4)$

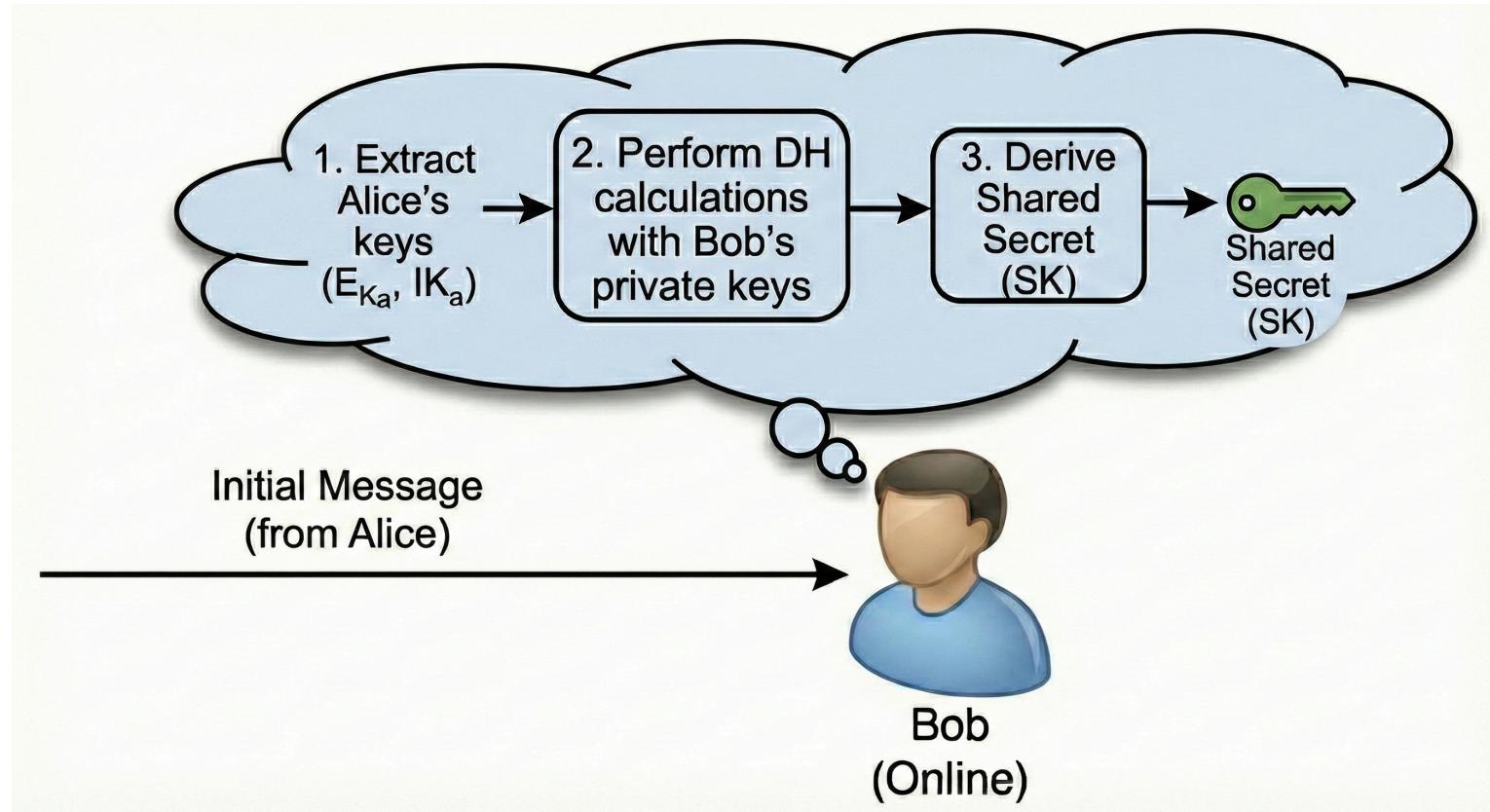
- **Assoziierte Daten (AD)**

- Konstruiert für die Authentifizierung
- $AD = \text{Encode}(IK_A) \parallel \text{Encode}(IK_B)$
- Optionale Identifikationsdaten (Benutzernamen etc.) können angehängt werden

- **Senden der Nachricht**

- Alice sendet Initialnachricht an Bob:
 - Alices Identitätsschlüssel (IK_A)
 - Alices Ephemerer Schlüssel (EK_A)
 - Identifikatoren für die verwendeten Prekeys
 - Ein mit SK (und AD) verschlüsselter initialer Geheimtext

Verarbeiten der Initialnachricht



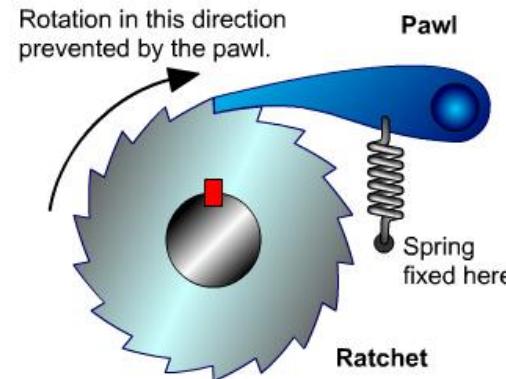
Verschlüsselung – Double Ratchet

Warum wird diese benötigt?

