E-Laun: OTAR resistant to evil launchers

Alexandre Duc, Grégoire Guyot, Pascal Perrenoud

University of Applied Sciences and Arts Western Switzerland (HES-SO), Switzerland

November 5, 2025

Table of content

Section 1: Introduction

Context

- When operating satellites, **secure communication** between the ground segment and the spacecraft is critical.
- This requires the use of **cryptographic mechanisms** (e.g encryption, authentication, and key management).
- But how are these **keys** actually established, refreshed, or replaced once the spacecraft is in orbit?

Standard: SDLS

What is SDLS?

SDLS (Space Data Link Security) is the CCSDS standard that specifies how to secure communications between ground and spacecraft.

- SDLS defines frame protection services: **confidentiality**, **integrity** and **anti-replay**.
- It also specifies a **key update mechanism** and how keys are bound to Security Associations (SAs).
- In practice, SDLS describes how to **generate new symmetric keys** and how to **use them** within protected channels.

Over-The-Air Rekeying (OTAR)

Definition

Over-The-Air Rekeying (OTAR) is the process of securely updating cryptographic keys through an existing communication channel without physically accessing the spacecraft.

- It allows mission operators to **replace or refresh keys remotely**, even after launch.
- This ensures continued **confidentiality**, **integrity**, and **control** over communications.
- OTAR is essential for:
  - Long-duration missions,
  - Multi-satellite constellations,
  - Rapid key compromise recovery.

BUT...

In SDLS, OTAR relies entirely on **symmetric cryptography** meaning that both sides must already share the same secret key. This creates serious operational and security limitations.

Limitations of symmetric OTAR

- SDLS currently supports **symmetric-only** rekeying.
- As it relies on symmetric cryptography, it is easy to implement and offers good performance (and is PQ...).
- Security relies entirely on a **preloaded master key**.
- This creates operational risks:
  - No forward secrecy - if the master key is compromised, all sessions are exposed.
  - Key renewal requires secure ground updates.
  - In constellations, with sat-sat communications, the number of keys grows exponentially.

Goal: enable dynamic and secure rekeying *without relying on pre-shared master keys.*

From symmetric to asymmetric OTAR

- We extend SDLS with **asymmetric (Diffie–Hellman–based)** mechanisms.
- Fresh symmetric session keys are derived **on demand**, not preloaded.
- This provides:
  - **Forward secrecy** and compromise recovery.
  - Rekeying without secure pre-distribution.
  - Seamless integration into existing SDLS operations.
  - Reduced key management overhead - ideal for constellations.
- These principles lead to two **DH-based OTAR protocols.**

Contributions recap

- Specification of protocols built on Diffie-Hellman and a KDF.
- Analysis of their security properties.
- Definition of two new Extended Procedures (EPs) for integration into SDLS.
- Definition of Evil Launchers and analysis of their impact on our protocols.
- Preliminary work towards a **post-quantum transition**.

Section 2: Design proposal

Constraints

Before designing the protocols, we identified a set of key constraints:

Technical constraints

- Limited resources and space on the satellites.
- Entropy can be scarce in such hardware.
- SDLS compliance; only extended procedures as acceptable hooks.
- Tolerate loss and avoid interactive handshakes.

Security constraints

- Authenticity of all the entities.
- Anti-replay protection.
- Perfect forward secrecy.
- Evil launchers resistance!

High-level overview I

There are two main steps in asymmetric OTAR:

1. Setup
2. Asymmetric OTAR

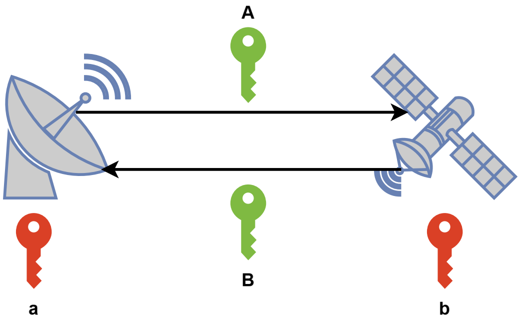
High-level overview II

Setup: Both protocols rely on an *identity* keypair. We first need to generate and distribute them *safely*.

