CRYPTO 101:
REAL-WORLD DEPLOYMENTS

6. THE SIGNAL PROTOCOL

Alfred Menezes
cryptography101.ca

Outline

1. Signal and WhatsApp
2. Forward secrecy
3. Post-compromise security
4. The Signal protocol

Introduction

The Signal protocol was designed by Moxie Marlinspike and Trevor Perrin.

It is free, open source, and is used in:

- Signal (free messaging app).
- Skype ("private conversations" optional feature).
- WhatsApp.

- Facebook Messenger uses an end-to-end encryption protocol that is similar to Signal.

WhatsApp

- WhatsApp is owned by Facebook.
- Has over 3 billion users (India, Brazil, Mexico, France, UK, …), and handles 10’s of billions of messages everyday.
- Is banned, or has been temporarily blocked, in several countries.
- Has very low revenues — it’s free, works over WiFi, no advertisements, and no user data to mine (except metadata).

Signal objectives (1)

Participants: Alice, Bob, WhatsApp, ThirdParty (E).

1. Long-lived sessions: Alice and Bob establish a long-lived secure communications session. The session lasts until events such as app reinstall or device change.

2. Fresh session keys:
   Each message is encrypted/authenticated with a fresh session key.
   Encrypt-then-MAC is used: $c = \text{AES-CBC}_{k_1}(m)$, $t = \text{HMAC}_{k_2}(c)$, where k_1, k_2 are each 256 bits in length.

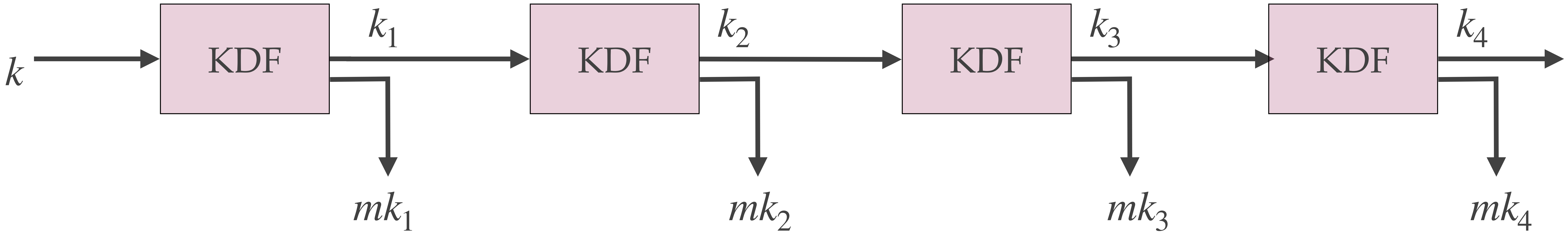
3. Asynchronous setting:
   Alice can send Bob a secure message even if Bob is offline. Messages can be delayed, delivered out of order, or can be lost entirely without problem.

Signal objectives (2)

4. Immediate decryption: Bob can decrypt a ciphertext as soon as he receives it.
5. End-to-end encryption: WhatsApp and E do not possess any of Alice's or Bob's secret keys, nor do they get access to any plaintext.
   - However, WhatsApp (but not E) does get all the metadata, e.g., who sent a message to whom and when, your contacts, your profile name, etc.
6. Forward secrecy:
   If a party's state is leaked, then none of the previous messages should be compromised (assuming they have been deleted from the state).
7. Post-compromise security:
   Parties recover from a state compromise (if the attacker remains passive).