- **Ende-zu-Ende-Sicherheit:** Vertraulichkeit zwischen zwei Parteien
- **Forward Secrecy:** Schutz vergangener Nachrichten bei Schlüsseldiebstahl
- **Post-Compromise Security:** Automatische „Selbstheilung“ nach Kompromittierung

Warum der Name „Ratchet“ (Ratsche)?

- **Einweg-Prinzip:** Bewegung nur vorwärts (wie mechanische Ratsche)
- **Unumkehrbarkeit:** Keine Rückrechnung auf alte Schlüssel möglich
- **Sofortige Löschung:** Nachrichtenschlüssel werden nach Gebrauch vernichtet



Warum „Double“ (Doppel)?

Kombination zweier Mechanismen:

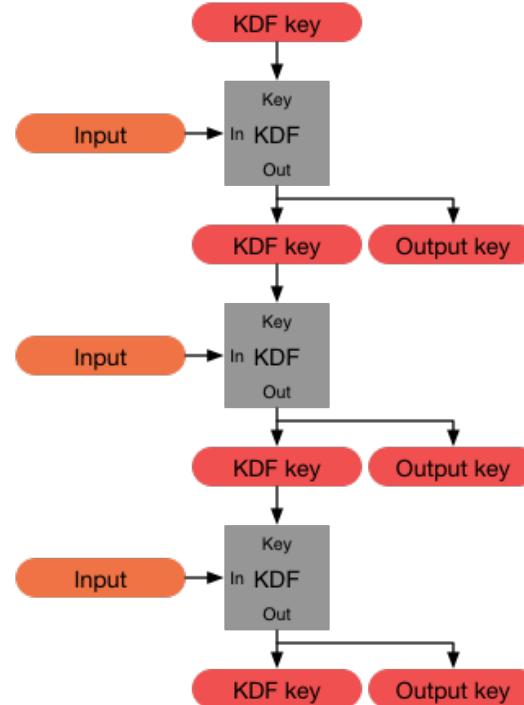
1. **Diffie-Hellman-Ratchet:**

- Auslösung bei Generierung neuer privater Schlüssel
- Erneuert das Schlüsselmaterial („Selbstheilung“)

