E-Laun: OTAR resistant to evil launchers

Alexandre Duc, Grégoire Guyot, Pascal Perrenoud
School of Engineering and Management Vaud, HES-SO University of Applied Sciences and Arts Western Switzerland
Yverdon-les-Bains, Switzerland

Emails: {alexandre.duc, gregoire.guyot, pascal.perrenoud}@heig-vd.ch

***Abstract*—The Consultative Committee for Space Data Systems (CCSDS) link security depends on long-lived symmetric keys. Although the Space Data Link Security Protocol (SDLS) *does* support in-orbit rekeying, it does so only with symmetric techniques, i.e., by encrypting new keys under a pre-shared master key or by deriving keys from it, leaving missions exposed should that master key be compromised. We adapt well-known Diffie-Hellman handshakes to SDLS and propose two protocols called *Double* and *Triple Diffie–Hellman* (2DH/3DH) to extend SDLS with asymmetric rekeying without altering its frame format. In 2DH, the ground station sends a single ephemeral public key and derives fresh symmetric keys with the satellite's static key, staying within minimal on-board resources. The 3DH variant lets both ends contribute with ephemeral secrets, providing perfect forward secrecy at the cost of one extra communication, an extra scalar multiplication and the need for a good entropy source on the spacecraft. Both protocols result in replay-safe, over-the-air rekeying mechanisms that fit the bandwidth and storage limits of small satellites, thereby eliminating dependence on any long-term symmetric secret. We also analyzed the implications of our protocols with respect to threats from potentially malicious launch operators, referred to as *evil launchers*, who might gain unauthorized access to spacecraft cryptographic key material before deployment. We show how our protocols remain secure against such threats. Moreover, given the future risks posed by quantum computing, we conducted a preliminary state-of-the-art review of post-quantum cryptographic (PQC) algorithms. Our analysis identified suitable PQC algorithms in such a use case. Future work will focus on integrating these PQC algorithms within the proposed rekeying framework, preparing space communication for long-term resilience.**

***Index Terms*—Space security, CCSDS, SDLS, Key agreement, Extended procedures, Over-the-air rekeying, Evil launchers, Post-quantum cryptography**

I. Introduction

Satellite links secured under the Consultative Committee for Space Data Systems (CCSDS) standards rely exclusively on pre-shared symmetric keys. In the *Space Data Link Security Protocol* (SDLS), as defined in [1], the built-in key-update mechanism also uses only symmetric primitives, so a single compromised master key endangers the entire mission: once exposed, the key must be re-generated and laboriously replaced, halting operations and consuming valuable time. While SDLS provides frame-level confidentiality, integrity, and an anti-replay counter, it remains agnostic about *how* those symmetric keys are established, refreshed, or retired once the spacecraft is on orbit. The Blue Book on cryptographic primitives [2] does list some asymmetric algorithms but stops short of specifying *when*, *how* or *in which message flows* they should be invoked. Mission operators therefore face two unsatisfactory choices: preload a large bank of keys during integration, or periodically uplink new keys in the clear—both approaches expose the system to long-term key compromise, initialization vector (IV) exhaustion after power resets, and operational delays when key banks run out.

This project is funded by Innosuisse, under grant 108.459 IP-ICT.

To close this gap, we investigate lightweight asymmetric *over-the-air rekeying* (OTAR) mechanisms that fit within the severe resource constraints of small satellites yet interoperate seamlessly with existing SDLS tooling. Our design goal is to let a spacecraft derive fresh symmetric keys *on demand*, using only a few additional frames and cryptographic primitives that add marginal code and memory overhead. In this paper, we restrict our scope to the *ground-to-satellite* link—that is, a single ground station communicating with a spacecraft over a Telecommand (TC) uplink and a Telemetry (TM) downlink. Inter-satellite links, multi-ground scenarios, and constellation-wide group-key management are left for future work. In this work, we specifically address asymmetric rekeying for the single ground-to-satellite communication scenario, presenting two lightweight Diffie–Hellman-based protocols. With a static keypair pre-shared in both cases, the protocols go as follows:

