Hybrid Key Agreement/KEM Construction and Integration to IPsec IKEv2 VPN

Basil Hess, Vladimir Soukharev, Derek Orpen / InfoSec Global

ETSI Quantum Safe Cryptography / Seattle / 7 Nov 2019
Cryptographic Threat | Quantum Computer

How to guarantee data confidentiality today for 10+ years?
PQ Hybrid Key Exchange

Record now decrypt later?
IPsec IKEv2 Handshake

The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange” (RFC 7296)
DH and KEM Key Agreement (1-RTT, ephemeral)

DH Key Agreement

1. Keygen \((pk_i, sk_i)\)

Initiator

2. Keygen \((pk_r, sk_r)\)

Responder

3. \(k = \text{DH}(pk_i, sk_r)\)

KEM Key Exchange

1. Keygen \((pk_i, sk_i)\)

Initiator

2. KEM Enc \(k, c = \text{Enc}(pk_i)\)

Responder

3. \(k = \text{Dec}(c, sk_i)\)

Algorithms: DH, ECDH, SIDH

Algorithms: All NIST PQC candidates
Hybrid DH/KEM Agreement

1-RTT Protocol

Initiator

1
Keygen DH
\((p_{k_{i,1}}, s_{k_{i,1}})\)

Keygen KEM
\((p_{k_{i,2}}, s_{k_{i,2}})\)

Responder

\(p_{k_{i,1}}\| p_{k_{i,2}}\)

2
Keygen DH
\((p_{k_{R,1}}, s_{k_{R,1}})\)

\(s_1 = DH(p_{k_{i,1}}, s_{k_{R,1}})\)

KEM Enc
\(s_2, c_2 = Enc(p_{k_{i,2}})\)

\(s = KDF(s_1|s_2)\)

3
\(s_1 = DH(p_{k_{R,1}}, s_{k_{i,1}})\)

\(s_2 = Dec(c_2, s_{k_{i,2}})\)

\(s = KDF(s_1|s_2)\)

Algorithms: Any combination of DH and KEM

Optimizations and extensions

- Multiple DH, multiple KEM
- Use combiner to preserve IND-CCA security with multiple KEMs
- Parallelize Keygen DH
- Avoid Keygen with static keys
IKEv2 with PQ crypto

SIKEp434: 330 bytes PK, 346 bytes ciphertext - OK

FrodoKEM640: 9616 bytes PK, 9720 bytes ciphertext - Fragmentation
IKEv2 Hybrid/PQ Fragmentation & Solutions

- IP layer UDP fragmentation – unreliable and often blocked
- IKE_SA_INIT – RFC 7296
  - Avoid packets larger than ~1500 bytes MTU
- IKE fragmentation – RFC 7383
  - Fragmentation on IKE layer
  - Only applies to encrypted messages, not IKE_SA_INIT
- IKE_INTERMEDIATE – IETF draft
  - Additional encrypted key exchange messages after IKE_SA_INIT.
  - IKE fragmentation applies to IKE_INTERMEDIATE messages
  - Doesn’t specify the key exchange method
- Hybrid qske based on IKE_INTERMEDIATE – IETF draft
  - Defines how key exchange is applied to IKE_INTERMEDIATE
IKEv2 Hybrid/PQ Fragmentation & Solutions

- IP layer UDP fragmentation – unreliable and often blocked
- IKE_SA_INIT – RFC 7296
  - Avoid packets larger than ~1500 bytes MTU
- IKE fragmentation – RFC 7383
  - Fragmentation on IKE layer
  - Only applies to encrypted messages, not IKE_SA_INIT
- IKE_INTERMEDIATE – IETF draft
  - Additional encrypted key exchange messages after IKE_SA_INIT.
  - IKE fragmentation applies to IKE_INTERMEDIATE messages
  - Doesn’t specify the key exchange method
- Hybrid exchange based on IKE_INTERMEDIATE – IETF draft
  - Defines how key exchange is applied to IKE_INTERMEDIATE
Setup and Design