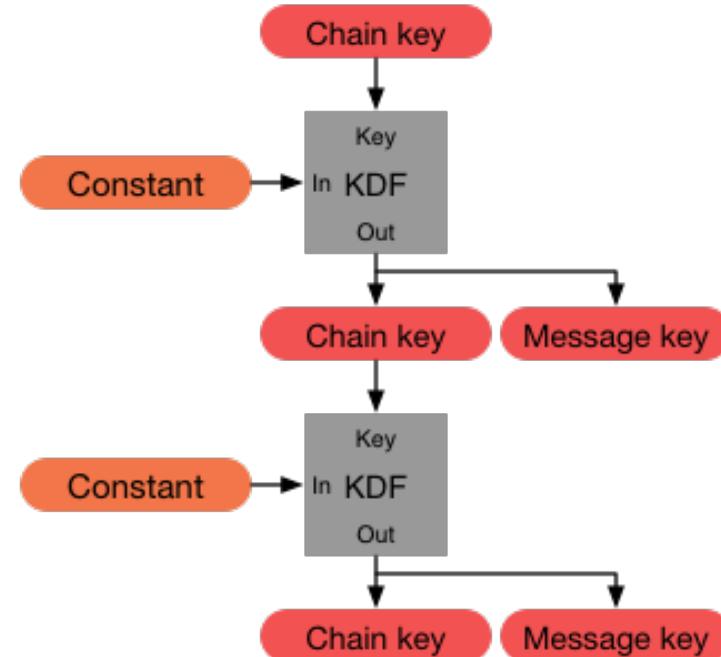
2. **Symmetrische Ratchet (KDF):**

- Auslösung pro Nachricht
- Schnelle Generierung neuer Schlüssel

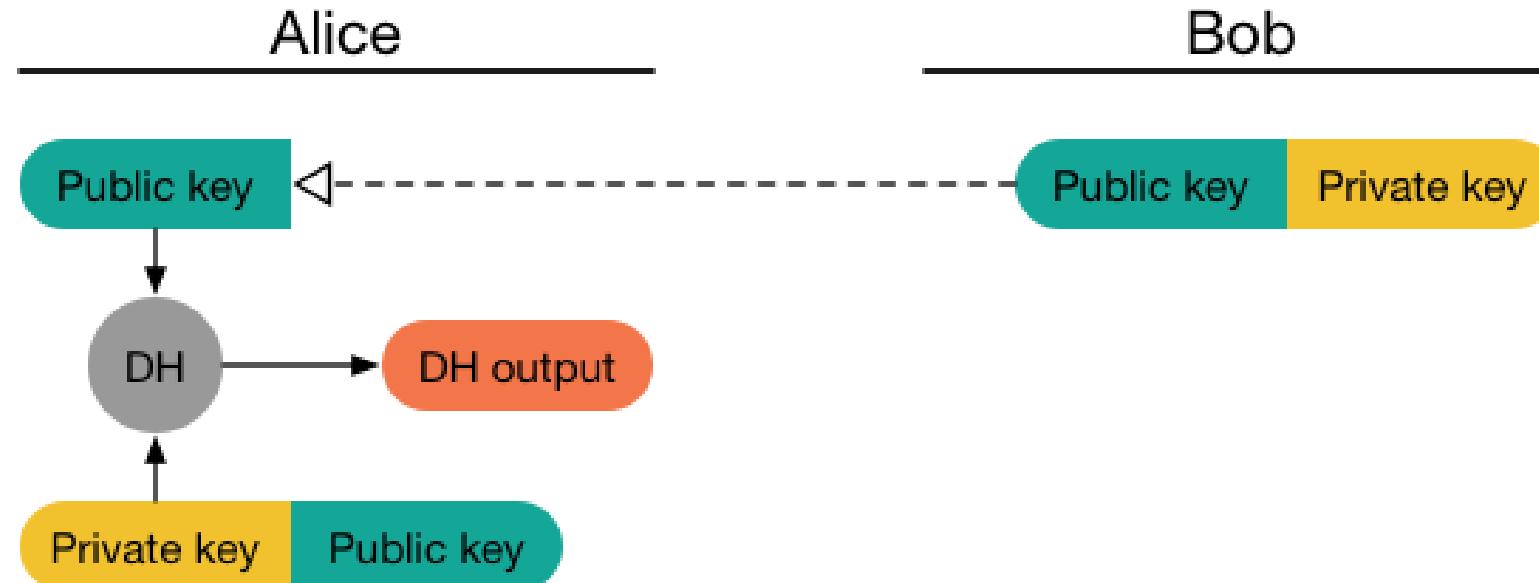
KDF-Ketten (Key Derivation Function)