- *Double Diffie–Hellman (2DH).* Derived from the Noise Protocol [3], the 2DH protocol starts with the ground station sending a single ephemeral public key to the spacecraft; the resulting shared secret provides partial forward secrecy, meaning that should the satellite's long-term secret become compromised, the security of past communication sessions would be jeopardized as each key derivation involves an ephemeral contribution only from the ground. Although the forward secrecy definition concerns past communications, in our protocol it would also compromise future communications. This approach imposes less computational burden on the satellite and requires no satellite-side entropy.
- *Triple Diffie–Hellman (3DH).* In this variant, described by Kudla and Paterson [4], both parties contribute with fresh ephemeral keys, yielding perfect forward secrecy at the cost of one extra communication, a third scalar multiplication on the spacecraft and a good entropy source on the satellite. The 3DH protocol also allows to come back to a safe state in case of the compromise of a private key.

A. Contributions

1) We specify 2DH and 3DH protocols built on Diffie-Hellman and a Key derivation function, and analyze their security properties under the satellite threat model.
2) We formalize a *static bootstrap* that authenticates the parties once and seeds an SDLS *Secure Association*, so that subsequent frames inherit built-in replay protection. This *static bootstrap* is only used in our 2DH handshake.
3) We define two new Extended Procedures (EPs), compliant with the guidelines of [5], that integrate our rekeying protocols seamlessly into the SDLS framework.
4) We analyze the post-quantum cryptography state of the art and we discuss the adaptations of our protocols.
5) In the context of satellite launches, we define an *evil launcher* as a third-party launch provider who may be *honest-but-curious*, meaning they perform all expected launch procedures correctly but may attempt to gain unauthorized access to cryptographic materials loaded onto the spacecraft prior to deployment. We show that our solution addresses this issue.

B. Related work

In her PhD thesis, Fiona Weber addresses SDLS limitations with symmetric cryptography [6]. As her main topic is post-quantum cryptography, she proposes two hybrid *Key Encapsulation Mechanisms* (KEM), called Double-KEM and Triple-KEM. Relying on hybrid cryptography, she shows how to build a post-quantum resistant way to exchange keys. In a similar fashion to our protocols, it uses a static keypair for authenticity and an ephemeral keypair for forward secrecy.

Smailes et al. [7] introduce KEYSPACE, a framework for evaluating Public Key Infrastructure (PKI) mechanisms in interplanetary and satellite networks. They identify the core goals and requirements for secure key management in high-latency, intermittently connected environments and provide standardized experiments to compare PKI systems. Using KEYSPACE, they show that existing terrestrial PKI protocols such as CRLs and OCSP can be adapted to satellite settings when certificate authorities are properly distributed, and they propose novel extensions to improve performance and revocation coverage in deep-space networks.

C. Paper organization

Section II reviews SDLS link security and outlines the technical constraints that shape our design. Section III justifies the choice of primitives. Section IV details the 2DH and 3DH protocols, respectively. Section V discusses a concrete implementation of the protocols. Section VI discusses key-distribution and proposes an evaluation of risks posed by evil launchers and shows how our protocols mitigate the threat. Section VII discusses the quantum threats and the state of the art to mitigate it. Section VIII summarizes our contributions and identifies directions for future research.

II. Context and preliminaries

In SDLS, the concept of a *Secure Association* (SA) is introduced as such: a unidirectional logical channel identified by an SA number, an anti-replay counter, and a long-term symmetric key. Sequence counters prevent duplicate frames, but the standard is silent on how this key is generated or refreshed. In practice, keys are preloaded before launch or uploaded through existing links.

We define technical constraints as the practical limits—imposed by hardware, bandwidth, and operating conditions—that any cryptographic design must respect before security objectives can even be considered.

