Key-Update Mechanism for SDLSP

Andreas Hülsing Tanja Lange **Fiona Weber**

TU/e

8. November 2023

Author list in alphabetical order, see
https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf.

Section 1

Authenticated Key Exchange

Motivation

- SDLSP secures communication with symmetric keys.
- These *can* be replaced, but the update uses only symmetric cryptography.
  - Cannot recover from corruption!
  - The total number of keys grows quadratically with the number of parties.
  - The keys that a party has to know up-front grows linearly.
- Future mega-constellations may massively increase the number of communicating parties.

Authenticated Key Exchange – In General

- Two parties, each with a long-term key-pair for authentication
- At least one party usually generates an ephemeral key-pair
  - Not used outside the exchange, secret-key disposed after exchange.
- The final output of an AKE is a shared secret that only the involved parties know.

Authenticated Key Exchange – In Our Use-Case

- Mission-Control and the Satellite both have a key-pair to authenticate themselves.
- They may have a previous shared secret. (The previous symmetric key)
- AKE computes a new shared secret that is secure even if the old one is leaked.
- Both parties can be certain of the identity of their peer.
- Can be run independently of a messaging-phase.

Advantages

- Total keys only scale *linearly* with the number of parties.
- Usable with a Public-Key-Infrastructure (PKI) – No need to preload all keys.
- Possible to recover from corruption.

Security-Goals

Confidentiality

Attacker does not learn information about resulting key.

- Forward-Secrecy: Even if he later corrupts a party.
- Post-Compromise-Secrecy: Even if he had corrupted the party before.
- Long-Term Security: Deal with “store-now, decrypt-later”-attacks.

Authenticity

Attacker cannot impersonate a different party.

- Prevent replay-attacks (common vulnerability).
- Good news: Attacks inherently have to be performed “live”.

Hybrid Security

- Use two schemes in case one is broken
- Typically EC-schemes, e.g. Hashed Diffie-Hellman using X25519 and ECDSA.
- Can be done on protocol or primitive-level
  - primitive-level is generally simpler
  - it also results in an primitive-agnostic protocol $\Rightarrow$ More options for implementers
- Fallback does not necessarily have to be pre-quantum!
- Combination trivial for Signatures.
- Less trivial for KEMs, but Hashing shared secrets and ciphertexts works.

Updating long-term keys

- Long-term keys may also get corrupted $\rightarrow$ should be updatable as well.
- Our protocol contains a mechanism for that.

Section 2

Possible Approaches

Signatures + KEM

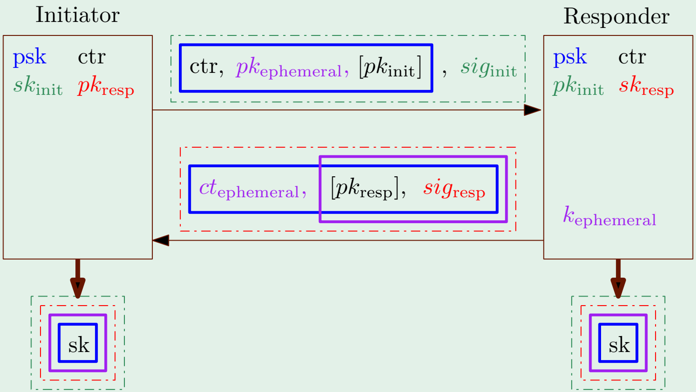

Figure 1: Signatures+KEM: The traditional Way.

- Requires replay-protection! (`ctr`)
- 1 Roundtrip
- Key-confirmation sensible, but not required.
- long-term-key-updates required if signature-scheme is stateful.
- Stateful scheme would enable few- and one-time signatures.