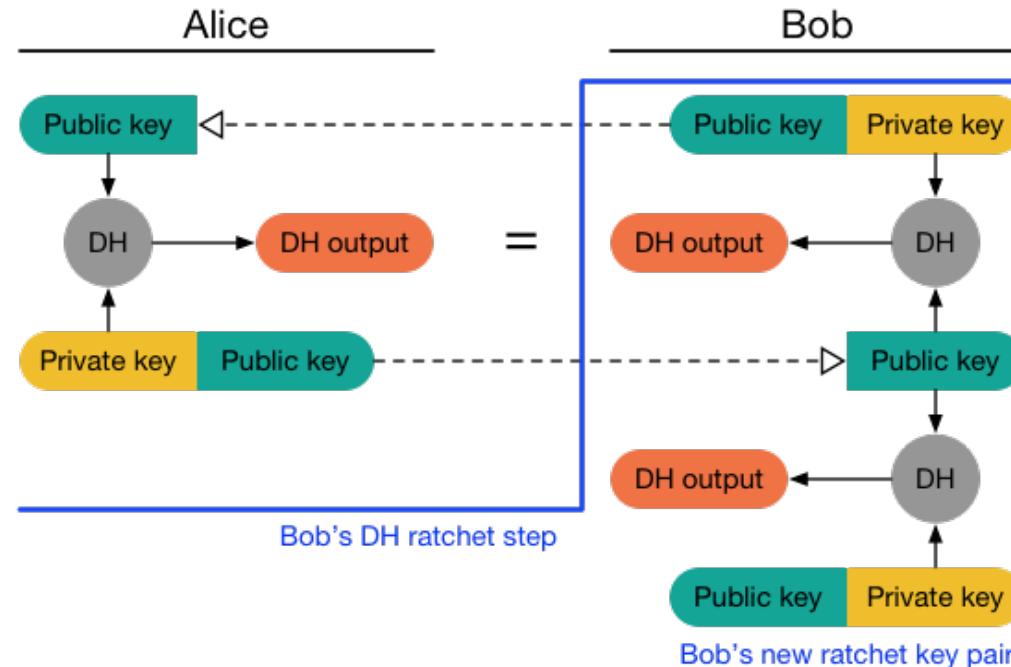
Symmetrische Schlüssel-Ratchet



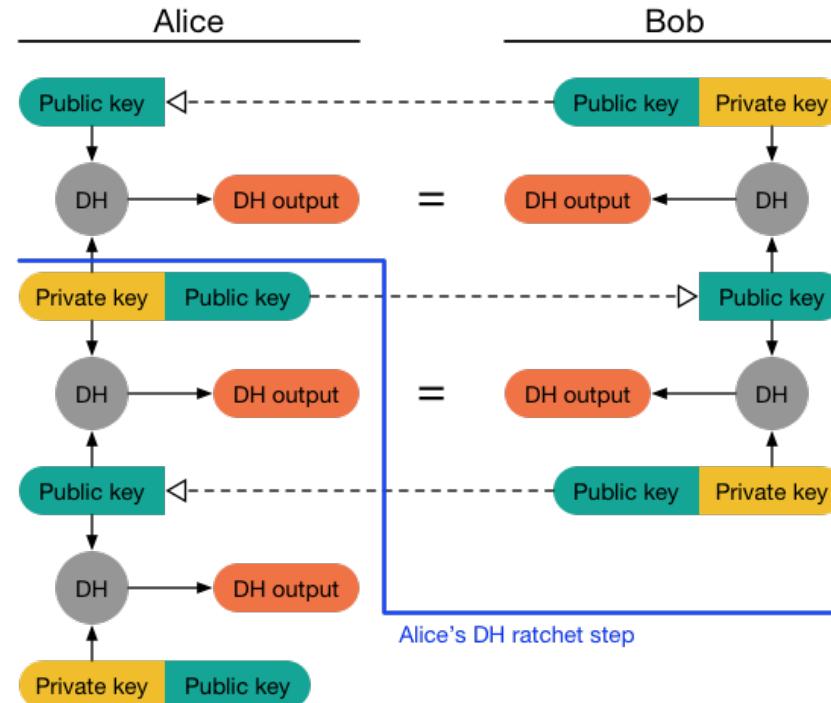
Diffie-Hellman-Ratchet



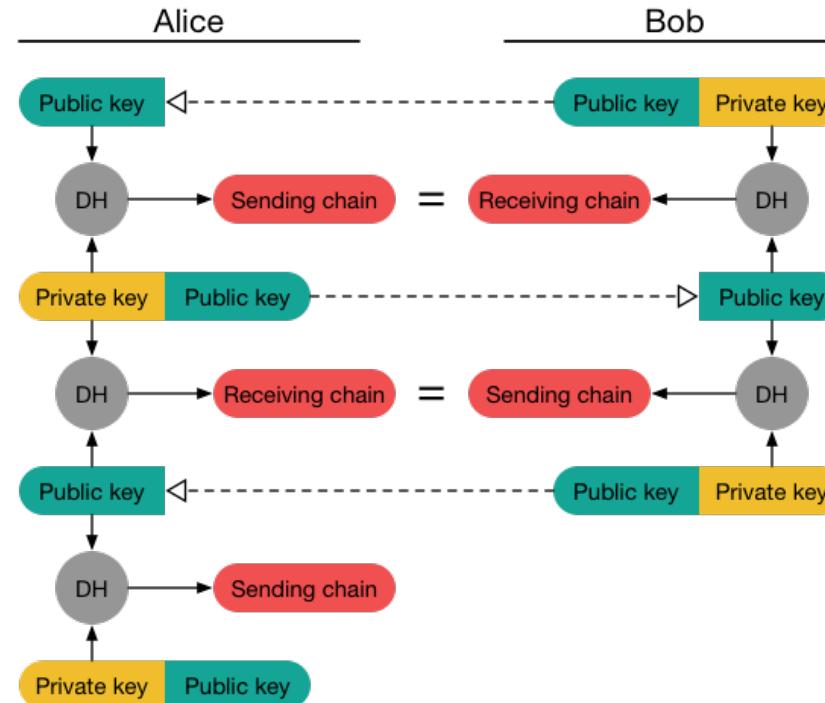
Diffie-Hellman-Ratchet