Forward secrecy

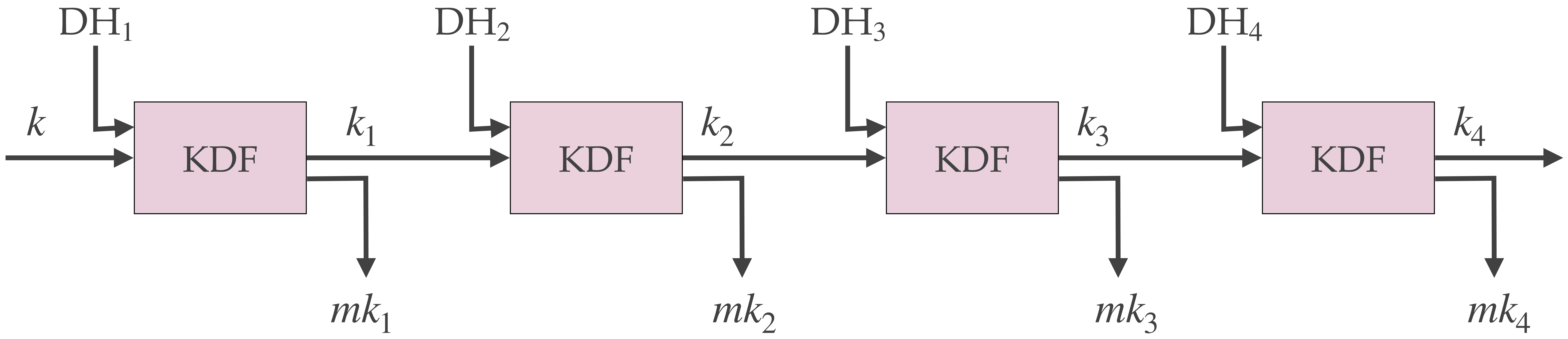
Suppose that Alice and Bob share a secret key k. They can ratchet k and derive message encryption keys $mk_1, mk_2, mk_3, \ldots$ as follows:



- Keys are deleted as soon as they are no longer needed.
- For example, k is deleted as soon as k_1 and mk_1 are computed. Also, mk_1 is deleted as soon as it is used to encrypt (or decrypt) a message.
- Suppose that E learns k_2 and mk_2 (e.g., by gaining access to Alice's device). Then E can compute $k_3, mk_3, k_4, mk_4, \ldots$. However, E cannot compute mk_1. Thus, any ciphertext that was generated using mk_1 cannot be decrypted by E.

Post-compromise security

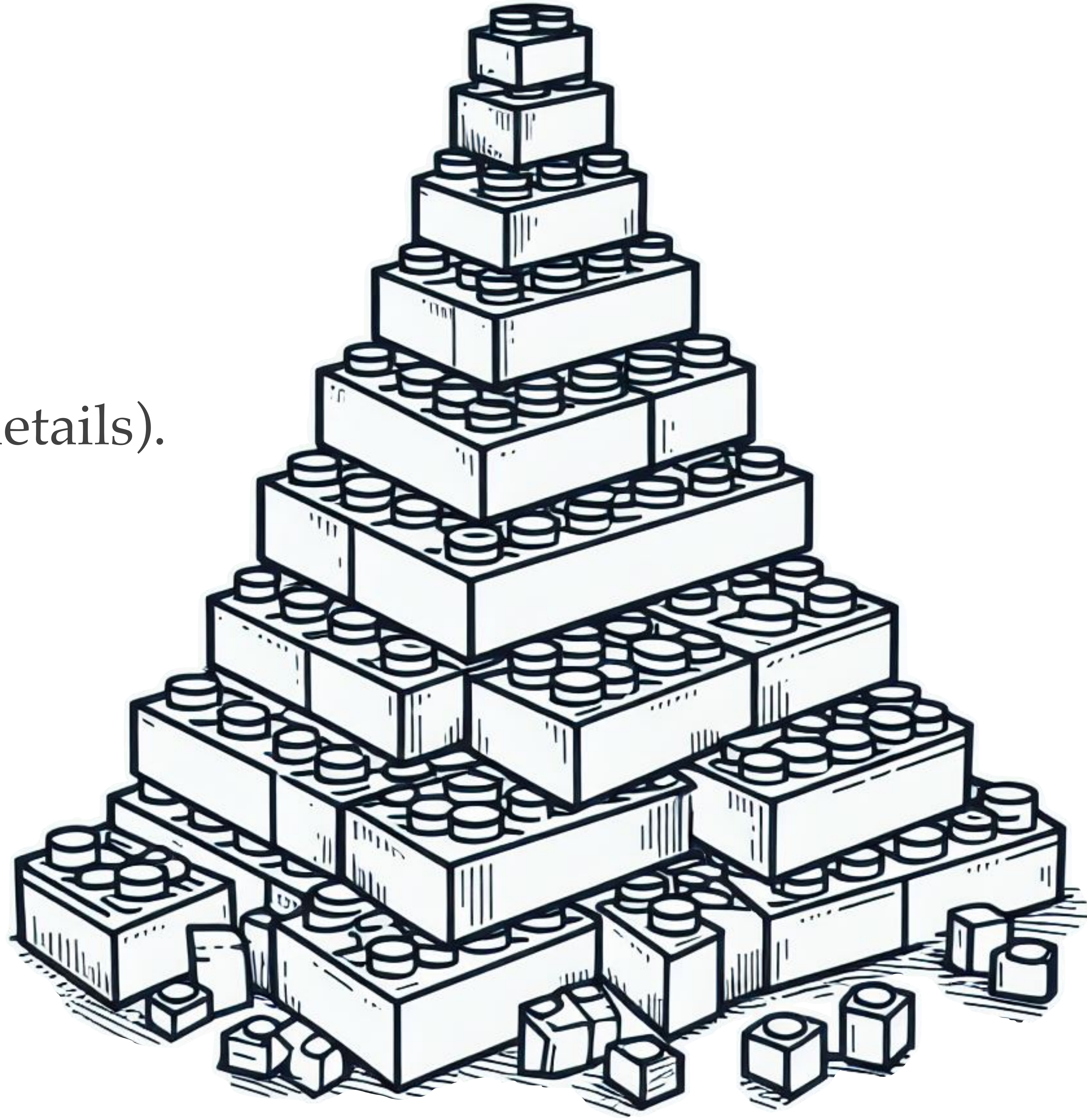
In order to achieve post-compromise security, a fresh ECDH shared secret established by Alice and Bob is used each time the KDF is applied.



