CS 5435

Signal Protocol

Vitaly Shmatikov

Signal Protocol

Developed by Whisper Systems

- Open-source implementation

Used by Signal, Whatsapp, Facebook Messenger "Secret Conversations"

End-to-end encryption: third parties <u>and providers</u> do not have access to messages and calls

Forward secrecy: encryption keys cannot be used to go back in time to decrypt previously transmitted messages… also future secrecy

Signal Protocol

1. Client registers with messaging server
2. Two clients set up a session
3. Exchange messages

Client Keys

Long-term identity key pair IK

- Generated when client program is installed

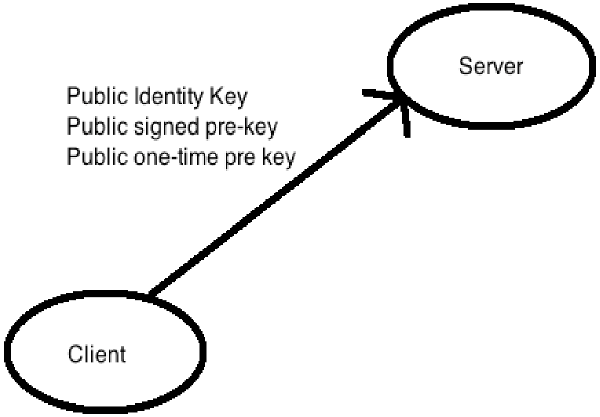
Medium-term signed pre-key pair SPK

- Generated when client program is installed
- Changes periodically

Ephemeral one-time pre-key pair OPK

- Selected from a list generated when client program is installed; when the list is used up, another list is generated

Client Registration



Client signs her public pre-key:

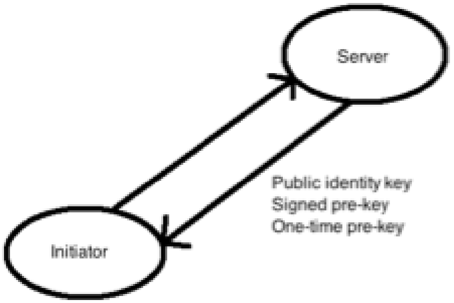
SSPK = {SPK} signed with IK_{priv}

Sends to server her pre-key bundle:

{ IK_{pub} || SPK_{pub} || SSPK || OPK_1 || OPK_2 || . . . }

where OPK_1, OPK_2, … are ephemeral one-time pre-key public keys

Session Setup



Sender obtains recipient’s pre-key bundle

$\{ \mathrm{IK}_{pub,recip} \;||\; \mathrm{SPK}_{pub,recip} \;||\; \mathrm{SSPK} \;||\; \mathrm{OPK}_{recip,i} \}$

only one of the recipient’s ephemeral pre-keys included

Verifies recipient’s signature on SSPK

Generates new ephemeral key pair $\mathrm{EK}_{\{pub,priv\},send}$

Computes master secret

Diffie-Hellman

$$
\begin{aligned}
ms = {} & \mathrm{ECDH}(\mathrm{IK}_{priv,send}, \mathrm{SPK}_{pub,recip}) \;||\\
& \mathrm{ECDH}(\mathrm{EK}_{priv,send}, \mathrm{IK}_{pub,recip}) \;||\; \mathrm{ECDH}(\mathrm{EK}_{priv,init}, \mathrm{SPK}_{pub,recip}) \;||\\
& \mathrm{ECDH}(\mathrm{EK}_{priv,send}, \mathrm{OPK}_{recip,i})
\end{aligned}
$$

Delete $\mathrm{EK}_{priv,send}$ and all intermediate values

Session Keys

Root key: 32-byte value used to generate chain keys

Chain key: 32-byte value used to generate message keys

Message key: 80-byte key used to encrypt messages

- 32-byte key for AES-256 encryption
- 32-byte key for HMAC-SHA256 cryptographic checksum
- 16-byte initialization vector