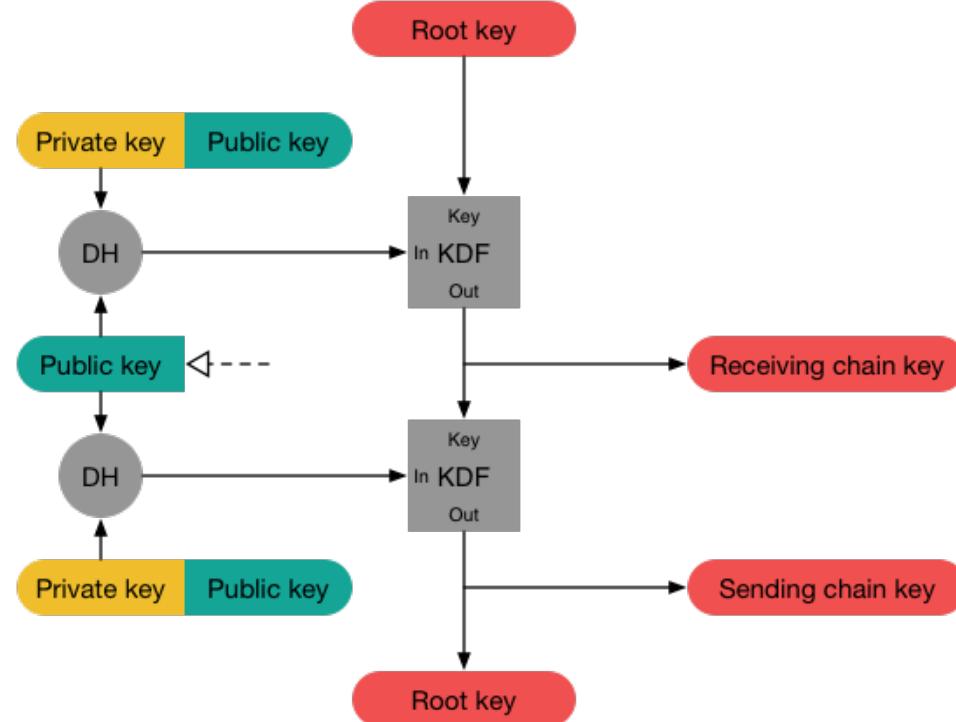
Diffie-Hellman-Ratchet



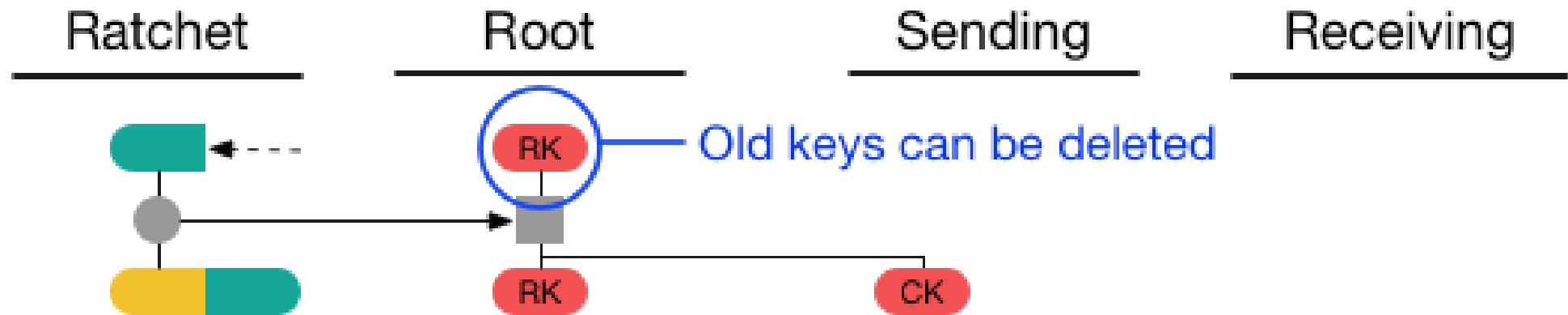
Diffie-Hellman-Ratchet



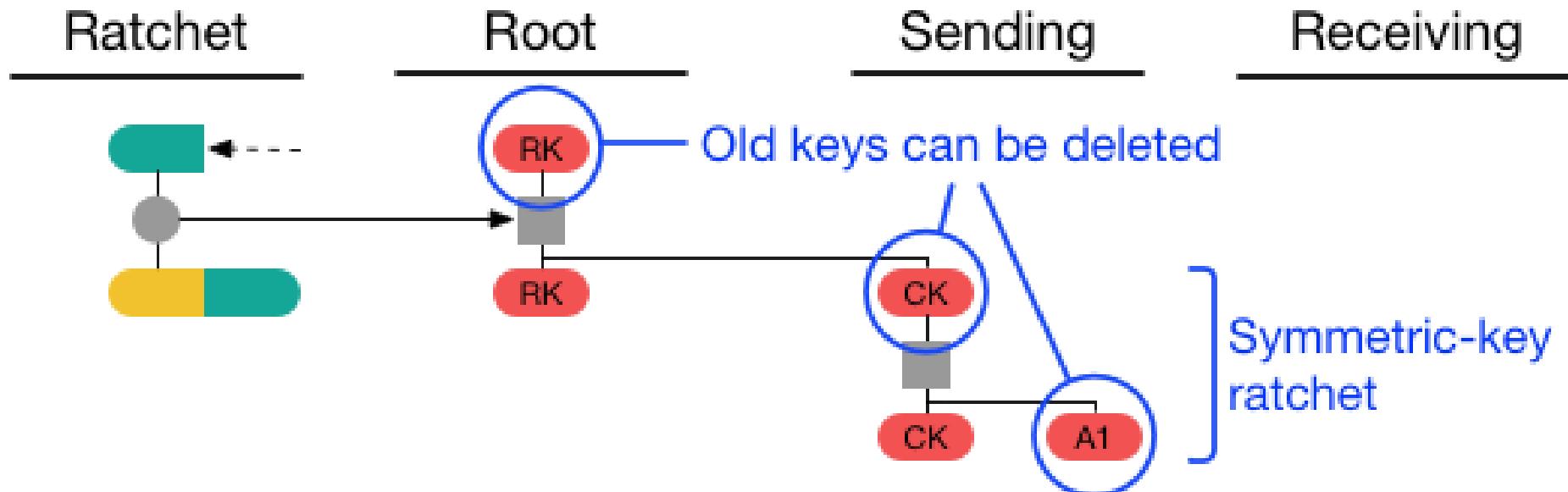
Diffie-Hellman-Ratchet



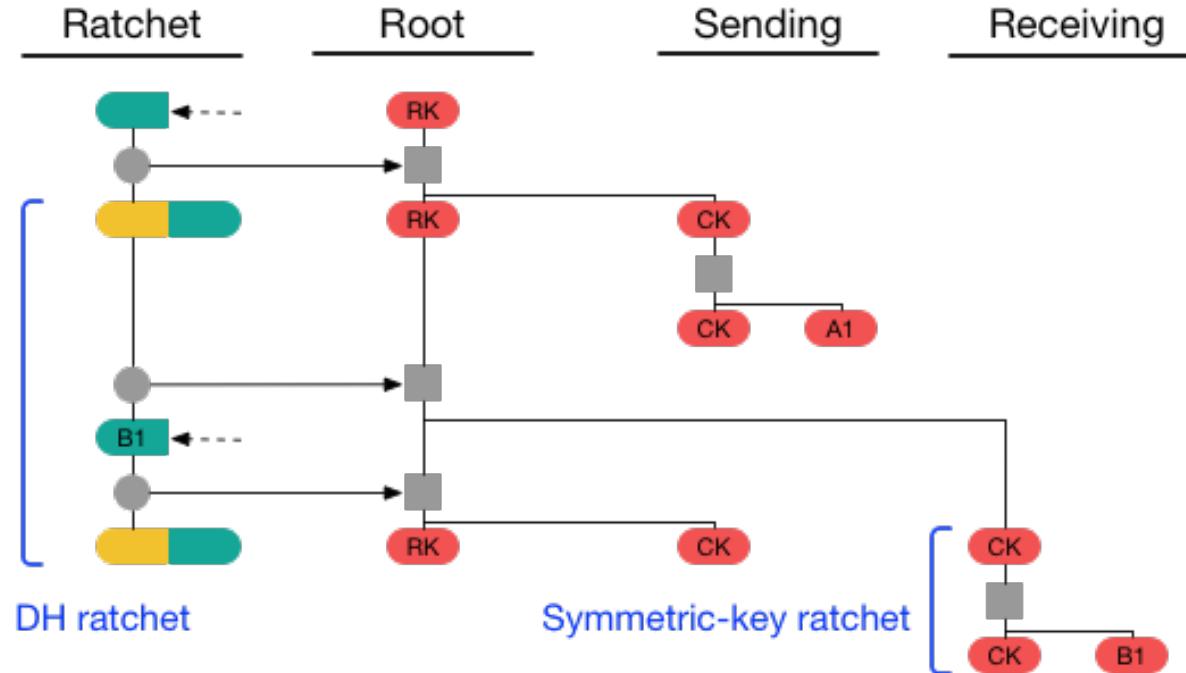