Storage and secure storage are limited on constrained devices used in space. We must ensure our protocol does not require too much space and that storage usage grows linearly with the number of parties. We must also consider the code size and complexity of the primitives and implementation we choose.

As communication with satellites is noisy and lossy, we need to minimize communications and especially interactive computations that depend on round-trip communication.

Embedded devices often lack good random sources, but some protocols need random ephemeral values. We have to consider the sensitivity to randomness quality of the primitives we choose.

III. Design rationale

This section explains how security objectives and spacecraft constraints drive our choice of cryptographic primitives and protocol structure. It is important to note that the protocols proposed are not quantum-safe, i.e., not resistant against quantum computers. This issue is further discussed in Section VII.

A. Cryptographic Requirements

We consider the following cryptographic properties for the design of our protocol. *Confidentiality* ensures no external party can gain unauthorized access to data. *Integrity* verifies that the exchanges were not modified by an active attacker. *Authenticity* allows to verify the identity of the parties involved in the exchange. *Forward secrecy* ensures that if a long-term secret is compromised, previous sessions stay secure.

B. Protocol Survey

Table I summarizes the asymmetric key-agreement protocols we evaluated against these goals. RSA-based protocols, such as RSA-KEM [8] and RSA-OAEP [9] mainly suffer from significant computational overhead due to their large keys sizes and slow performance. ECIES [10], despite its efficiency in terms of computation and smaller key sizes, inherently lacks forward secrecy. Moreover, ECIES does not implicitly authenticate the sender, necessitating additional signatures and resulting in further overhead. Ephemeral Diffie-Hellman ensures forward secrecy and can provide implicit authenticity through long-term keys. However, its requirement for bidirectional communication and randomness on both sides

TABLE I
CANDIDATE PROTOCOLS FOR OTAR. [1] STATIC-EPHEMERAL. [2]SATELLITE NEEDS TO SIGN.

Algorithm	**RSA-KEM**	**RSA-OAEP**	**ECIES**	**DH**	**S-E[1] DH**	**2DH**	**3DH**
Authenticity	Signature	Signature	Signature	Signature	Implicit[2]	Implicit	Implicit
Forward secrecy	No	No	No	Yes	Ground only	Ground only	Yes
Performance	Slow	Slow	Fast	Fast	Fast	Fast	Fast
Key size	3072 bit	3072 bit	256 bit	256 bit	256 bit	256 bit	256 bit
Communication size	6144 bit	6144 bit	768 bit	768 bit	768 bit	256 bit	256 bit
Randomness	Ground only	Ground only	Ground only	Both sides	Ground only	Ground only	Both sides
Communications	G → S	G → S	G → S	G → S → G	G → S	G → S	G → S → G

makes it challenging to implement efficiently within resource-constrained environments. Static-ephemeral DH addresses the issue of randomness by requiring it only on the sender side. Nevertheless, this comes at the cost of reduced forward secrecy, as the satellite's static key remains unchanged. Additionally, authenticity mechanisms must be explicitly added, increasing complexity. Ultimately, we selected 2DH and 3DH protocols due to their favorable balance. The 2DH protocol minimizes satellite-side computation and randomness requirements, delivering partial forward secrecy, while 3DH provides perfect forward secrecy at the expense of slightly increased computational overhead and the necessity for a reliable entropy source onboard. Both approaches offer implicit authentication without requiring separate signatures, thus making them optimal for secure and efficient OTAR in satellite communication.

IV. DESIGN PROPOSAL

In this section, we show how two DH-based key exchange protocols can be integrated into SDLS. We define a Double Diffie-Hellman and a Triple Diffie-Hellman protocol. We describe how both can be integrated into the overall protocol, discussing the advantages and drawbacks of each design. We also introduce two extended procedures necessary to generate new keys under these protocols.