Triple-KEM and Dual-KEM

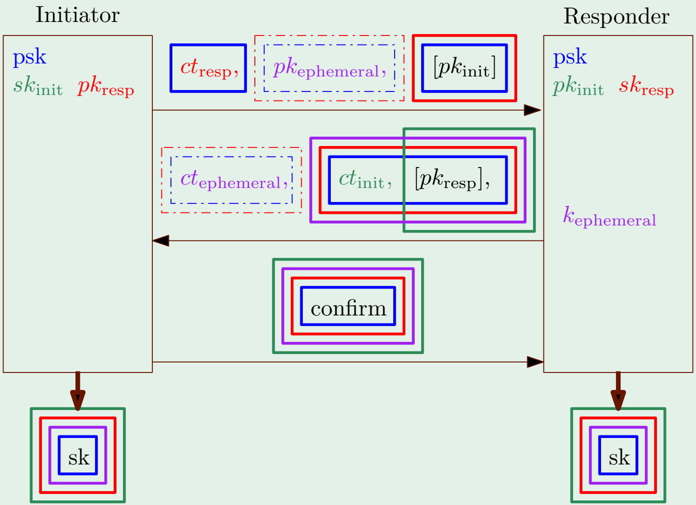

Figure 2: Triple-KEM: The more modern way.

- Usually more efficient (KEMs instead of signatures).
- Essentially invulnerable to replay-attacks.
- Option to mix KEMs.
- Dropping $\{ct, pk, sk\}_{\text{resp}}$ gives **Dual-KEM**, which does not authenticate the receiver.

Considered KEMs

- Obvious Choice: Kyber
- Ten times larger: Frodo
- Worth a look for special use-cases: Classic McEliece
- Not Size-Competitive with Kyber: BIKE and HQC
- Similar to Kyber, but lost PQC: Saber, NTRU, NTRU prime
- Broken: SIKE

Section 3

Our Recommendations

Our Recommendations

Our primary recommendation for general use is:

- **Triple-KEM**, using **Kyber** (and X25519) for all three KEMs

If satellite-authenticity is a given and the bandwidth-savings are important:

- **Dual-KEM**, using **Kyber** (and X25519) for both KEMs

Triple-KEM with Kyber

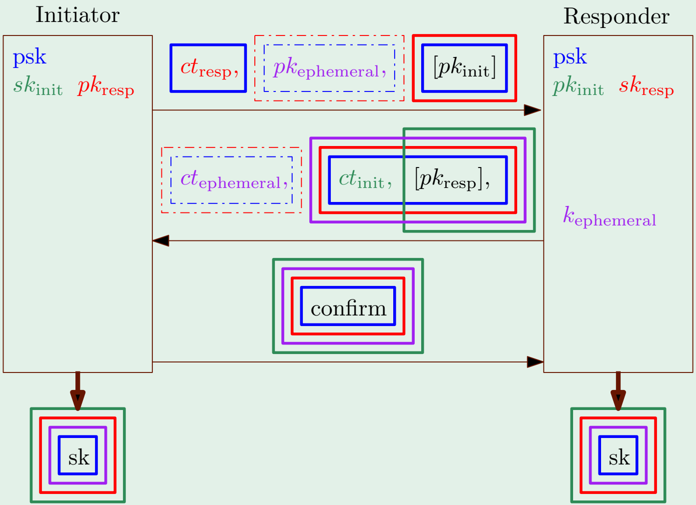

Figure 3: Triple-KEM

Packet sizes in bytes at different security-levels:

- Level 1: 1664, 1616, 16
- Level 3: 2368, 2256, 16
- Level 5: 3232, 3216, 16

With long-term-key updates:

- Level 1: 2496, 2464, 16
- Level 3: 3584, 3488, 16
- Level 5: 4832, 4832, 16

Security Analysis

We analyzed the protocol in a custom eCK-NEC model (= eCK, No Ephemeral Corruption)

- Simplified version of established eCK-model
- Assumes ephemeral randomness cannot be corrupted.
- Provides strong Confidentiality and Authenticity guarantees.

eCK-NEC

Security is usually defined via a “Game” in which an adversary tries to reach a winning-condition.

- n_i initiators and n_r responders run up to n_{s_i}/n_{s_r} initiator/responder-sessions each
- Adversary controls parties actions and the network
- Adversary can corrupt long-term keys and session-keys
- Winning conditions forbid trivial attacks
- Adversary wins
  - if he is able to distinguish an honestly generated key from randomness, or
  - if he is able to impersonate a party without corrupting its long-term-key.