Double Ratchet



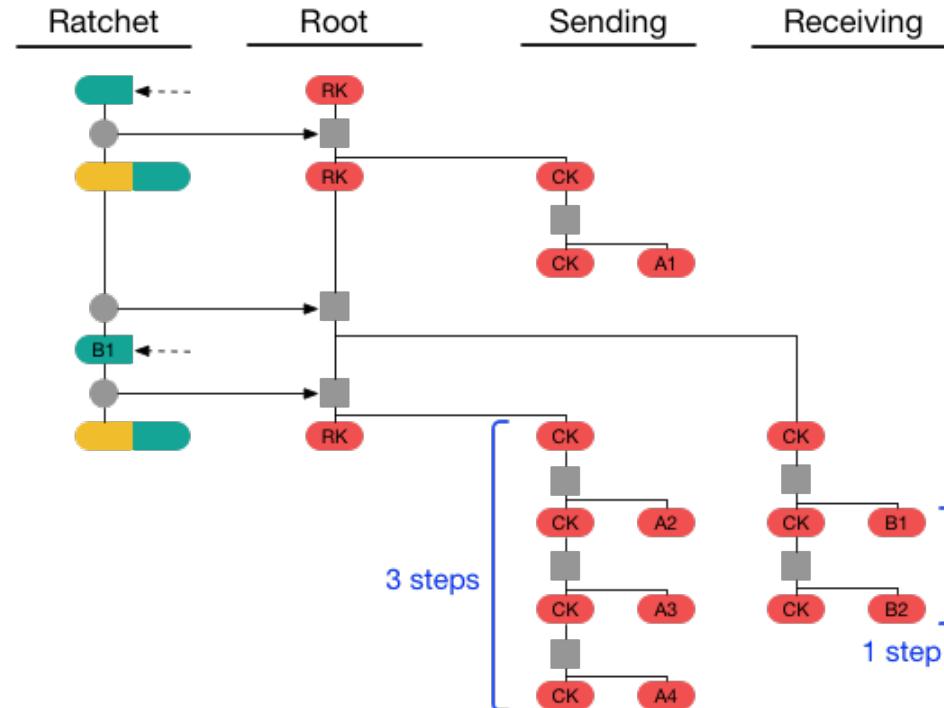
Double Ratchet



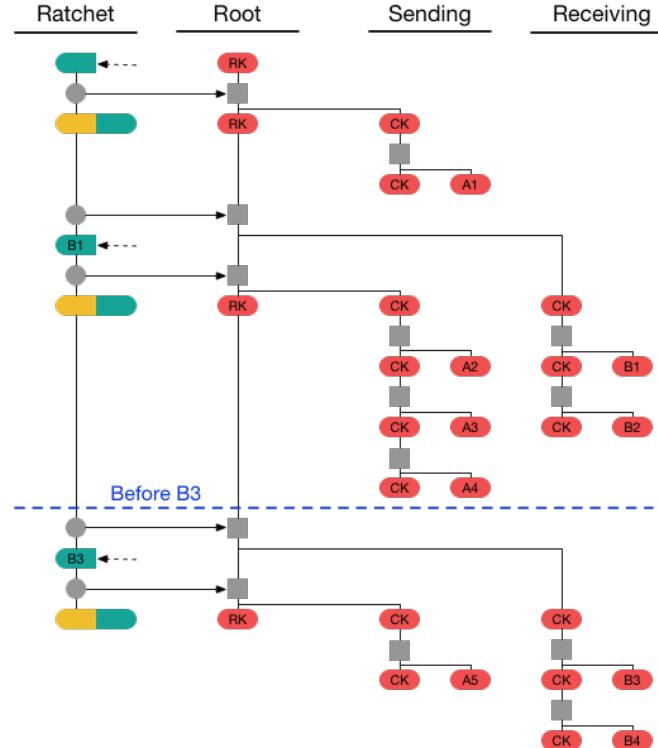
Double Ratchet



Double Ratchet



Double Ratchet



Gruppenchats

Allgemeines (1/2)

