Hybrid Quantum-Safe Key Exchange for IKEv2

CJ Tjhai, M Tomlinson, A Cheng
Post-Quantum

Graham Bartlett
Cisco Systems
Motivations

- The viability of quantum computers is an engineering problem
- Threats to Diffie-Hellman key-exchange in IKEv2
- There are a number of algorithms that are believed to be quantum-safe.
  - NIST call for proposals on quantum-safe algorithms
- FIPS compliant
• Optional key exchange payload carrying quantum-safe public data

• Goals:
  • To allow quantum-safe key exchange to be used alongside DH key-exchange while we are transitioning to a post-quantum era.
  • To keep the modifications to IKEv2 to a minimum
  • To maintain compatibility with IKEv2
  • To provide a path to phase out vulnerable DH key exchange in the future (if required)
IKE_SA_INIT exchange

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDR, SAi1, KEi, [QSKEi,]</td>
<td>HDR, SAr1, KEr, [QSKEr,]</td>
</tr>
<tr>
<td>Ni</td>
<td>Nr, [CERTREQ]</td>
</tr>
</tbody>
</table>

SKEYSEED = prf(Ni | Nr, g^ir | QSSS)
• Initiator understands QSKE, but Responder does not.

• Policy-based fallbacks: allows non quantum-safe SAs to be established
  • SAi payload contains a combination of proposals, some offer KE and the rest offer both KE and QSKE.
  • QSKE payload is marked NON-CRITICAL

• Policy-based fallback: only quantum-safe SAs are allowed
  • SAi payload contains no proposal that offers KE only;
  • QSKE payload is marked CRITICAL (if required)

• What current IKEv2 implementations will do if it receives a proposal containing a transform type that it doesn't understand?
Creating a new Child SA

HDR, SK{SA, Ni, [KEi,] [QSKEi,] TSi, TSr} -->

<-- HDR, SK{SA, Nr, [KEr,] [QSKEr,] TSi, TSr}

KEYMAT = prf+(SK_d, g^ir(new) | QSSS(new) | N_{i} | N_{r})

Rekeying a Child SA

HDR, SK{N(REKEY_SA), SA Ni, [KEi,] [QSKEi,] TSi, TSr} -->

<-- HDR, SK{SA, Nr, [KEr,] [QSKEr,] TSi, TSr}
Rekeying an IKE SA

HDR, SK\{SA, Ni, KEi, QSKEi,\} -->

<-- HDR, SK\{SA, Nr, KEr, QSKEr,\}

SKEYSEED = prf(SK_d(old), g^{ir(new)} | QSSS(new) | Ni | Nr)
Fragmentation Issues

- QSKE payload is large
- Fragmentation mitigation methods considered:
  - Send QSKE payload as part of IKE_AUTH
  - Immediate CHILD_SA rekeying right after IKE_AUTH
  - Use QSKE payload of small size in IKE_SA_INIT, e.g. SIDH; more options in CREATE_CHILD_SA
  - Introduce a new state between IKE_SA_INIT and IKE_AUTH that is for exchanging QSKE payload
  - TCP encapsulation
• Address fragmentations, crypto-agility and backward compatibility.

\[ \text{HDR, } \text{SAi, KEi, Ni, } N(QSKE) \rightarrow \]
\[ \text{HDR, SAr, N(COOKIE), } [N(QSKE)] \]

\[ \text{HDR, } N(\text{COOKIE}), \text{ SAi, KEi, } \]
\[ \text{Ni, QSKEi}(1/2) \rightarrow \]
\[ \text{HDR, QSKEi}(2/2) \rightarrow \]
\[ \text{HDR, SAr, KEr, Nr, QSKEr}(1/2) \]
\[ \text{HDR, QSKEr}(2/2) \]

IKE_AUTH exchange
• As a demonstration, check-out our forked of strongSwan: https://github.com/post-quantum/strongswan/blob/qske/README.QSKE.md

• Note that it is in qske branch