Secure Messaging Authentication against Active Man-in-the-Middle Attacks

Benjamin Dowling  b.dowling@sheffield.ac.uk
ETH Zurich, University of Sheffield
Britta Hale  britta.hale@nps.edu
Naval Postgraduate School
2021-09-08
Signal as Asynchronous Protocol

Asynchronous messaging protocol:

<table>
<thead>
<tr>
<th>Alice</th>
<th>Server</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reg Phase</strong></td>
<td>$PKB_A$</td>
<td>$PKB_B$</td>
</tr>
<tr>
<td>$id_B$</td>
<td>$PKB_B$</td>
<td>$ctxt$</td>
</tr>
<tr>
<td><strong>Init Phase (X3DH)</strong></td>
<td>$ctxt$</td>
<td>$ctxt$</td>
</tr>
<tr>
<td><strong>Sym Phase</strong></td>
<td>$ctxt$</td>
<td>$ctxt$</td>
</tr>
<tr>
<td><strong>Asym Phase</strong></td>
<td>$ctxt$</td>
<td>$ctxt$</td>
</tr>
</tbody>
</table>
Safety numbers!

Codes (QR & Numeric) contain long-term public keys and party identifiers:

\[
\text{local_fprint} = H_i(0 \parallel \text{fvers} \parallel \text{idpk}_A \parallel \text{id}_A, \text{idpk}_A)
\]

\[
\text{remote_fprint} = H_i(0 \parallel \text{fvers} \parallel \text{idpk}_B \parallel \text{id}_B, \text{idpk}_B)
\]

\[
\text{safety number} = \text{local_fprint} \parallel \text{remote_fprint}
\]

Claim: this will “verify the security of [the users] encryption.” (Signal App)

We identify weaknesses within this safety number authentication construct.
**Issue 1: No Session Authentication**

- Safety number is computed from purely public (and static) information.
  
  \[
  \text{local\_fprint} = H_i(0||fvers||idpk_A||id_A, idpk_A) \\
  \text{remote\_fprint} = H_i(0||fvers||idpk_B||id_B, idpk_B)
  \]

- If an attacker has learned the long-term secret key of the communicating partner, then creating a PreKeyBundle is trivial.

- Impersonating attacks possible, verifying the Safety Number does not detect this attack.

**User Authentication Phase**

If \(A\) display \(\neq\) Bob display, authentication successful.
Issue 2: Attacks possible with Display control

- Safety number is computed from purely public (and static) information.
  \[
  \text{local\_fprint} = H_i(0||fvers||idpk_A||id_A, idpk_A) \\
  \text{remote\_fprint} = H_i(0||fvers||idpk_B||id_B, idpk_B)
  \]
- If an attacker can control user display via an overlay (access to secret state not necessary), then a forged safety number is displayed to the verifying party.

\[
\begin{align*}
\text{Initiator} & \quad \text{Attacker} \quad \text{Responder} \\
lf = H_i(0||fvers||idpk_A||id_A, idpk_A) & \quad \text{RevealUser} & \quad rf = H_i(0||fvers||idpk_B||id_B, idpk_B) \\
lf = H_i(0||fvers||idpk_A||id_A, idpk_A) & \quad Trunc(lf)||Trunc(rf) & \quad rf = H_i(0||fvers||idpk_B||id_B, idpk_B) \\
\text{Alice display} = \text{Trunc}(lf)||\text{Trunc}(rf) & \quad \quad \text{Bob display} = \text{Trunc}(lf)||\text{Trunc}(rf) \\
\end{align*}
\]
If Alice display ≠ Bob display, authentication successful.
Contributions

1. Security model to capture user-mediated authentication protocols (META)

2. Efficient and clean adaptation of Signal to achieve session authentication and per-epoch authentication: Modified Device-to-User Signal Authentication (MoDUSA)
**High-level security goal:** When a session at a Device “accepts”, then there exists another honest and matching session (subset transcription matching).
META: Mediated Epoch Three-party Authentication Threat Model

Attacker is capable of leaking long-term and device state.
Attacker is able to compromise *directions* on the User-to-Device channel independently, sending messages between the User and the Devices.
Compromise Settings

**Compromised User:**

- The attacker **cannot** leak current epoch secrets.
- The attacker **cannot** RevealUser on both devices, allowing the attacker to inject messages from the Devices to the User.
- The attacker **cannot** CorruptUser, allowing the attacker to inject messages from the User to the Devices.

**Compromised Device:**

- The attacker **can** leak any secrets from the devices; and
- The attacker **cannot** RevealUser on either device, preventing the attacker from injecting any messages from the Devices to the User.
- The attacker **cannot** CorruptUser, allowing the attacker to inject messages from the User to the Devices.
Active Post-Compromise Security

Adversary behavior after compromise

Passive

CompUser:
Security under traditional PCS and unrestricted partner user interface adversarial abilities

Active

CompDev:
Security under restricted user interface adversarial abilities
Signal Authentication Insecure in Both Settings

Insecure Under Compromised User:

Tactic: Use RevealUser

Attack succeeds since Signal Safety Numbers are over purely public information.

Insecure under Compromised Device:

Tactic: Leaking session state, inject messages

Since Signal Safety Numbers doesn’t authenticate per-session information, this attack is successful.
Modify the Signal Key Schedule to add an additional authentication key.
Hashed Transcripts

Pair of hashed transcripts of all public-key values sent between the two parties. Pair maintained in case (due to asynchronicity) one party has not received the most recent ratchet public key.

\[ H_{i-1} = \text{Hash} \left( PKB \parallel g^{A_0} \parallel g^{B_0} \parallel \ldots \parallel g^{A_i} \right) \]

\[ H_i = \text{Hash} \left( PKB \parallel g^{A_0} \parallel g^{B_0} \parallel \ldots \parallel g^{A_i} \parallel g^{B_i} \right) \]
Safety numbers now update per epoch. Verification of safety number implies agreement on transcript of all cryptographic information.

\[ FP_{i-1} = MAC \left( ak_{i-1}, H_{i-1}^{i-1} \parallel \text{role} \right) \]

\[ FP_{i} = MAC \left( ak_{i}, H_{i} \parallel \text{role} \right) \]

Role separation prevents reflection attacks.

Maintain pair of safety numbers to account for potential asynchronicity.
Results of Analysing MoDUSA in META

MoDUSA:

<table>
<thead>
<tr>
<th>Auth. Initiator</th>
<th>Auth. Responder</th>
<th>CD Without E.</th>
<th>CD with E.</th>
<th>CU Without E.</th>
<th>CU With E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display match</td>
<td>Display match</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Display match</td>
<td>Scan match</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Scan match</td>
<td>Display match</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Scan match</td>
<td>Scan match</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Display non-match</td>
<td>Scan non-match</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Scan non-match</td>
<td>Display non-match</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scan non-match</td>
<td>Scan non-match</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **CD**: Compromised Device.
- **CU**: Compromised User.
- **E**: Eavesdropper.
Summary

Device-to-Device Channel

User-to-Device Channel

Display, User Interaction, etc...

Alice

Root Keys

Bob

Chain Keys

\[ g_A^{i-2} \]

\[ g_A^{i-1} \]

\[ g_B^{i+1} \]

KDF

KDF

KDF

SendDevice\((m)\)

CorruptUser

RevealUser

Corrupt(B)

SendDevice\((m)\)

(m')

\((pk_A, sk_A)\)

\((esk[T], st[T])\)

\((pk_B, sk_B)\)

\((esk[T], st[T])\)