Gruppe	
Mitglieder	$C_M(U_i)$
Profilschlüssel	$C_{PK}(P(U_i))$
Profile	$C_P(P(U_i))$

verschlüsselte Mitgliederliste (UIDs)

verschlüsselte Schlüssel für Profildaten

verschlüsselte Profildaten zu ggb. UID

Allgemeines (2/2)

- Problem: Server soll Gruppenmetadaten (Mitgliederliste, Beschreibung, Bild, usw.) nicht lesen und schreiben können
- früher: jeder Nutzer verwaltet lokal Gruppenmetadaten
- heute: Gruppenmetadaten liegen verschlüsselt auf dem Server mit geteiltem Schlüssel → **GroupMasterKey**
- Gruppennachrichten werden Peer-to-peer verschlüsselt und versendet
- **Wie kann Server Veränderungen der Gruppenmetadaten authentifizieren?**

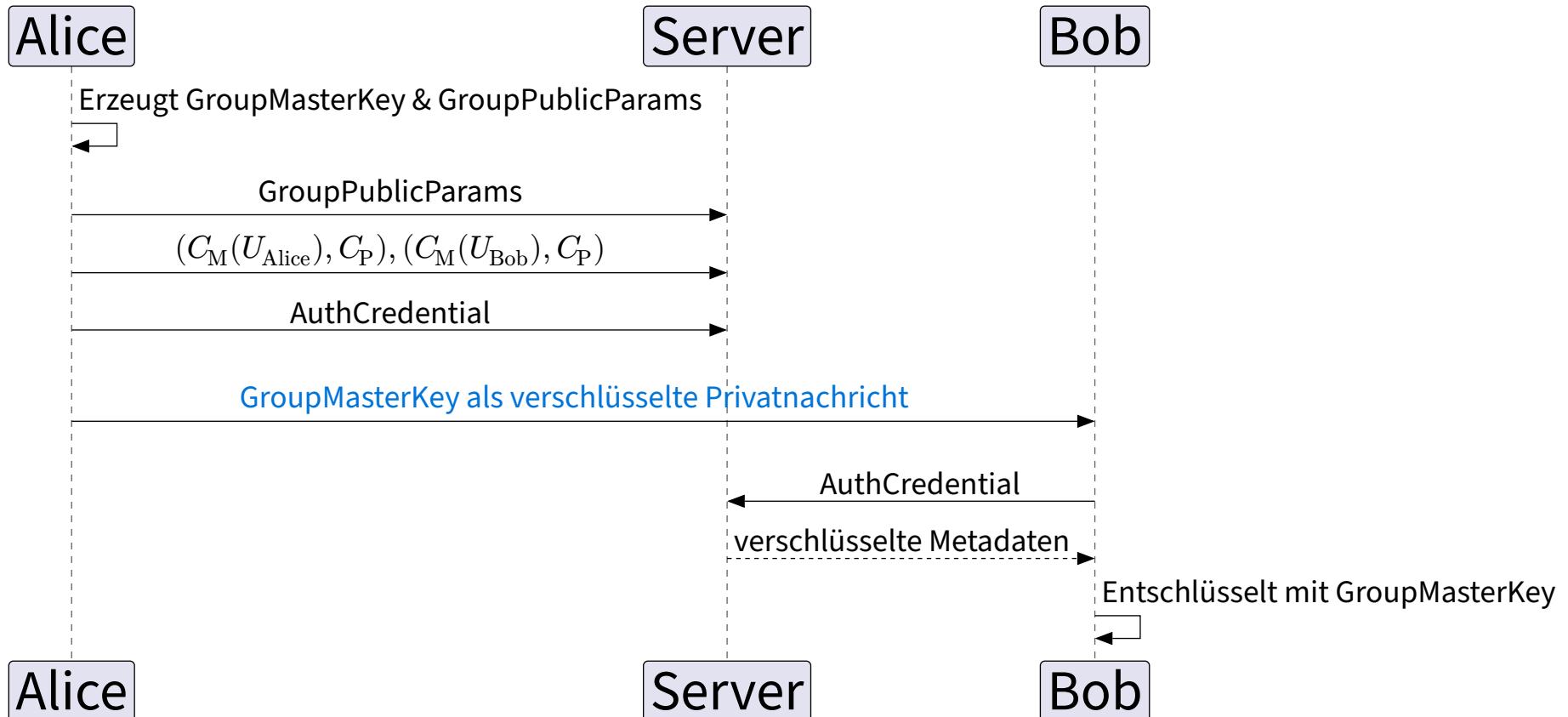
Anonymous Credentials (1/2)

- UserID U liegt nur verschlüsselt $C(U)$ auf dem Server → kann vom Server nur in diesem Zustand verwendet werden
- User muss zeigen, dass er der Besitzer von $C(U)$ ist ohne U dem Server zu verraten → Zero-Knowledge proof
- außerdem müssen Kollisionen $C(U_1) = C(U_2)$ zwingend verhindert werden um Spoofing zu verhindern
- neue Verschlüsselung keyed-verification anonymous credentials (KVAC)

