Zusammenfassung WhatsApp Whitepaper zu Verschlüsselung

# Ziele der Verschlüsselung:

1. End-2-end sodass keiner, auch nicht WhatsApp, Inhalt der Nachrichten lesen kann (WhatsApp hat aber immernoch Meta/Verbindungs-Daten)
2. Auch wenn Private Schlüssel auf Gerät komprimitiert werden sollen vergangene Nachrichten nicht entschlüsselt werden können 🡪 AKA Forward Secrecy
3. Asynchroner Session Aufbau 🡪 Alice will Meldungen an WhatsApp Server schicken obwohl Bob gerade offline ist
4. Nicht garantierte Übertragung der Nachrichten, verspätet, out of order oder gar lost
5. Messages werden mit AES-256 im CBC Modus verschlüsselt und mittels HMAC-SHA256 autentifiziert
6. Auch Anhänge/Medien werden verschlüsselt. Dazu erzeugt der Sender on-the-fly einen symmetrischen Schlüssel, verschlüsselt die Daten mit AES256/HMAC-SHA256 und lädt die verschlüsselten Daten in den WhatsApp Blob-Store. Anschliessend wird eine normale und daher verschlüsselte Nachricht an den Empfänger gesendet mit einem Verweis auf den Blob-Store und dem on-the-fly generierten Schlüssel
7. Gruppennachrichten werden mittels eines allen Gruppentteilnehmers bekannten und bei neuen/weniger Teilnehmners jeweils zu aktualisierendem Sender Key bzw. davon abgeleiteten Messages Keys einmal verschlüsselt. Somit muss nicht der Client allen Gruppenteilnehmner die Nachricht selber zustellen, sondern der WhatsApp Server übernimmt das verteilen 🡪 Perfmance
8. Public-Key können per QR-Code bzw. 60-Ziffern nummer verifiziert werden
9. Die Verschlüsselten Protokoll PDU's werden zusätzlich in einem Verschlüsselten Transportkanal übertragen

Annahme für weitere Beispiele: Alice initiiert Kommunikation mit Bob

# Beschreibung Key Types

## Public-Key Types

|  |  |  |  |
| --- | --- | --- | --- |
| Bezeichnung | Algorithmus | Generierzeitpunkt/Lifetime | Anzahl im System |
| Identity Key Pair | Curve25519 | Installationszeit | 1 |
| Signed Pre Key  wird mit Identity Key signiert | Curve25519 | Installationszeit + Periodische neu-generierung | jeweils 1 gleichzeitig |
| One-Time Pre Keys  Queue mit vielen Schlüsselpaaren. | Curve25519 | Bei Installation wird eine gewisse Menge generiert und je nach Bedarf werden weitere im Betrieb generiert | mehrere gleichzeitig |

## Session Key Types

### Root Key

256-Bit – Generierung von Chain Keys

### Chain Key

256-Bit – Generierung von Message Keys

### Message Key

80-Byte Key,

Aufteilung:

* 32-Byte(256-Bit) für AES-256 symetrische Verschlüsselung 🡪 Wahrung von Confidentiality
* 32-Byte für HMAC-SHA256 für Signatur 🡪 Wahrung von Integrity
* 16-Bytes für IV(Initialisierungsvektor)

# Ablauf Client Registrierung

Client sendet seinen Identity Key, seinen signieren Signed Pre Key sowie mehrere One-Time Pre Keys zum Server. Alle Private Keys bleiben auf Client

# Session Aufbau Alice

1. Alice fragt bei WhatsApp-Server Identity-Key von Bob, den Signed Pre Key und einen One-Time Pre Key an. Server entfernt den One-Time Pre Key, da dieser nur einmal verwendet werden darf. Wenn keine One-Time Pre Keys mehr vorhanden wird auch keine zurückgegeben 🡪 Bob hat sich schon lange nicht mehr angemeldet und konnte die Keys nicht rechtzeitig auffrischen
2. Alice merkt sich die infos, werden in Zukunft mit IBob, SBob und OTBob bezeichnet
3. Alice generiert ein ephemeral Curve 25519 Key-Pair 🡪 EAlice
4. Alice berechnet das master secret wie folgt: master\_secret = ECDH(IAlice, SBob) || ECDH(EAlice, IBob) || ECDH(EAlice, SBob) || ECDH(EEAlice, OBob). Falls bei 1 keine Keys mehr von Bob auf Server, wird letzter Teil weggelassen
5. Alice erstellt Session Keys Root Key und Chain Keys mittels HKDF und master\_secret aus 4
6. Ab jetzt kann Alice WhatsApp Meldungen für Bob schicken, welche dann gepuffert werden. allerdings wird jeweils EAlice und IAlice mitgeschickt.

# Session Aufbau Bob

1. Bob berechnet das master\_secret mittels seinen eigenen Private Keys und der Public-Keys von Alice.
2. Der One-Time Pre Key wird auch auf dem Client gelöscht
3. Bob berechnet ebenfalls den Root Key und die Chain Keys basierend auf dem master\_secret

# Messages verschicken

Jede Message wird mit einem eigenen Key verschlüsselt 🡪 Forward Secrecy

### Berechnung eines Message Keys für die Symmetrische Verschlüsselung

1. Message Key = HMAC-SHA256(Chain Key, 0x01)
2. Chain Key = HMAC-SHA256(Chain Key, 0x02) 🡨 Chain Key wird mit jeder Nachricht eins weiter gezählt. WhatsApp bezeichnet das als "raching forward", im prinzip ist das einfach eine art Counter Mode CTR

### Berechnung eines Chain Key aus einem Root Key

Mit jeder Nachricht wird ein ephemeral Public-Key mitgesendet. Immer wenn eine Antwort ankommt, wird einen neuer Chain Key und Root Key wie folgt berechnet

1. ephemeral\_secret = ECDH(Ephemeralsender, Ephemeralrecipient)
2. Chain Key, Root Key = HKDF(Root Key, ephemeral\_secret)

# Medien/Attachments versenden

1. Generierung eines AES256 und HMAC-SHA256 Schlüssels für symetrische Verschlüsselung sowie einen zufällligen IV.
2. Attachment wird mittels Key AES-256 im CBC Modus verschlüsselt und mit HMAC-SHA256 wird die Integrität sichergestellt
3. Der Sender lädt diese verschlüsselten und signierten Daten in den WhatsApp Blob-Store hoch und erhält einen Handle auf diesen
4. Jetzt versendet der Client eine normale verschlüsselte Nachricht an den Empfänger welcher die in 1 generierten symmetrischen-Schlüssel enthält sowie der Blob-Store-Verweis aus Schritt 3
5. Der Empfänger kann mittels des Schlüssels/Hashes die Daten aus dem Blob-Store herunterladen, die MAC-Prüfen und entschlüsseln

# Gruppennachrichten

Beim Hinzufügen/Verlassen eines Gruppenteilnehmers wird jeweils ein Gruppen-Key ermittelt, wobei alle Beteiligten einmalig über neue Teilnehmer informiert werden(Neuling stellt sich vor) und ihre Gruppenschlüssel aktualisieren. An

# WhatsApp Calls

1. Session wird analog zu normaler Session aufgebaut
2. der Anrufer generiert ein 32-Byte master-secret für die SRTP(verschlüsselte Sprachübertragung)
3. Der Anrufer sendet dem Angerufenen eine verschlüsselte Nachricht mit Inhalt "Hallo, Telefon bitte abnehmen" und dem SRTP master-secret