A. Notation

To describe the two protocols we designed, we will use the following cryptographic primitives, without considering an exact underlying algorithm to implement it. The primitives are the following:
$\mathsf{DH}(y, X)$ — Diffie–Hellman Key Exchange that combines a private key with a peer's public key to yield a shared secret.
$\mathsf{KDF}(seed, ctx, \ell)$ — Key derivation function that expands a *seed* that contains good entropy but is not uniformly distributed into ℓ-bit key material, domain-separated by the context string *ctx*.
$\mathsf{Gen}() = (x, X)$ — Generates a key pair: a private key and its associated public key (*not* the `KeyGen` extended procedure defined later in IV-D).

B. Setup

Each party must establish a long-term *identity key pair* before any rekeying can take place. The procedure comprises two steps:

1) *Key-pair generation.* Each party invokes the key generation algorithm $\mathsf{Gen}()$ to obtain its own keypair. This long-term keypair is the identity of each party.
2) *Secure public-key distribution.* Each public key must be conveyed to every other party over a channel that is *at least as trustworthy* as the identity we wish to bind. This provisioning step is outside the scope of the present work; in practice it is performed during manufacturing, launch-site integration, or via a dedicated secure uplink distinct from the operational TM/TC link.

After provisioning, the spacecraft stores its own private key and the ground station's public key, while the ground station stores its private key and the public key of each spacecraft it controls. All subsequent key-agreement messages rely on this authenticated identity layer.

C. Double Diffie-Hellman protocol

Figure 1 illustrates the message flow. After the static bootstrap (Section IV-E), the ground station generates an ephemeral key pair and invokes the `KeyGen` EP, embedding the ephemeral public key and a list of IDs that will be the symmetric keys to derive (*key_ids*). The spacecraft acknowledges and each party locally computes two DH exchanges before using the KDF function to compute a shared secret.

1) Security Analysis: The protocol achieves implicit *authentication* because the long-term key of each party is involved in the key derivation. As the satellite does not participate in the exchange with an ephemeral value, *forward secrecy* is only *partial*. *Replay attacks* are defeated by the SDLS sequence counter established in the static bootstrap; an adversary re-emitting an old `KeyGen` frame presents an out-of-window counter and is discarded.

2) Performance and Footprint: The spacecraft performs two DH exchanges and one KDF call. Communication overhead is a single frame carrying the ground's ephemeral public key (32 B) plus key identifiers. Hence 2DH is well suited for small satellites with limited entropy, at the cost of reduced forward secrecy.

D. Triple Diffie-Hellman protocol

The Triple Diffie–Hellman (3DH) scheme extends 2DH by adding a spacecraft-generated ephemeral key. In our design, the ground station first emits a `KeyGen` EP embedding its public ephemeral key together with the list of *key_ids* to generate. It then sends a separate lightweight `Get` EP, to which

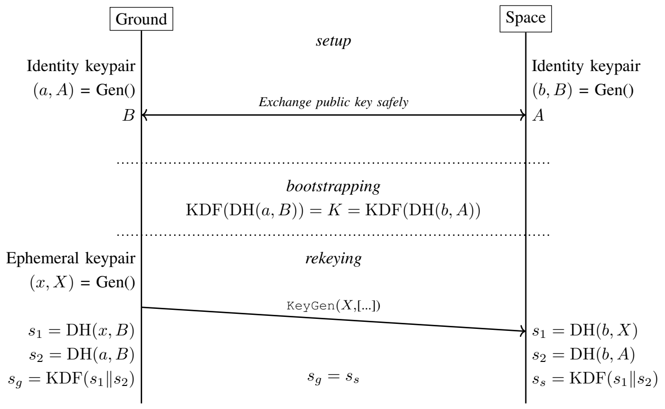
Ground
Space
setup
Identity keypair
(a, A) = Gen()
Identity keypair
(b, B) = Gen()
Exchange public key safely
B
A
bootstrapping
$\mathrm{KDF}(\mathrm{DH}(a, B)) = K = \mathrm{KDF}(\mathrm{DH}(b, A))$
Ephemeral keypair
(x, X) = Gen()
rekeying
KeyGen(X,[...])
$s_1 = \mathrm{DH}(x, B)$
$s_2 = \mathrm{DH}(a, B)$
$s_g = \mathrm{KDF}(s_1 \| s_2)$
$s_g = s_s$
$s_1 = \mathrm{DH}(b, X)$
$s_2 = \mathrm{DH}(b, A)$
$s_s = \mathrm{KDF}(s_1 \| s_2)$