1. Generate an identity keypair. Both sides generate a static, long-term *identity* keypair.
2. Safely distribute the public key to all the other parties.

Then the ground stations and satellites store:

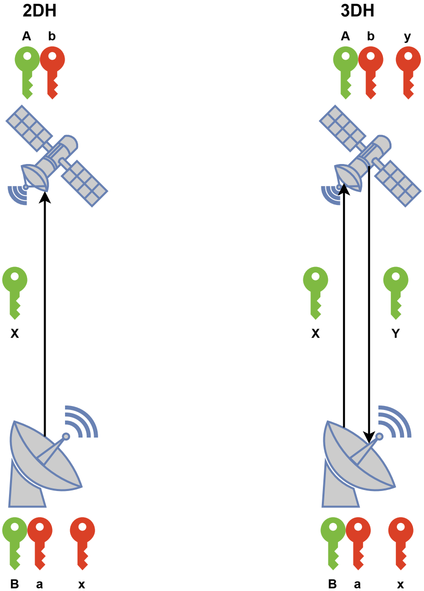
- Ground: 1x private key + 1x public key / satellite
- Satellite: 1x private key + 1x ground's public key



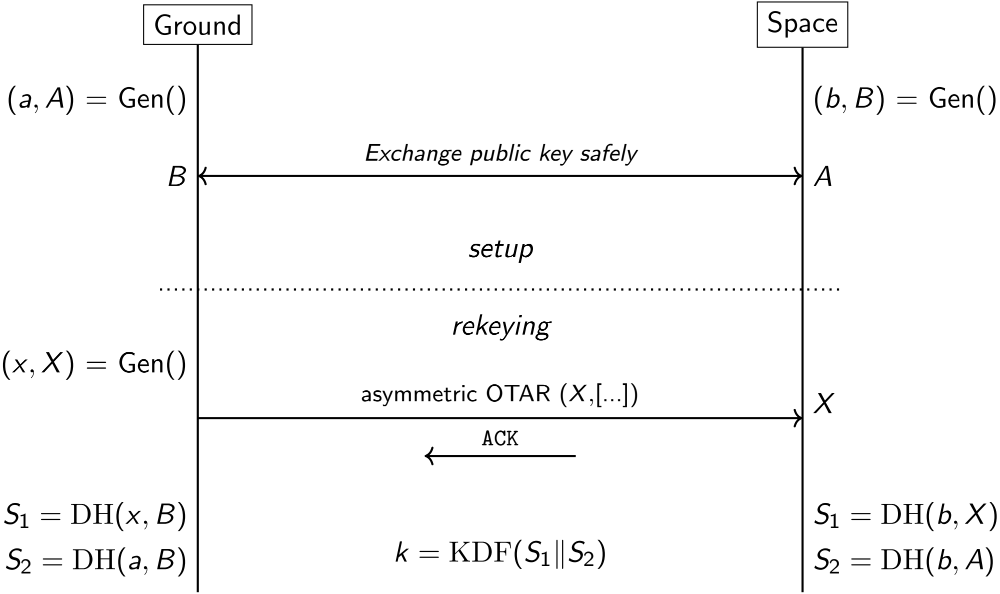
High-level overview III

Asymmetric OTAR:

1. The ground generates an ephemeral key.
2. It sends an *asymmetric OTAR* EP to the satellite.
3. When the satellite receives the EP, it:
   - Generates an ephemeral key (3DH only).
   - Derives the shared secret.
4. 3DH only: later, the ground sends another EP to retrieve the ephemeral public key of the satellite.
5. The ground can derive the shared secret.