- Here, $DH_i = ECDH(X_i, Y_i)$, where X_i is contributed by Alice and Y_i is contributed by Bob.
- Suppose that E learns k_2 and mk_2.
  Then, E cannot compute k_3 or mk_3 unless she also learns x_3 or y_3.

Cryptographic ingredients

1. AES-CBC: 128-bit IV, 256-bit key.
2. HMAC: with SHA-256 and a 256-bit key.
3. KDF: A key derivation function (HMAC or HKDF, but we will not get into the details).
4. Curve25519.
5. Elliptic curve key pairs: (X, x) where $x \in_R [1, n-1]$ is a secret key and $X = xP$ is the corresponding public key.
6. ECDH.
7. EdDSA: an ECDSA-like signature scheme.



Signal protocol

Three stages:

1. Registration
2. Root key establishment
3. Message transmission

Note 1: All of Alice's and Bob's messages are sent via WhatsApp's servers.

Note 2: All communication between Alice/Bob and WhatsApp is encrypted/authenticated using a TLS-like protocol.

Note 3: Alice (and Bob) always deletes a secret key as soon as she no longer needs it.

Registration

1. After Alice has downloaded the WhatsApp app, she sends WhatsApp:
   - ID_A: her identifier (cell phone number)
   - A: her long-term public key
   - U: her medium-term public key
   - $\text{Sign}_A(U)$: her signature on U
   - $S_1, S_2, \ldots, S_\ell$: one-time public keys

   (Alice securely stores her secret keys $a, u, s_1, s_2, \ldots, s_\ell$.)
2. Similarly, Bob sends WhatsApp ID_B, B, V, $\text{Sign}_B(V)$, $T_1, T_2, \ldots, T_\ell$.

Root key establishment

Alice (the initiator) wishes to connect with Bob (the responder).