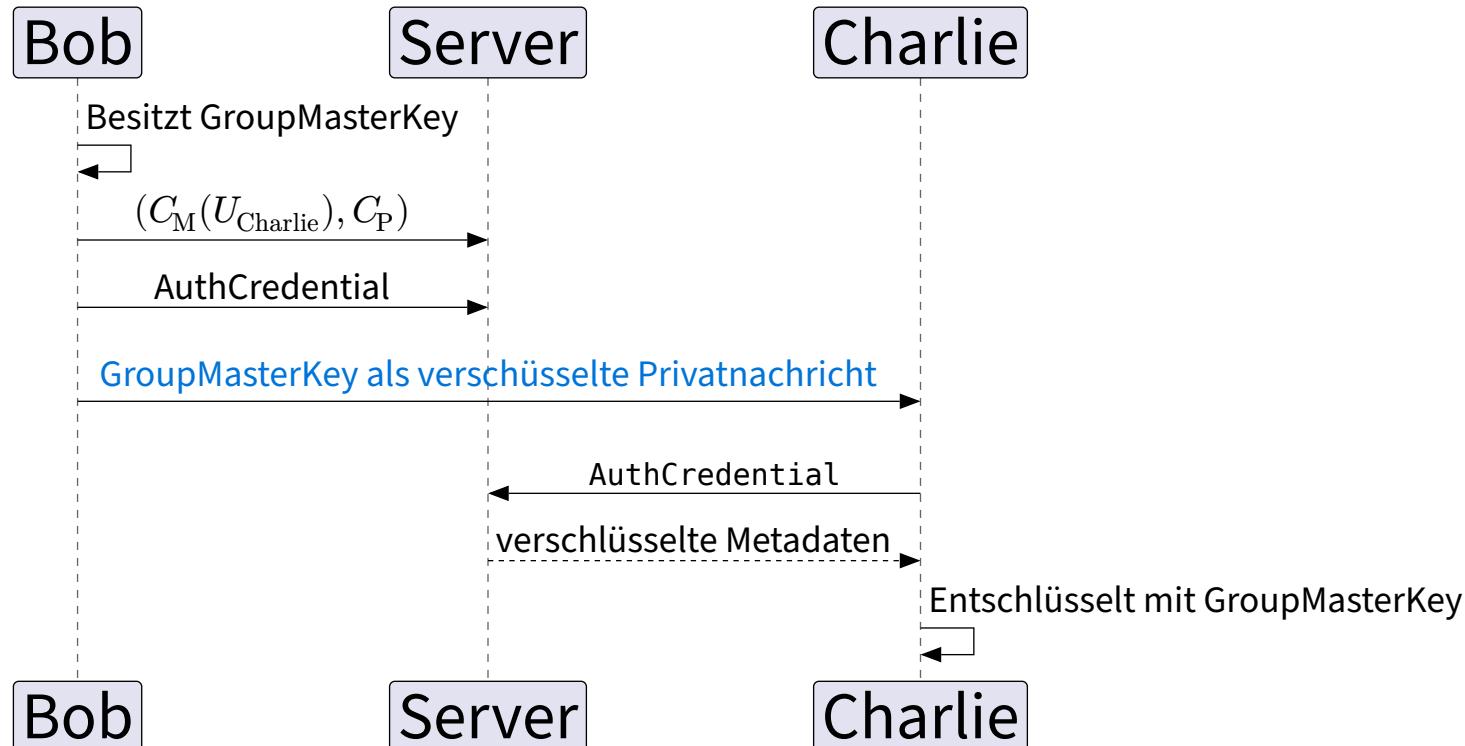
Anonymous Credentials (2/2)

- GroupMemberEntry ist UID mit deterministischer Verschlüsselung
- Server erzeugt regelmäßig AuthCredentials pro UID → müssen von User geholt werden
- User kann mit AuthCredential und eigener UID verschlüsselten GroupMemberEntry wieder erzeugen → Authentifizierung

Beispiel: Gruppe erstellen



Beispiel: Gruppenmitglied hinzufügen



Zusammenfassung

- Sesame
- RingRTC
- X3DH
- Double Ratchet
- Anonymous Credentials
- Secure Backups

Quellen

<https://signal.org/docs/specifications/doubleratchet/>

- [1] „Technology preview: Sealed sender for Signal“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/blog/sealed-sender/>
- [2] „Meredith Whittaker (@Mer__edith@mastodon.world)“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: https://mastodon.world/@Mer__edith/111563866152334347

- [3] „How to build large-scale end-to-end encrypted group video calls“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/blog/how-to-build-encrypted-group-calls/>
- [4] „Introducing Signal Secure Backups“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/blog/introducing-secure-backups/>
- [5] „Specifications >> The X3DH Key Agreement Protocol“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/docs/specifications/x3dh/>

- [6] „Rome wasn't built in a Signal day - DESOSA 2020“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://desosa.nl/projects/signal-android/2020/03/19/rome-wasnt-built-in-a-signal-day>
- [7] „Specifications >> The X3DH Key Agreement Protocol“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/docs/specifications/x3dh/>
- [8] „Signal Messenger: Speak Freely“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/docs/specifications/doubleratchet/>

- [9] „Mechanisms: Ratchet and Pawl“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://www.notesandsketches.co.uk/Ratchet.html>
- [10] „Technology Preview: Signal Private Group System“. Zugegriffen: 19. Januar 2026. [Online]. Verfügbar unter: <https://signal.org/blog/signal-private-group-system/>
- [11] M. Chase, T. Perrin, und G. Zaverucha, „The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption“. Zugegriffen: 8. Januar 2026. [Online]. Verfügbar unter: <https://eprint.iacr.org/2019/1416>