User Authentication within Groups

Britta Hale

Based on joint work with Benjamin Dowling

*The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.
Note that the long-term identity keys used by the protocol MUST be distributed by an "honest" authentication service for clients to authenticate their legitimate peers.

[[ OPEN ISSUE: Signatures under the identity keys, while simple, have the side-effect of preclude deniability. We may wish to allow other options, such as (ii) a key chained off of the identity key, or (iii) some other key obtained through a different manner, such as a pairwise channel that provides deniability for the message contents.]]
init_secret_[n-1] (or Θ)

update_secret -> HKDF-Extract = epoch_secret

  ---> Derive-Secret(., "sender data", GroupContext_[n])
      = sender_data_secret

  ---> Derive-Secret(., "handshake", GroupContext_[n])
      = handshake_secret

  ---> Derive-Secret(., "app", GroupContext_[n])
      = application_secret

  ---> Derive-Secret(., "confirm", GroupContext_[n])
      = confirmation_key

  Derive-Secret(., "init", GroupContext_[n])

  init_secret_[n]
Lessons from Signal Authentication

If you wish to verify the security of your encryption with [signal], compare the number above with the number on their device. Alternatively, you can scan the code on their phone, or ask them to scan your code. Learn more.
Epoch-Level Authentication
Modified Device-to-User Signal Authentication
New QR-code computation:

\[
\begin{align*}
\text{fprint}^{i-1} &= \text{HMAC}(ak^{i-1}, H^{i-1} \| \text{fvers} \| \text{role}) \\
\text{fprint}^i &= \text{HMAC}(ak^i, H^i \| \text{fvers} \| \text{role})
\end{align*}
\]

*Session specific

*Asynchronicity in computation
# Achievable Guarantees

<table>
<thead>
<tr>
<th>Auth. Initiator $I$</th>
<th>Auth. Responder $I'$</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
init_secret_[n-1] (or 0)

update_secret -> HKDF-Extract = epoch_secret

   --> Derive-Secret(. , "sender data", GroupContext_[n])
         = sender_data_secret

   --> Derive-Secret(. , "handshake", GroupContext_[n])
         = handshake_secret

   --> Derive-Secret(. , "app", GroupContext_[n])
         = application_secret

   --> Derive-Secret(. , "confirm", GroupContext_[n])
         = confirmation_key

  Derive-Secret(. , "init", GroupContext_[n])

  init_secret_[n]
Epoch-Level Authentication
Group-Level Authentication
For Group Membership Auth

Group Auth Key

ABCDEF

ABCD

AB

A B

EFG

EF

E F

For Group Membership Auth
For Group Membership Auth

Group Auth Key

ABCDEF

ABCD

EFG

AB

CD

EF

A

B

C

D

E

F

G

For Entity Auth?

SigKeyA

Entity Auth breaks under static long-term SigKey