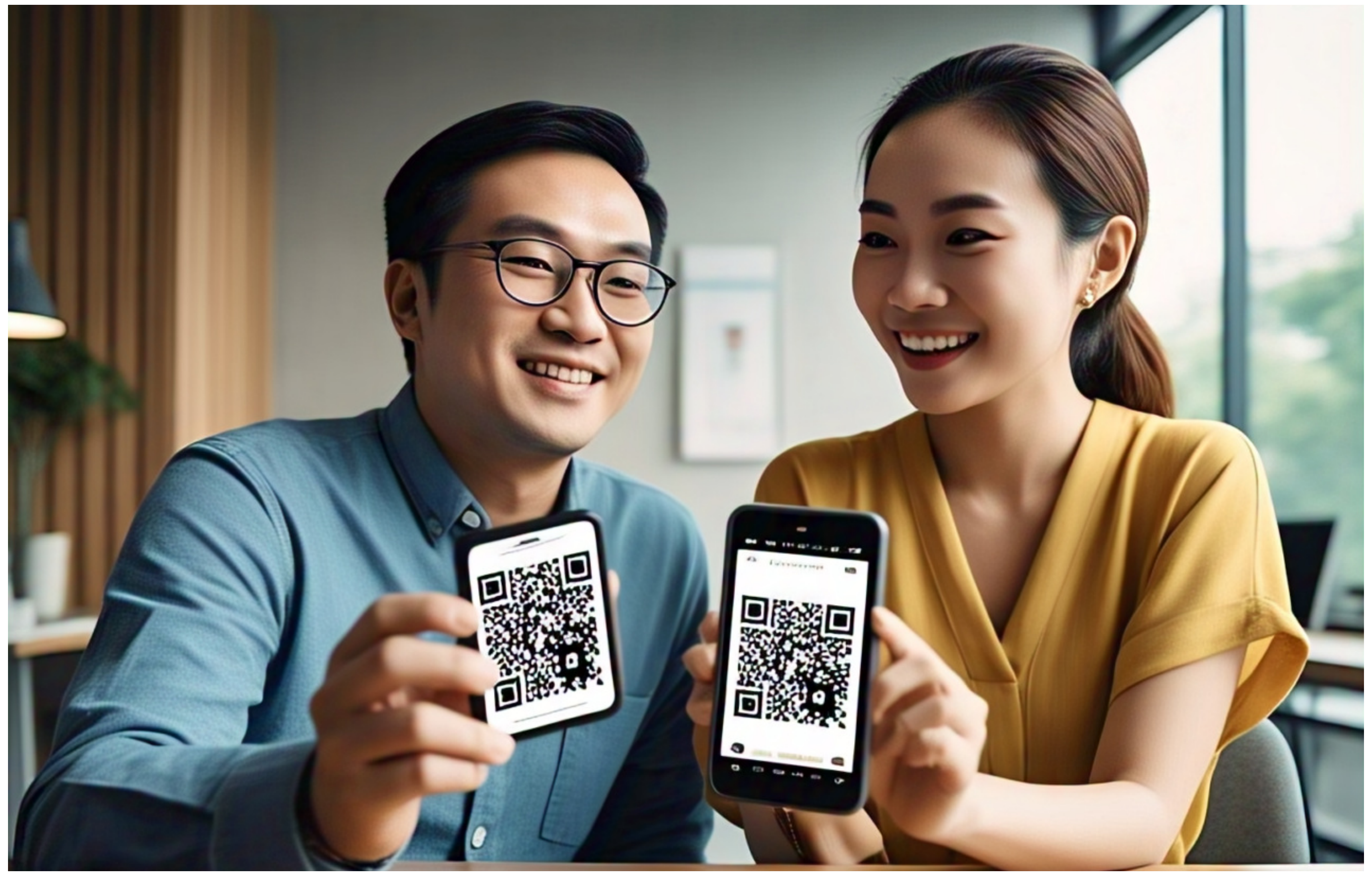
1. Alice → WhatsApp: request to create a session with Bob.
2. WhatsApp → Alice: B, V, $\text{Sign}_B(V)$, T_1 (and deletes T_1).
3. Alice does the following:
   (a) Verify $(V, \text{Sign}_B(V))$ using B.
   (b) Select an ephemeral key pair (Z, z).
   (c) Compute the 256-bit root key $\text{root}_0 = \text{KDF}(aV, zB, zV, zT_1)$.

Note: Given A and Z, Bob can compute $\text{root}_0 = \text{KDF}(vA, bZ, vZ, t_1Z)$.

Verifying long-term public keys

Purpose:
Provide some protection against active MITM attacks.

- QR codes and 60-digit numbers encode identifiers and long-term public keys: (ID_A,A) and (ID_B,B).
- Alice and Bob *should* verify these prior to sending each other messages.



Message transmission

Alice maintains three key chains:

1. A root key chain — to seed the other two chains.
2. A sending key chain — to generate message sending keys.
3. A receiving key chain — to generate message receiving keys.

Bob also maintains three key chains:

1. A root key chain — the same one as Alice's.
2. A receiving key chain — the same as Alice's sending chain.
3. A sending key chain — the same as Alice's receiving chain.

Example of message transmission (1)

Consider the following example:

1. Alice → Bob: $M_{11}^1, M_{11}^2, M_{11}^3, M_{11}^4$.
   (Alice's first sending chain of 4 messages)

2. Alice ← Bob: M_{12}^1, M_{12}^2.
   (Alice's first receiving chain of 2 messages)

3. Alice → Bob: M_{22}^1.
   (Alice's second sending chain of 1 message)

4. Alice ← Bob: $M_{23}^1, M_{23}^2, M_{23}^3$.
   (Alice's second receiving chain of 3 messages)


Alice's root key chain
DH11
DH12
DH22
DH23
root0
KDF
root11
KDF
root12
KDF
root22
KDF
root23
sk011
rk012
sk022
rk023
Alice's first sending key chain
sk011
KDF
sk111
KDF
sk211
KDF
sk311
KDF
sk411
msk111
msk211
msk311
msk411
Alice's first receiving key chain
rk012
KDF
rk112
KDF
rk212
mrk112
mrk212
Alice's second sending key chain
sk022
KDF
sk122
msk122
Alice's second receiving key chain
rk023
KDF
rk123
KDF
rk223
KDF
rk323
mrk123
mrk223
mrk323
1. Alice → Bob: M111, M211, M311, M411 .
2. Alice ← Bob: M112, M212.
3. Alice → Bob: M122.
4. Alice ← Bob: M123, M223, M323.