Key Generation

1. s salt (0 if omitted); x is key material

   key k = HMAC_SHA256(s, x)

2. info is string of characters like “WhisperGroup”

   T(0) = “”

   …

   T(i) = HMAC_SHA256(k, T(i–1) || info || i)

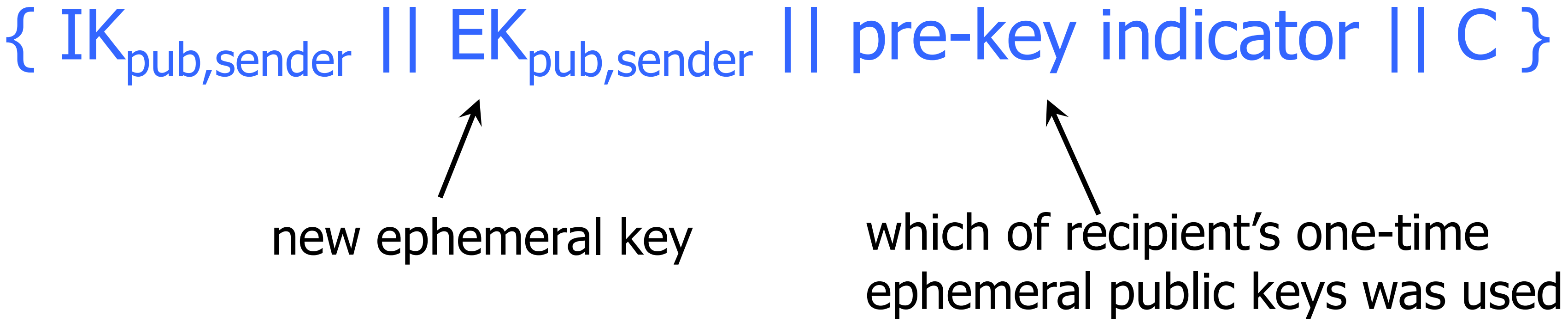
   HDKF(s,x) = T(1) || T(2) || … L octets

Sending Messages

Sender creates message key

k_m = HMAC_SHA256($k_{c,1}$, 1)

Encrypts message using AEAD (authenticated encryption) scheme with AES-256 in CBC mode for encryption and HMAC_SHA256 for authentication, producing ciphertext C



{ $IK_{pub,sender}$ || $EK_{pub,sender}$ || pre-key indicator || C }

new ephemeral key

which of recipient's one-time ephemeral public keys was used

Receiving Messages

Recipient computes master secret analogously to sender but using his private keys and sender's public key

Computes the root and chain keys

Deletes ephemeral key pair OPK_i used for this exchange

Can now exchange messages…

Changing Message Keys

For each message sent before receiving a reply, use a hash ratchet to change message key:

$$k_{m,i+1} = \text{HMAC\_SHA256}(k_{c,i}, 1)$$
$$k_{c,i+1} = \text{HMAC\_SHA256}(k_{c,i}, 2)$$

After receiving a reply, compute new chain and root keys:

$$x = \text{HKDF}(k_r, \text{ECDH}(EK_{pub,recip}, EK_{priv,send}))$$

where $EK_{pub,recip}$ is from the received message, $EK_{priv,send}$ is the private key associated with $EK_{pub,send}$ appearing in the message to which this was a reply

- First 32 octets are the new chain key, next 32 octets new root key

Basically, fresh Diffie-Hellman for every message-reply pair

Properties

"The protocol provides confidentiality, integrity, authentication, participant consistency, destination validation, forward secrecy, post-compromise security (aka future secrecy), causality preservation, message unlinkability, message repudiation, participation repudiation, and asynchronicity"

Service knows who is talking to whom

- Signal's privacy policy says that they do not keep identities longer than needed to transmit each message
- Only keeps the last time user connected to server

What else?