Moving forward: Forward Secrecy in OpenPGP

Justus Winter <justus@sequoia-pgp.org>

DeltaX Freiburg, 2018-07-21

https://sequoia-pgp.org/talks/2018-08-moving-forward
Forward Secrecy

- compromise of long-term keys does not compromise session keys
  - not: Backward Secrecy aka Future Secrecy aka Post Compromise Security
- TLS (DHE-*), OTR, Signal, OMEMO
- in short:
  - use Diffie-Hellman key exchange to derive session keys
  - use long-term keys to authenticate the exchange

Forward Secrecy is a property of transport security
Data at rest vs. data in motion

- OpenPGP may also be used for backups, archives, etc.
- OpenPGP already supports this distinction!

Key Flags\(^1\) to the rescue:

0x04 - This key may be used to encrypt communications.

0x08 - This key may be used to encrypt storage.

<table>
<thead>
<tr>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequelize</td>
</tr>
<tr>
<td>✔</td>
</tr>
</tbody>
</table>

\(^1\)Section 5.2.3.21 of RFC4880
Approximating Forward Secrecy I

- suggested by Brown et al in 2001\(^2\)
- use short-lived encryption subkeys
- generate and publish in advance
- trivial to implement, requires no changes to peers

% gpg -k futura
pub ed25519 2018-06-11 [SC] [expires: 2019-06-10]
  D2784F6DDEB59AB4162CCD3E0F08F2796B0B71E2
uid [ unknown] Futura Proofoa <futura@example.org>
[...]
sub cv25519 2018-07-16 [E] [expires: 2018-07-23]
sub cv25519 2018-07-09 [E] [expires: 2018-07-16]

\(^2\)https://tools.ietf.org/html/draft-brown-pgp-pfs-03
example:
- encryption keys valid for a week
- publish half a year worth of keys

cons:
- all messages sent in a week are encrypted using the same key
- generating keys in advance is a window for compromise

pros:
- good backwards compatibility
- way better than the status quo

Compatibility

<table>
<thead>
<tr>
<th></th>
<th>Sequoia</th>
<th>GnuPG</th>
<th>OpenKeychain</th>
<th>openpgp.js</th>
<th>rnp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>
Interludum: Multi-device support

OpenPGP lacks a convincing story for multi-device support. Two options:

1. sharing decryption-capable keys across devices
   - + hides number of devices
   - - requires synchronization between co-agents

2. distinct per-device decryption-capable keys
   - + requires synchronization only at setup
   - ± requires synchronization with remote peers
   - + still possible to hide number of devices by sharing
   - - requires minor modifications
   - - size of the certificate
   - - complexity

\[^3\]p≡p and Autocrypt synchronize using hidden mails:
- OpenPGP is transport protocol independent
- how does that work in practice?

\[^3\]dkg’s post to the MLS list

Justus Winter <justus@sequoia-pgp.org> Moving forward: Forward Secrecy in OpeDeltaX Freiburg, 2018-07-21
Simple per-device encryption keys

% gpg -k two
pub  ed25519  2018-06-08 [SC] [expires: 2019-06-07]
     2B7757D8AF283468A0574699910E554478CCDE00
uid  [ unknown] Two Fish <two@example.org>
sub  cv25519  2018-06-08 [E] [expires: 2019-06-07]
sub  cv25519  2018-06-08 [E] [expires: 2019-06-07]

Compatibility

<table>
<thead>
<tr>
<th></th>
<th>Sequoia</th>
<th>GnuPG</th>
<th>OpenKeychain</th>
<th>openpgp.js</th>
<th>rnp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Certification-capable subkeys

- short-lived encryption subkeys require recurrent synchronization
- our proposal⁴:

- use a set of keys per device:
  - a certification subkey: to issue their own subkeys
  - a signing subkey
  - n encryption subkeys
- use e.g. a QR-code containing an encrypted key and a binding signature to provision a new device
- requires clarification in the RFC, minor changes in implementations

