Quantum Resistant IKEv2 Update

draft-tjhai-ipsecme-hybrid-qsk-ikev2-03


IETF 104
Quantum Computers will make classical (EC)DH insecure

Quantum Safe Key Exchange methods (QSKE) are not well studied yet and currently no single QSKE method is trusted by cryptographers
  - besides most of QSKE methods have large public keys

The idea is to make it possible in IKEv2 to perform several different key exchanges in a row, combining classical KE methods with quantum safe ones
  - it is assumed that combination of QSKE methods of different types is more secure than any of them alone
Additional KEs are negotiated in IKE_SA_INIT and performed in a series of new INTERMEDIATE exchanges between IKE_SA_INIT and IKE_AUTH.

<table>
<thead>
<tr>
<th>Initiator</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDR(IKE_SA_INIT), SA, Ni, KEi, N --&gt;</td>
<td>HDR(IKE_SA_INIT), SA, Nr, KEr, N</td>
</tr>
<tr>
<td>HDR(INTERMEDIATE), SK {Ni2, KEi2} --&gt;</td>
<td>HDR(INTERMEDIATE), SK {Nr2, KEr2}</td>
</tr>
<tr>
<td>HDR(INTERMEDIATE), SK {Ni3, KEi3} --&gt;</td>
<td>HDR(INTERMEDIATE), SK {Nr3, KEr3}</td>
</tr>
<tr>
<td>HDR(IKE_AUTH), SK {IDi, AUTH, TSi, TSr} --&gt;</td>
<td>HDR(IKE_AUTH), SK {IDr, AUTH, TSi, TSr}</td>
</tr>
</tbody>
</table>

After each exchange the IKE SA keys are updated.

New SKEYSEED is computed as \( \text{prf}(\text{SK}_d(\text{old}), \text{KEn\_result} | \text{Nin} | \text{Nrn}) \).

Then, \( \text{SK}_d, \text{SK}_a, \text{SK}_r, \text{SK}_e, \text{SK}_r, \text{SK}_p, \text{SK}_p \) are updated as:

\[
\{ \text{SK}_d | \text{SK}_a | \text{SK}_r | \text{SK}_e | \text{SK}_r | \text{SK}_p | \text{SK}_p \} = \text{prf}+ (\text{SKEYSEED}, \text{Nin} | \text{Nrn} | \text{SPIi} | \text{SPIr})
\]

All INTERMEDIATE exchanges are authenticated in IKE_AUTH by inclusion of prf of their content in AUTH payload calculation.
Changes from -02 version

- Additional key exchanges are now negotiated using new Transform Types in SA Payload
- Using multiple key exchanges in CREATE_CHILD_SA is defined
- IKE_AUX is changed to INTERMEDIATE (to be aligned with draft-smyslov-ipsecme-ikev2-aux-02)
- IANA considerations section is added
- VendorID and temporary IDs for PQ KE methods are removed from the draft
QSKE Negotiation

• Seven new Transform Types are defined:
  o Additional Key Exchange 1
  o Additional Key Exchange 2
  ...
  o Additional Key Exchange 7

• All these Transform Types, as well as Transform Type 4, share the same Transform IDs registry – *Diffie-Hellman Group Transform IDs* (to be renamed to *Key Exchange Transform IDs*)

• QSKE methods will get code points from this registry (as well as classic (EC)DH groups)
QSKE Negotiation (2)

- If Initiator wants to do QSKE, he includes one or more transforms of type “Additional Key Exchange N” in the Proposal in SA Payload
- Transforms of these types contain Transform IDs identifying KE methods the Initiator proposes to perform in corresponding INTERMEDIATE exchanges
- The relative order of INTERMEDIATE exchanges is defined by N, so that KE from “Additional Key Exchange N” will be done before KE from “Additional Key Exchange N+1” etc.
QSKE Negotiation (3)

• There is no requirement that N in included transforms are contiguous (e.g. it’s OK to include only “Additional Key Exchange 2” and “Additional Key Exchange 5”)

• The Initiator may include NONE Transform ID in any of “Additional Key Exchange N” transforms, which means that it’s OK to completely skip INTERMEDIATE exchange for this N

• For compatibility with legacy implementations the Initiator may include two proposals – one with new Transform Types and the other – without them
QSKE Negotiation (4)