Informal Security

Proven for Triple-KEM in eCK-NEC-model under reasonable assumptions:

- **Honestly generated keys are indistinguishable from randomness.** (Confidentiality)
- **A party cannot be impersonated, as long as its long-term public key remains uncorrupted.** (Authenticity)

Conjectured:

- Honestly generated keys remain confidential if the pre-shared key remains uncorrupted.
- Honestly generated keys remain confidential as long as one party's long-term key and the peer's ephemeral randomness remain uncorrupted.
- As long as a connection remains confidential (see above), no passive attacker can learn more about a new long-term public-key than can be extracted from ciphertexts for that public key. (Identity Hiding)

The same holds for **Dual-KEM**, ***if*** *responder-authenticity is guaranteed out-of-band*.

Formal Security Triple-KEM

There is no adversary that can win the eCK-NEC-game against Triple-KEM, with:

$$\mathsf{Adv}^{\mathsf{eCK\text{-}NEC}}_{\mathcal{A},\,\mathsf{3KEM}}\left(1^\lambda\right) \leq \left(\begin{array}{rrcl} & 3 & \cdot & \mathsf{Adv}^{\mathsf{coll\text{-}res}}_{\mathcal{A}_1,\,\mathsf{H}}\left(1^\lambda\right) \\ + & n_i \cdot n_{s_i} & \cdot & \mathsf{EKEM}.\delta \\ + & n_i \cdot n_{s_i} \cdot n_r \cdot n_{s_r} \cdot 3 & \cdot & \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\,\mathsf{EKEM}}\left(1^\lambda\right) \\ + & n_{s_r} \cdot n_i \cdot n_r \cdot \frac{1}{1-\mathsf{IKEM}.\delta} & \cdot & \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A}_4,\,\mathsf{IKEM}}\left(1^\lambda\right) \\ + & n_{s_i} \cdot n_i \cdot n_r \cdot \frac{1}{1-\mathsf{RKEM}.\delta} & \cdot & \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A}_4,\,\mathsf{RKEM}}\left(1^\lambda\right) \\ + & n_i \cdot n_{s_i} \cdot n_r \cdot n_{s_r} \cdot 3 & \cdot & \mathsf{Adv}^{\mathsf{PRHO}}_{\mathcal{A},\,\mathsf{NHO}}\left(1^\lambda\right) \\ + & \left(n_{s_i} + n_{s_r}\right) \cdot n_i \cdot n_r & \cdot & \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}_6,\,\mathsf{AEAD}}\left(1^\lambda\right) \\ + & n_i \cdot n_{s_i} \cdot n_r \cdot n_{s_r} \cdot 2 & \cdot & \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A},\,\mathsf{KDF}}\left(1^\lambda\right) \end{array}\right)$$

Formal Security Dual-KEM

There is no adversary that can win the eCK-NEC-game against Dual-KEM, with:

$$\mathsf{Adv}^{\text{eCK-NEC}}_{\mathcal{A},\,\text{2KEM}}\left(1^{\lambda}\right) \leq \left(\begin{array}{rrcl} & 2 & \cdot & \mathsf{Adv}^{\text{coll-res}}_{\mathcal{A}_1,\,\text{H}}\left(1^{\lambda}\right) \\ + & n_i \cdot n_{s_i} & \cdot & \text{EKEM}.\delta \\ + & n_i \cdot n_{s_i} \cdot n_r \cdot n_{s_r} \cdot 3 & \cdot & \mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{A},\,\text{EKEM}}\left(1^{\lambda}\right) \\ + & n_{s_r} \cdot n_i \cdot n_r \cdot \frac{1}{1-\text{IKEM}.\delta} & \cdot & \mathsf{Adv}^{\text{IND-CCA}}_{\mathcal{A}_4,\,\text{IKEM}}\left(1^{\lambda}\right) \\ + & n_i \cdot n_{s_i} \cdot n_r \cdot n_{s_r} \cdot 2 & \cdot & \mathsf{Adv}^{\text{PRHO}}_{\mathcal{A},\,\text{NHO}}\left(1^{\lambda}\right) \\ + & n_{s_r} \cdot n_i \cdot n_r & \cdot & \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}_6,\,\text{AEAD}}\left(1^{\lambda}\right) \\ + & n_i \cdot n_{s_i} \cdot n_r \cdot n_{s_r} \cdot 2 & \cdot & \mathsf{Adv}^{\text{PRF}}_{\mathcal{A},\,\text{KDF}}\left(1^{\lambda}\right) \\ + & & & \mathsf{Adv}^{\text{eCK-NEC}_{\text{Case A}}}_{\mathcal{A},\,\text{2KEM}}\left(1^{\lambda}\right) \end{array}\right)$$