Compatibility

<table>
<thead>
<tr>
<th></th>
<th>Sequoia</th>
<th>GnuPG</th>
<th>OpenKeychain</th>
<th>openpgp.js</th>
<th>rnp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>✓/?!</td>
</tr>
</tbody>
</table>

⁴Post to openpgp@ietf.org
Example key

- primary key [C]
  - subkey [E_r]
  - subkey [A]

- new key, maybe on a Gnuk

Justus Winter <justus@sequoia-pgp.org> Moving forward: Forward Secrecy in OpeDeltaX Freiburg, 2018-07-21
Per-device keys: Example

Example key

primary key [C]
- subkey [E_r]
- subkey [A]
- subkey [C]: "desktop"
  - subkey [S]
  - n subkeys [E_t]

- new key, maybe on a Gnuk
- commission desktop
Per-device keys: Example

Example key

- primary key [C]
  - subkey [E_r]
  - subkey [A]
  - subkey [C]: "desktop"
    - subkey [S]
    - n subkeys [E_t]
  - subkey [C]: "laptop"
    - subkey [S]
    - n subkeys [E_t]

- new key, maybe on a GnuK
- commission desktop
- commission laptop from desktop
Per-device keys: Example

Example key

primary key [C]
- subkey [E_r]
- subkey [A]
- subkey [C]: "desktop"
  - subkey [S]
  - n subkeys [E_t]
- subkey [C]: "laptop"
  - subkey [S]
  - n subkeys [E_t]
- subkey [C]: "mobile phone"
  - subkey [S]
  - n subkeys [E_t]

- new key, maybe on a GnuK
- commission desktop
- commission laptop from desktop
- commission phone from desktop
Per-device keys: Example

Example key

primary key [C]

• subkey [E_r]
• subkey [A]
• subkey [C]: "desktop"
  • subkey [S]
  • n subkeys [E_t]
• subkey [C]: "laptop"
  • subkey [S]
  • n subkeys [E_t]
• subkey [C]: "mobile phone"
  • subkey [S]
  • n subkeys [E_t]

• new key, maybe on a GnuK
• commission desktop
• commission laptop from desktop
• commission phone from desktop
• decommissioning desktop recursively decommissions all devices
Per-device keys: Example

Example key

- primary key [C]
  - subkey [E_r]
  - subkey [A]
  - subkey [C]: "desktop"
    - subkey [S]
    - n-subkeys [E_t]

- desktop is decommissioned
Example key

- primary key [C]
  - subkey [E_r]
  - subkey [A]
  - subkey [C]: "desktop"
    - subkey [S]
    - n subkeys [E_t]
  - subkey [C]: "laptop"
    - subkey [S]
    - n subkeys [E_t]

- desktop is decommissioned
- commission laptop again from GnuK
### Example key

**primary key [C]**
- subkey \[E_r\]
- subkey \[A\]
- subkey \[C\]: "desktop"
  - subkey \[S\]
  - n subkeys \[E_t\]
- subkey \[C\]: "laptop"
  - subkey \[S\]
  - n subkeys \[E_t\]
- subkey \[C\]: "mobile phone"
  - subkey \[S\]
  - n subkeys \[E_t\]

- desktop is decommissioned
- commission laptop again from GnuK
- commission phone from laptop
Signal’s Double Ratchet

- DH and KDFs ratchets to derive session keys
- also provides Backward Secrecy
- sending and receiving ratchets
- SK derived from KDF ratchet
- DH ratchet pingpongs
- per device keys
- one DR per device pair
- Signal and OMEMO use a server for initial DH keys

\(^a\)Double Ratchet specification
Ditching the server

Signal/OMEMO:
- generate n DH keys on devices, publish
- initiator picks one from the server
- nasty race condition in OMEMO

our idea:
- ditch the server
  - sacrifice protecting the first mail
  - include initialization in a traditional OpenPGP encrypted message

  multiple devices
  - initiator generates all keys for one’s own devices
  - encrypts these keys with the per-device encryption subkeys
Alice has two devices, a laptop (L), and a phone (P). Bob has a desktop (D). Alice wants to send Bob a message from her laptop, they have not used the ratchet algorithm before. 3 ratchets: (L,D), (P,D), and (L,P).
Alice generates four DH pairs. Two for the laptop, two for the phone.