Fig. 1. Sequence diagram of the EPs exchanged for 2DH

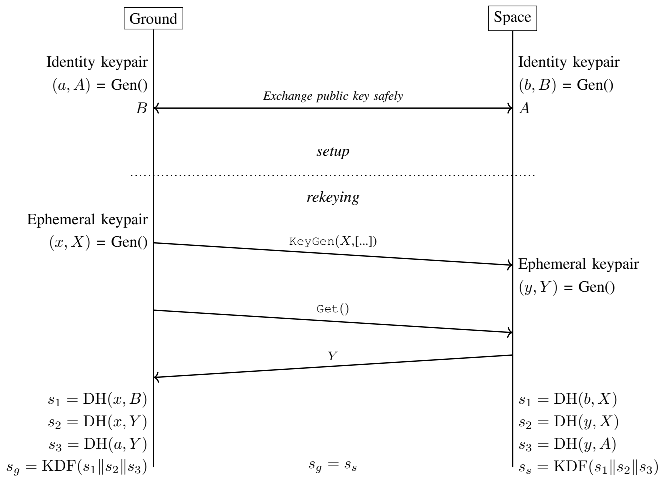
Ground
Space
Identity keypair
(a, A) = Gen()
Identity keypair
(b, B) = Gen()
Exchange public key safely
B
A
setup
rekeying
Ephemeral keypair
(x, X) = Gen()
KeyGen(X,[...])
Ephemeral keypair
(y, Y) = Gen()
Get()
Y
$s_1 = \mathrm{DH}(x, B)$
$s_2 = \mathrm{DH}(x, Y)$
$s_3 = \mathrm{DH}(a, Y)$
$s_g = \mathrm{KDF}(s_1 \| s_2 \| s_3)$
$s_g = s_s$
$s_1 = \mathrm{DH}(b, X)$
$s_2 = \mathrm{DH}(y, X)$
$s_3 = \mathrm{DH}(y, A)$
$s_s = \mathrm{KDF}(s_1 \| s_2 \| s_3)$

Fig. 2. Sequence diagram of the EPs exchanged for 3DH

the spacecraft replies with its own ephemeral public key (Figure 2). This separation is intentional: the `Get` message does not carry new key material but acts as a poll/confirmation step, allowing the ground station to verify that the spacecraft has received the key-generation request and is ready to respond. Keeping it distinct from `KeyGen` improves reliability over lossy or high-latency links, since the ground can retransmit or re-poll without regenerating keys, and the spacecraft only starts a 3DH exchange once it knows the ground is listening. After exchanging these messages, both sides compute the three Diffie–Hellman values and feed them to the KDF to derive the shared secret.

1) Security Analysis: This protocol provides *full forward secrecy* as both parties participate in the exchange with an ephemeral value. This also mitigates *replay attacks*. As for 2DH, *authentication* is implicit, as both parties' long-term identity key is involved in the key derivation.

2) Performance and Footprint: The spacecraft now performs three DH exchanges and one KDF call. The extra `Get` frame adds $32\,\mathrm{B}$ of link traffic. These costs are modest for platforms with sufficient entropy, while offering significantly stronger secrecy guarantees than 2DH.

E. Static Bootstrap and Replay Protection

A naive use of 2DH is vulnerable to replay attacks: an adversary can record the entire sequence—key exchange followed by an operational command—and later re-inject it to trigger the same action again. In 3DH, this risk is avoided because the satellite contributes an ephemeral secret, so a replayed trace produces a different shared secret. With 2DH, however, an explicit countermeasure is required.