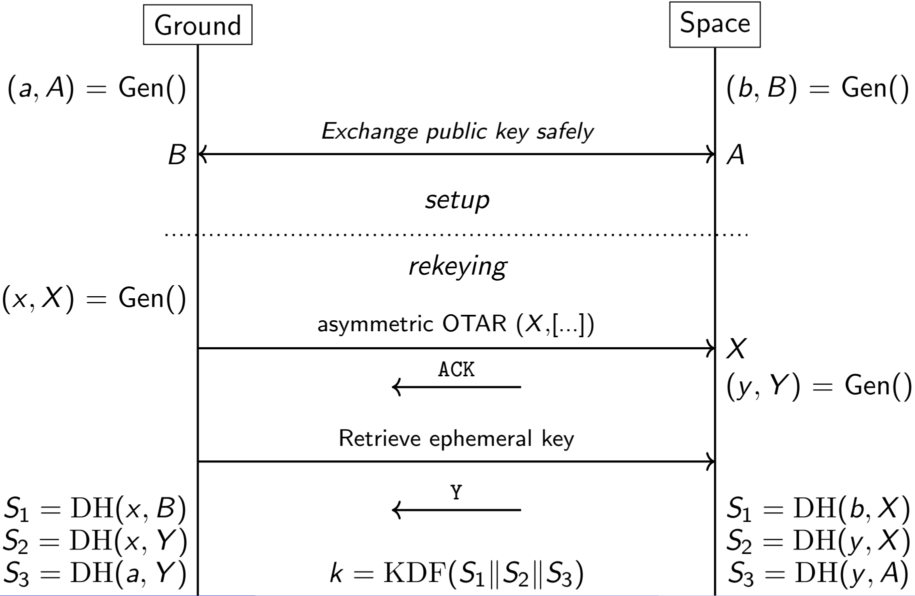


2DH with EPs



Ground
Space
$(a, A) = \text{Gen}()$
$(b, B) = \text{Gen}()$
Exchange public key safely
B
A
setup
rekeying
$(x, X) = \text{Gen}()$
asymmetric OTAR (X,[...])
X
ACK
$S_1 = \mathrm{DH}(x, B)$
$S_2 = \mathrm{DH}(a, B)$
$k = \mathrm{KDF}(S_1 \| S_2)$
$S_1 = \mathrm{DH}(b, X)$
$S_2 = \mathrm{DH}(b, A)$

3DH with EPs



Ground
Space
(a, A) = Gen()
(b, B) = Gen()
Exchange public key safely
B
A
setup
rekeying
(x, X) = Gen()
asymmetric OTAR (X,[...])
X
ACK
(y, Y) = Gen()
Retrieve ephemeral key
Y
S1 = DH(x, B)
S2 = DH(x, Y)
S3 = DH(a, Y)
k = KDF(S1||S2||S3)
S1 = DH(b, X)
S2 = DH(y, X)
S3 = DH(y, A)

2DH or 3DH

The 3DH construction is strictly more secure than 2DH **BUT** it requires two things:

- Randomness on the satellite.
- Two-ways communications.

If one of those two constraints is a no-go, we can use 2DH.

Issues of 2DH

The 2DH construction has the following issues:

- There is no forward-secrecy on the satellite.
- There is a risk of replay attack.
  - This can be mitigated using a key derived from static keypairs[1].
- There is also a risk regarding Evil Launchers, as we will see later.

[1] We analyzed this in the paper, Section "IV.E Bootstrapping"

Extended procedures

Asymmetric OTAR

Initiate rekeying by sending an ephemeral public key and the list of the IDs to create in the keystore.