Alice sends a SEIP container with the message and the DH keys.

\[
\text{Alice}_L \rightarrow \text{Bob} \quad \text{DRInit}\{\text{pub: [DH}_{L,D}^{\text{pub}}, \text{DH}_{P,D}^{\text{pub}}], \text{sec: [Enc}_P(Sgn}_L(\text{DH}_{P,D}, \text{DH}_{P,L}, \text{DH}_{L,P}^{\text{pub}})))\}\]
Double Ratchet initialization II

- Bob generates two DH key pairs, initializes his ratchets.
- Bob sends his DH public key, and reflects all secrets.

Bob -> Alice

DRESK\{pub: \text{DH}_{D,L}^{\text{pub}}, \text{sec: Enc}_P(\text{Sgn}_L(\text{DH}_{P,D}, \text{DH}_{P,L}, \text{DH}_{L,P}^{\text{pub}})), \text{esk: ...}\}
DRESK\{pub: \text{DH}_{D,P}^{\text{pub}}, \text{sec: Enc}_P(\text{Sgn}_L(\text{DH}_{P,D}, \text{DH}_{P,L}, \text{DH}_{L,P}^{\text{pub}})), \text{esk: ...}\}
Alice’s phone decrypts the initial DH key pairs generated on the desktop, and uses them to initialize her ratchets.

The Double Ratchet algorithm initialization is now complete.

To send more messages, she advances her two phone ratchets by creating two new DH pairs.

\[ \text{Alice}_P \rightarrow \text{Bob} \]

\[
\begin{align*}
\text{DRESK}\{\text{pub: } & \text{DH}_{P,D}^{\text{pub}}, \text{sec: } \emptyset, \text{esk: } \text{Cipher}_{R(P,D)}(SK), \text{PN}, N_s} \\
\text{DRESK}\{\text{pub: } & \text{DH}_{P,L}^{\text{pub}}, \text{sec: } \emptyset, \text{esk: } \text{Cipher}_{R(P,L)}(SK), \text{PN}, N_s} 
\end{align*}
\]
Double Ratchet in OpenPGP

What is needed to implement Forward Secrecy using the Double Ratchet algorithm?

- per-device keys
- two new packets, DRIinit and DRESK
- keeping a lot of state in implementations

Juicy, but tricky. So let’s go for Brown’s short-lived encryption subkeys version first!

Ask questions! Get involved! Let’s get Forward Secrecy into OpenPGP!

Checkout our repository of weird keys\(^5\).

\(^5\)https://gitlab.com/sequoia-pgp/weird-keys
Users expect to be able to read past mails. Two options:

- store session keys
  - we (Sequoia) want to do that anyway for speed
  - compromise of session key store compromises messages
  - need to purge session key if message is deleted
    - *deletability*!
  - requires one-time synchronization for new devices

- re-encrypt with long-term archive key
  - not desirable if messages are on a server (IMAP)
Bonus: Privacy-preserving keys

- critical for revocations/key renewals/new keys
- traditional keyservers are problematic
  - expose the social graph
  - expose names/email addresses
- idea: strip 3rd-party-certificates, uids

### Compatibility

<table>
<thead>
<tr>
<th></th>
<th>Sequoia</th>
<th>GnuPG</th>
<th>OpenKeychain</th>
<th>openpgp.js</th>
<th>rnp</th>
</tr>
</thead>
<tbody>
<tr>
<td>null-uid</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>no-bound-uid</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>no-uid</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>direct-key</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>?</td>
<td>✓</td>
</tr>
</tbody>
</table>