Where $\mathsf{Adv}^{\text{eCK-NEC}_{\text{Case A}}}_{\mathcal{A},\,\text{2KEM}}\left(1^{\lambda}\right)$ Refers to the maximum achievable advantage for the adversary to cause an unpeered, complete initiator-session.

Discussion

- We worked under the assumption that there are only very few initiators, because there are not many mission-control-centers.
    - Analysis deals with **all** users of a protocol, if this protocol is used widely that has to include everyone who controls a Satellite.
- Our model does not consider the possibility to corrupt ephemeral randomness.
    - In our experience most practitioners tend to believe that the solution to broken RNGs are not mitigations on the protocol-level, but rather fixing them on the system-level.

Conclusion

- Enable asymmetric key-updates for better scaling and security.
- Use post-quantum-secure algorithms for long-term security.
- Use an Authenticated Key Exchange (AKE) as Key-Update Mechanism
- Our Recommendation: Triple-KEM with Kyber+X25519
- Proposal builds on Post-Quantum Noise
- Formal Security-analysis in a simpler version of a standard model.

Section 4

Appendix

KEMs – Sizes and Failure-rates

Scheme	SK	PK	CT	δ
X25519	32	32	32	0
Kyber-512	1632	800	768	2^{-139}
Kyber-768	2400	1184	1088	2^{-164}
Kyber-1024	3168	1568	1568	2^{-174}
mceliece348864	6492	261120	96	0
mceliece460896	13608	524160	156	0
mceliece6688128	13932	1044992	208	0
mceliece6960119	13948	1047319	194	0
mceliece8192128	14120	1357824	208	0
FrodoKEM-640	19888	9616	9720	$2^{-138.7}$
FrodoKEM-976	31296	15632	15744	$2^{-199.6}$
FrodoKEM-1344	43088	21520	21632	$2^{-252.5}$

Signatures – Sizes

Scheme	SK	PK	Sig
Dilithium2	2544	1312	2420
Dilithium3	4016	1952	3293
Dilithium5	4880	2592	4595
Falcon-512	1281	897	666
Falcon-1024	2305	1793	1280
ECDSA	32	32	64

Triple-KEM – Packet Sizes

Scheme	Packet 1	Packet 2	Packet 3
TK(Kyber512+X25519)	1664	1616	16
TKU(Kyber512+X25519)	2496	2464	16
TK(Kyber768+X25519)	2368	2256	16
TKU(Kyber768+X25519)	3584	3488	16
TK(Kyber1024+X25519)	3232	3216	16
TKU(Kyber1024+X25519)	4832	4832	16

Sign + KEM – Packet Sizes

Scheme	Packet 1	Packet 2	Packet 3
SK(Kyber512+X25519+Dilithium+ECDSA)	3348	3300	16
SKU(Kyber512+X25519+Dilithium+ECDSA)	4692	4644	16
SK(Kyber512+X25519+Falcon+ECDSA)	1594	1546	16
SKU(Kyber512+X25519+Falcon+ECDSA)	2523	2475	16
SK(Kyber512+X25519+XMSS-SHA2_10_256)	3364	3316	16
SKU(Kyber512+X25519+XMSS-SHA2_10_256)	3428	3380	16
SC(Kyber512+X25519,WOTS+(32,16))	3024	2992	16
SC(Kyber768+X25519,WOTS+(32,16))	2408	3312	16
SC(Kyber1024+X25519,WOTS+(32,16))	3792	3792	16
SC(Kyber1024+X25519,WOTS+(64,16))	10032	10032	16