Example of message transmission (2)

Alice's root key chain	Alice's sending key chains	Alice's receiving key chains
1. Select X_1		
2. $\mathrm{DH}_{11} = \mathrm{ECDH}(X_1, Y_1)$, $Y_1 = V$		
3. $\mathrm{KDF}(\mathrm{root}_0, \mathrm{DH}_{11}) \to \mathrm{root}_{11}, sk_{11}^0$.	4. $\mathrm{KDF}(sk_{11}^0) \to sk_{11}^1, msk_{11}^1$	
	5. $\mathrm{KDF}(sk_{11}^1) \to sk_{11}^2, msk_{11}^2$	
	6. $\mathrm{KDF}(sk_{11}^2) \to sk_{11}^3, msk_{11}^3$	
	7. $\mathrm{KDF}(sk_{11}^3) \to sk_{11}^4, msk_{11}^4$	
8. Receive Y_2		
9. Compute $\mathrm{DH}_{12} = \mathrm{ECDH}(X_1, Y_2)$		
10. $\mathrm{KDF}(\mathrm{root}_{11}, \mathrm{DH}_{12}) \to \mathrm{root}_{12}, rk_{12}^0$		11. $\mathrm{KDF}(rk_{12}^0) \to rk_{12}^1, mrk_{12}^1$
		12. $\mathrm{KDF}(rk_{12}^1) \to rk_{12}^2, mrk_{12}^2$
13. Select X_2		
14. Compute $\mathrm{DH}_{22} = \mathrm{ECDH}(X_2, Y_2)$		
15. $\mathrm{KDF}(\mathrm{root}_{12}, \mathrm{DH}_{22}) \to \mathrm{root}_{22}, sk_{22}^0$.	16. $\mathrm{KDF}(sk_{22}^0) \to sk_{22}^1, msk_{22}^1$	
17. Receive Y_3		
18. Compute $\mathrm{DH}_{23} = \mathrm{ECDH}(X_2, Y_3)$		
19. $\mathrm{KDF}(\mathrm{root}_{22}, \mathrm{DH}_{23}) \to \mathrm{root}_{23}, rk_{23}^0$		20. $\mathrm{KDF}(rk_{23}^0) \to rk_{23}^1, mrk_{23}^1$
		21. $\mathrm{KDF}(rk_{23}^1) \to rk_{23}^2, mrk_{23}^2$
		22. $\mathrm{KDF}(rk_{23}^2) \to rk_{23}^3, mrk_{23}^3$

Message transmission (2)

Notation:

- sk = chaining key for sending key chain.
- rk = chaining key for receiving key chain.
- msk = message sending key.
- mrk = message receiving key

Alice sending M_{ii}^{j} :

- $C_{ii}^{j} = \mathrm{AE}_{msk_{ii}^{j}}((A, B, X_i, j, L_{i-1}), M_{ii}^{j})$,
  where L_{i-1} is the length of Alice's $(i-1)$th sending chain.
- Here, $\mathrm{AE}_k(T, M) = \big(T,\ \ \text{AES-CBC}_{IV,k_1}(M),\ \ \mathrm{HMAC}_{k_2}(T, \text{AES-CBC}_{IV,k_1}(M))\big)$,
  where $k = (IV, k_1, k_2)$ with $IV \in \{0,1\}^{128}$, $k_1, k_2 \in \{0,1\}^{256}$.

Notes: The initiator (Alice) always sends the first message M_{11}^{1} to the responder (Bob).
Also, Alice's ephemeral public key Z is included in all her messages in her first sending chain.

References

1. The double ratchet algorithm
   tinyurl.com/DoubleRatchet
2. Signal source code
   tinyurl.com/SignalProtocol
3. WhatsApp encryption overview
   tinyurl.com/WhatsAppEnc
4. A formal security analysis of the Signal messaging protocol
   eprint.iacr.org/2016/1013