• AgileSec VPN based on strongSwan with central management
• Custom plugin with support for:
  • “Keyagree”, “KEM”, “Hybrid”
  • Algorithm choice encoded in Kei/Ker payload
• Using ISG AgileSec library
  • Offers algorithm-agnostic APIs: e.g. Keyagree, KEM and Hybrid
  • Algorithm implementation with dynamic loadable “providers”
    1. Optimized PQ provider
    2. libOQS provider
    3. Standard provider
• PQC and Hybrid support for:
  • DH, ECDH, X25519, X448, SIKE, SIDH, FrodoKEM, Kyber, NewHope, BIKE, NTRU, Saber
## Hybrid DH/KEM – Message Sizes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>IKE_SA_INIT (I)</th>
<th>IKE_SA_INIT (R)</th>
<th>NIST</th>
<th>Sec</th>
<th>Cat</th>
<th>MTU 1500</th>
<th>MTU 576</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25519+SIKEp434comp</td>
<td>472</td>
<td>522</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X25519+SIKEp503comp</td>
<td>504</td>
<td>561</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X25519+SIKEp434</td>
<td>610</td>
<td>659</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+NTRU-HPS2048</td>
<td>979</td>
<td>1012</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+LightSaber</td>
<td>952</td>
<td>1049</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+Kyber_512</td>
<td>1080</td>
<td>1049</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+NewHope512</td>
<td>1208</td>
<td>1433</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+NTRU-HRSS</td>
<td>1418</td>
<td>1451</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+BIKE_L1_CCA</td>
<td>3226</td>
<td>3259</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X25519+FrodoKEM640</td>
<td>9896</td>
<td>10033</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X448+SIKEp610comp</td>
<td>577</td>
<td>634</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X448+SIKEp610</td>
<td>766</td>
<td>823</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X448+Kyber_768</td>
<td>1488</td>
<td>1425</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X448+Saber</td>
<td>1296</td>
<td>1425</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X448+SIKEp751comp</td>
<td>635</td>
<td>700</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X448+SIKEp751</td>
<td>868</td>
<td>933</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Hybrid+

<table>
<thead>
<tr>
<th>Scheme</th>
<th>IKE_SA_INIT (I)</th>
<th>IKE_SA_INIT (R)</th>
<th>Cat</th>
<th>MTU 1500</th>
<th>MTU 576</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25519+SIKEp434_c+Kyber_512</td>
<td>1272</td>
<td>1258</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X25519+SIKEp434_c+LightSaber</td>
<td>1144</td>
<td>1258</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X25519+SIKEp434_c+NTRU-HPS2048</td>
<td>1171</td>
<td>1221</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
VPN IKEv2 Performance

- Real world factors
  - Keyagree / KEM performance (Cat. 1, Core i7-8559U 2.7 GHz)
    - X25519: < 0.1 ms, Lattices: ~0.4 ms ... ~15 ms, Isogenies: 10 ms ... 17 ms
  - Network latency, RTT
  - Network bandwidth
  - Packet loss
  - Protocol/Network stack

- Overhead
  - Fast scenario: 1 .. 10 ms
  - Typical scenario / roadwarrior: ~50 ms
  - Intercontinental site-to-site scenario: > 100 ms
  - Slow scenario (high latency / low bandwidth): > 500 ms
  - IKEv2 handshakes usually less often performed than HTTPS/TLS, less user impact

Overhead >= 1 ms
Hybrid IKE_SA_INIT Latency Estimate

- X25519+SIKEp434_c
- X25519+SIKEp434
- X25519+Kyber_512

Bandwidth in Mbps (50 ms RTT)

IKE_SA_INIT handshake in ms
Conclusions

• Hybrid key exchange is the ideal PQ migration path, do it **now**!
• Feasibility of IKE_SA_INIT based hybrid key exchange
• With 1500 byte MTU
  • ECC-Isogenies, ECC-Lattices
  • Hybrid+ also feasible: ECC-Isogenies-Lattices
• With 576 byte MTU, the choice is more limited
  • ECC-SIKEp434 (compressed), ECC-SIKEp503 (compressed)
• Unstructured lattices / code-based schemes require IKE_INTERMEDIATE with IKE fragmentation support
• Adding ECC hybrid to PQ key exchange adds negligible overhead (messages and performance)
• Future work: larger field experiment, standards, IKE_INTERMEDIATE
Thank you!
Questions?