- Transform Type 4 (Diffie-Hellman Group Transform IDs) is always included and is always performed in the IKE_SA_INIT (no change from regular IKEv2)

- Since Transform Type 4 and Additional Key Exchange transforms share the same registry, it’s also possible to perform one QSKE in the IKE_SA_INIT
  - this allows in future to not perform the series of Key Exchanges if a cryptographically sound QSKE with small public key appears
Example of Initiator’s policy (perform ECP_521 in IKE_SA_INIT, then NewHope, then FRODO, then either RLWE or LWE and at the end SIDH or NTRU or nothing)

SA Payload

| --- Proposal #1 ( Proto ID = IKE(1), SPI size = 8, |
| | 10 transforms, SPI = 0x052357bbc763eb14 ) |
| --- Transform ENCR ( Name = ENCR_AES_GCM_16) |
| --- Transform PRF ( Name = PRF_HMAC_SHA2_256) |
| --- Transform D-H ( Name = DH_ECP_521) |
| --- Transform Additional KE 1 ( Name = KE_NEWHOPE ) |
| --- Transform Additional KE 3 ( Name = KE_FRODO ) |
| --- Transform Additional KE 4 ( Name = KE_RLWE ) |
| --- Transform Additional KE 4 ( Name = KE_LWE ) |
| --- Transform Additional KE 6 ( Name = KE_SIDH ) |
| --- Transform Additional KE 6 ( Name = KE_NTRU ) |
| --- Transform Additional KE 6 ( Name = NONE ) |

| --- Proposal #2 ( Proto ID = IKE(1), SPI size = 8, |
| | 3 transforms, SPI = 0x052357bbc763eb14 ) |
| --- Transform ENCR ( Name = ENCR_AES_GCM_16) |
| --- Transform PRF ( Name = PRF_HMAC_SHA2_256) |
| --- Transform D-H ( Name = DH_ECP_521) |
Using QSKE in CREATE_CHILD_SA

• If Initiator wants to use QSKE in case of rekeying IKE SA or creating/rekeying Child SAs, then there must be a way to do it with existing CREATE_CHILD_SA

• The idea to put all KEs in a single CREATE_CHILD_SA message is not good:
  o the message would become large in size; although this message could be fragmented, a single lost fragment would require the whole message to be resent
  o Initiator would need to calculate many public keys before KE methods are actually negotiated
  o INVALID KE PAYLOAD semantics would become different comparing to the regular IKEv2 case
Using QSKE in CREATE_CHILD_SA (2)

- Additional KEs are performed in a series of INFORMATIONAL exchanges followed CREATE_CHILD_SA exchange.
- New Notification ADDITIONAL_KEY_EXCHANGE is used to link these exchanges, because they can be interleaved with another IKE exchanges.
- QSKEs are negotiated in the same manner as in IKE_SA_INIT.
- New SA is created only when the last of INFORMATIONAL exchanges is complete.
Using QSKE in CREATE_CHILD_SA (3)

- Example:

Initiator

```
| HDR(CREATE_CHILD_SA), SK {SA, Ni, KEi} --> |
| HDR(INFORMATIONAL), SK {Ni2, KEi2, |
| N(ADDITIONAL_KEY_EXCHANGE)(link1)} --> |
| HDR(INFORMATIONAL), SK {Ni3, KEi3, |
| N(ADDITIONAL_KEY_EXCHANGE)(link2)} --> |
```

Responder

```
<-- HDR(CREATE_CHILD_SA), SK {SA, Nr, KEr, |
| N(ADDITIONAL_KEY_EXCHANGE)(link1)} |

<-- HDR(INFORMATIONAL), SK {Nr2, KEr2, |
| N(ADDITIONAL_KEY_EXCHANGE)(link2)} |

<-- HDR(INFORMATIONAL), SK {Nr3, KEr3} |
```
Next Steps

• Clarify collisions handling in CREATE_CHILD_SA in case of additional exchanges
• Clarify how keys are computed in CREATE_CHILD_SA with additional exchanges
• Update IANA Considerations: add request to rename *Diffie-Hellman Group Transform IDs* to *Key Exchange Transform IDs*
Thank you!

• Questions? Comments? Feedback?
• Requirements for QSKE methods?
• Document adoption?