Section 5

Old Slides

Signatures + KEM

- The traditional way of doing things.

```
-> psk, ctr, e, s'[opt1], sig
<- ekem, s'[opt2], sig
```

- Requires replay-protection! (`ctr`)
- 1 Roundtrip
- Key-confirmation sensible, but not required.
- long-term-key-updates required if signature-scheme is stateful.

Signature-Chain

- Use One-Time Signatures and always update the long-term key.
- No case-distinction.
- strong Post-Compromise-Authenticity!

```
-> psk, e, s', sig
<- ekem, s', sig
```

Triple-KEM

The modern way of doing things.

```
-> psk, skem, e, s' [opt1]
<- ekem, skem, s'[opt2]
-> confirm
```

- Usually more efficient (KEMs instead of signatures).
- Essentially invulnerable to replay-attacks.
- Option to mix KEMs.

Considered KEMs

- Obvious Choice: Kyber
- Ten times larger: Frodo
- Worth a look for special use-cases: Classic McEliece
- Not Size-Competitive with Kyber: BIKE and HQC
- Similar to Kyber, but lost PQC: Saber, NTRU, NTRU prime
- Broken: SIKE

Considered Signatures (1)

- Obvious Choice: Dilithium
- Serious Contender: Falcon

Scheme	SK	PK	Sig
Dilithium2	2544	1312	2420
Dilithium3	4016	1952	3293
Dilithium5	4880	2592	4595
Falcon-512	1281	897	666
Falcon-1024	2305	1793	1280
ECDSA	32	32	64

- Broken: Rainbow
- Weakened and lost PQC: GeMSS
- No clear advantage over SPHINCS+ (next slide) and lost PQC: Picnic

Hash-Based Signatures

- SPHINCS+ is essentially unusable here
- XMSS and LMS may be worth a thought
  - stateful signature-schemes
- WOTS too.
  - one-time signature scheme.

Sign+KEM

Table 6: SK = Sign+KEM, SC = Signature-Chain, TK = Triple-KEM

Scheme	Packet 1	Packet 2
SK(Kyber512+X25519+Dilithium+ECDSA)	4692	4644
SK(Kyber512+X25519+Falcon+ECDSA)	2523	2475
SK(Kyber512+X25519+XMSS-SHA2_10_256)	3428	3380
SC(Kyber512+X25519,WOTS+(32,16))	3024	2992
TK(Kyber512+X25519)	2496	2464

Prefered Protocol and Security-Level

- All primitives can be changed to provide whatever security-level is desirable for them.
- Unless the reason for higher security-levels are brute-force attacks, different levels possibly quite reasonable.

$\Rightarrow$ Generally Level 1, sometimes Level 3

HMAC

- HMAC widely used as dual-PRF/split-PRF.
- Secure, but useless if hashfunction is a Random Oracle.
- Several used primitives assume that it is.
- Not proven to be secure otherwise.
- No known practical attacks.

Payload Encryption

- Noise encrypts long-term public keys and signatures
- Primary purpose: Identitiy hiding $\rightarrow$ Irrelevant here
- Overhead is comparatively small, but not zero.
- No analysis for case without encryption.
  - Relevant proofs do not rely on the encryption though.

Authentication Tag

- AES-GCM uses a ≤ 128 bit tag for authentication
- *Technically* limits authenticity to 128 bit, though likely irrelevant in practice.
- CCSDS recommends 256 bit keys, but 128 bit tags.

Stateful Signatures

- State-Reuse can effectively leak the secret key.
- Keys have to be stored securly on the satelite in the first place.
- How much can the control-center be trusted to manage its keys well?
- Is that need for trust worth the gain?