Retrieve ephemeral key

Retrieve the ephemeral key generated on the satellite. (Only for **3DH!**)

Summary

Property	2DH	3DH
Authenticity	static keys	static keys
Round-trips	0	1
Forward secrecy	no	yes
Randomness	only ground	both
Replay protection	via static bootstrap	yes

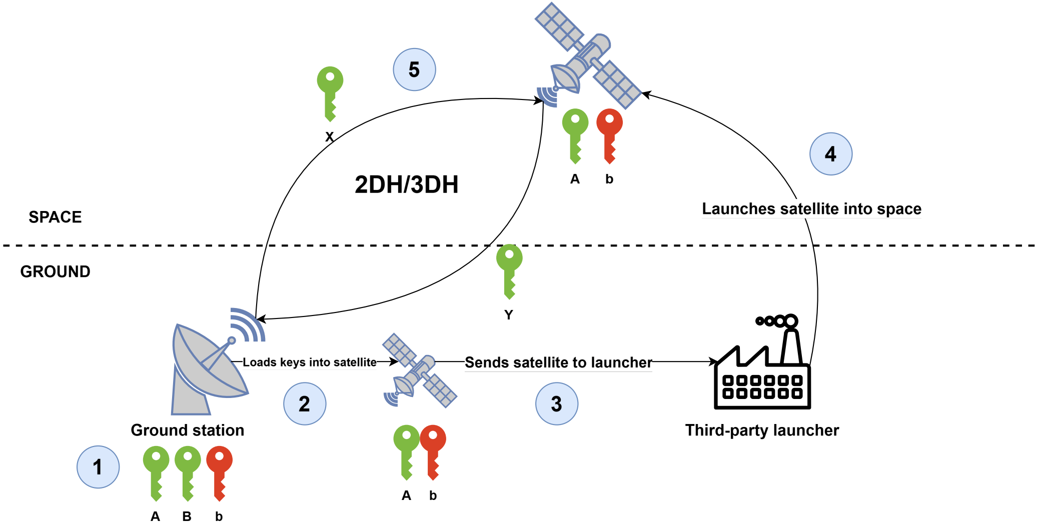
2DH is lighter and works without satellite randomness, while 3DH provides full forward secrecy and replay protection.

Section 3: Evil Launchers (E-Laun)

E-Laun consideration

- Using third-party launchers is a common practice when launching satellites into orbit.
- This can be a security issue as the satellite may contain sensitive data.
- Therefore, a security analysis is required!

Scenario



5
X
2DH/3DH
A
b
4
Launches satellite into space
SPACE
GROUND
Y
Loads keys into satellite
Sends satellite to launcher
2
3
Ground station
Third-party launcher
1
A
B
b
A
b

Adversarial model

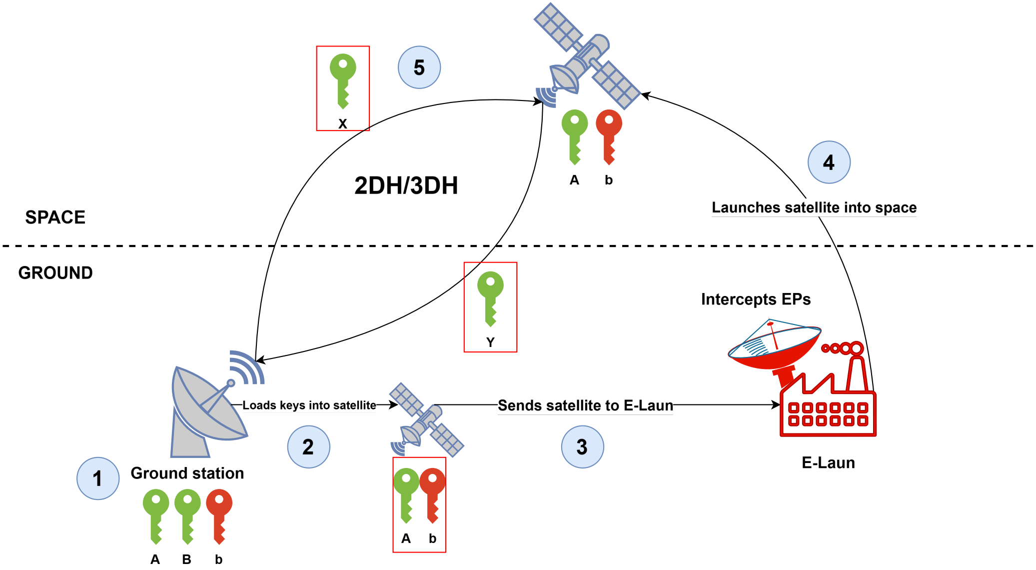
Considered capabilities

- E-Laun can **read** everything on the satellite between manufacturing and launch.
- E-Laun can **listen** to every communication between the ground and the satellite.
- E-Laun can send messages to the satellite.
- If it does not disrupt the satellite's activity, E-Laun can add content to it.

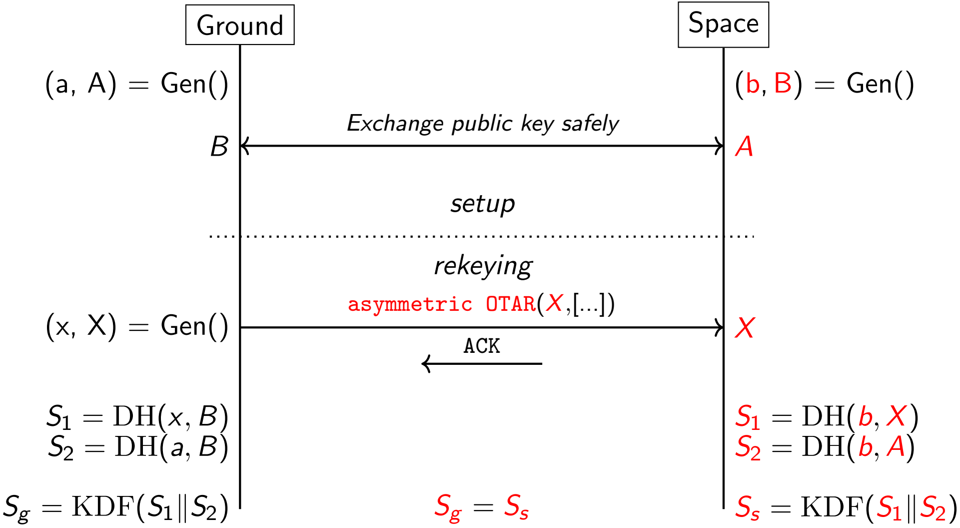
Not considered

- E-Laun performs a MITM between the satellite and the ground.
- E-Laun overwrites content in the satellite before launching it in space.

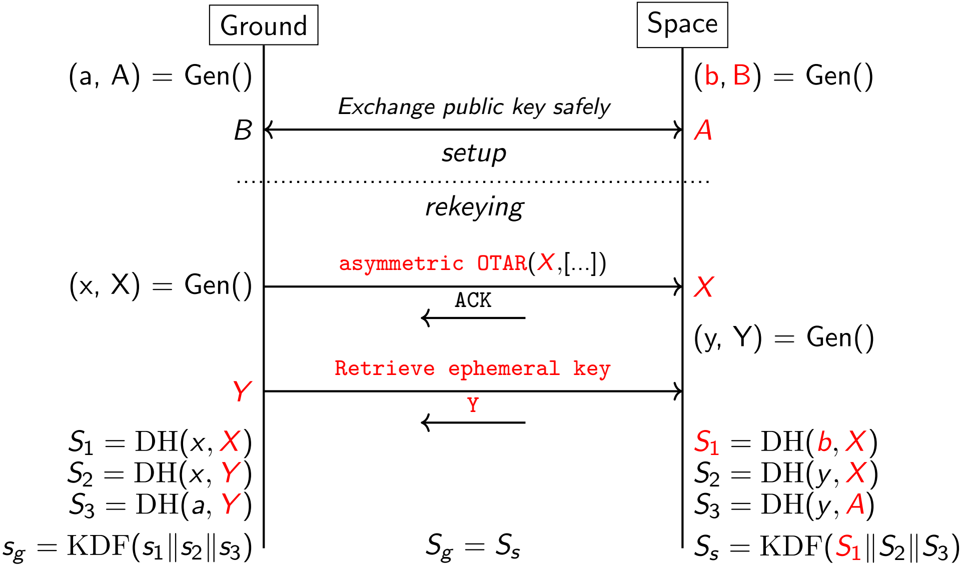
Attack scenario



E-Laun and 2DH



E-Laun and 3DH



Attack conditions summary

The attack works under two conditions:

1. E-Laun copies the private key of the satellite.
2. E-Laun captures a handshake.

The Evil Launcher **cannot** initiate a 2DH procedure with the satellite as it does not have the ground's private key.

Summary

3DH

3DH remains secure even if the third-party launcher is malicious.

2DH

- A **leak of the static secret key** due to an E-Laun would be catastrophic, as the attacker could recover past and future shared secrets from recorded handshakes.
- This attack scenario occurs if the launcher gains access to the spacecraft's long-term private key before deployment.

Section 4: Work in progress

Work in progress

1. Post-quantum transition
2. Key distribution

Subsection 4.1: Post-Quantum

Post-Quantum

- Our 2DH/3DH protocols are obviously not post-quantum...
- Adaptation is required (kind of quickly[2]).
- As Diffie-Hellman has no direct post-quantum alternative, the natural replacement is to use KEMs.
- A previous work already analyzed similar protocols with similar constraints (Double-KEM, Triple-KEM, signature+KEM) .
- Signature algorithms are only needed for explicit authentication; key exchanges alone can remain purely KEM-based.
  - But KEM + signature needs to be considered too.

[2] NIST depreciates pre-quantum algorithms in 2030 and will disallow them by 2035

Algorithm choices - Signatures

- **ML-DSA** (CRYSTALS-Dilithium): NIST's primary recommendation.
- **FN-DSA** (Falcon): the standard is still being finalized, but it is a very competitive alternative. But it relies on floating point arithmetics...
- **SLH-DSA** (SPHINCS+): not considered here due to its large signatures and slow performance.

Algorithm choices - KEMs

- Among the standardized post-quantum algorithms, we consider:
  - **ML-KEM** (formerly CRYSTALS-Kyber) for key exchange.
- **HQC** has also been selected for standardization.
- ML-KEM is generally faster and has better sizes than HQC.

ML-KEM vs classical

Sizes of ML-KEM[3] and X25519

ML-KEM-512

Parameter	Size (bytes)
Secret key (sk)	1632
Public key (pk)	800
Ciphertext (message)	768

Approx. 128-bit security

X25519

Parameter	Size (bytes)
Secret key (sk)	32
Public key (pk)	32
Shared secret (message)	32

Approx. 128-bit classical security

ML-KEM offers post-quantum security but with significantly larger key and ciphertext sizes.

[3] https://pq-crystals.org/kyber/

Subsection 4.2: Key Distribution

Key distribution

- As mentionned, we suppose a safe distribution of the identity keys.
- As seen earlier, it is not a trivial task for Ground-Satellite communication (because of the E-Laun).
- It is also non-trivial for Satellite-Satellite communications.
- We consider and analyze different solutions:
    - Centralized management of the keys.
    - PKI.
    - Lightweight PKI.

Section 5: Conclusion

Conclusion

We have shown that:

- 2DH and 3DH provide an **asymmetric OTAR mechanism compatible with SDLS**.
- 3DH offers **stronger security guarantees** than 2DH, but its reliance on onboard randomness can make 2DH more practical in constrained environments.
  - If 3DH is implemented but randomness is not guaranteed, its security effectively **degrades to that of 2DH**.
- **Evil Launchers** represent a realistic threat, yet their impact can be effectively mitigated through 3DH.

But we still need to:

- Concretize a post-quantum alternative.
- Find ways to safely distribute keys among satellites.

Thank you for your attention!

Any question?