We therefore precede every 2DH key exchange with a *static bootstrap* that installs a Security Association (SA) authenticated by a long-term key:

1) Both parties use their long-term static public keys to derive a shared secret.
2) This bootstrap secret is fed into a KDF to derive the key material used by a unidirectional SA.
3) The SA maintains a monotonically increasing sequence counter, appended to every protected frame.

Subsequent frames—such as an EP `KeyGen`—are sent under this SA. The receiver checks both the integrity tag and the counter: messages with a stale value are silently discarded, and any bit-level modification is detected by the integrity check.

F. KDF context

To ensure domain separation when deriving keys, we use the "context" field of the KDF and feed it with the public keys involved in the derivation and the key IDs to generate. This way, we ensure we do not generate the same key for two different IDs, which could lead to serious threats like IV reuse. This context has to be strictly identical on both parties. We defined exactly the format of this context for 2DH and 3DH, as poorly designed context could be used to perform attacks.

G. Entropy

As stated, a good entropy source is required on the satellite for 3DH. Such a source of randomness has always been a challenge to obtain in constrained embedded devices. NIST provides recommendations in [11].

Should entropy available in the system be so low that the satellite's ephemeral key becomes predictable, using 3DH in this state would reduce to the security of 2DH, where only the ground station adds entropy to the key exchange.

H. Summary

The protocol 3DH is *strictly more secure* than 2DH. By adding a spacecraft-generated ephemeral key, 3DH introduces a third DH term that does not depend on any long-term secret; consequently, compromise of *either* party's identity key in the future does *not* reveal past session keys, achieving perfect forward secrecy. It also ensures protection against replay attacks.

This security gain, however, comes at the cost of additional resources: (i) the spacecraft must draw fresh randomness for its own ephemeral scalar, a non-trivial requirement when on-board TRNG quality is limited, (ii) an additional DH computation slightly raises execution time and (iii) an additional round-trip communication is required to deliver the spacecraft's ephemeral public key, adding $32\,\mathrm{B}$ plus SDLS frame headers.

If reliable randomness is available on the space-side, 3DH is the recommended option. Otherwise, 2DH provides a security level that can be sufficient for some missions.

V. Concrete implementation

The exact implementation of the cryptographic primitives has an important impact on performance and communication sizes.

As performance is an important factor for resource-constrained satellites, X25519 [12] is a straightforward choice for the DH exchange, as it is designed with performance in mind. In case we need a FIPS-compliant algorithm, P-256 [13] is a good alternative, but is slightly slower and more code heavy. As a key derivation function, HKDF [14] is a FIPS-compliant standard. Coupled with SHA256, it is lightweight and efficient.

We implemented a proof of concept of the two protocols using the programming language Rust to demonstrate its simplicity and efficiency.

VI. Key distribution and evil launchers

In practice, static keys are often generated outside the satellite and preloaded before launch, and it is not uncommon to use third-party launchers to place a satellite into orbit. With a secret key preloaded in the satellite, an *honest-but-curious* launcher could read and exploit this private key.

Using 2DH, if the attacker captures a handshake between the ground station and the satellite, the attacker would be able to derive the shared secret, allowing to read all the communications but also communicate with the satellite through the secure channels without being detected. The attacker has to capture the message emitted by the ground to the satellite and cannot initiate a communication with the satellite.

In the 3DH protocol, as both parties participate with an ephemeral value, the attacker does not gain any advantage by having the satellite's private key. The communications with 3DH are safe as the 3DH protocol allows to come back to a secure state, even if an evil launcher compromises the satellite's private key.

VII. Toward quantum resistance

Our current 2DH and 3DH protocols, as evaluated in this paper, are not post-quantum secure: any scheme based on Diffie–Hellman over classical groups will be broken by a large-scale quantum adversary. We therefore consider how our constructions could evolve toward post-quantum or hybrid variants, even though such an implementation and assessment fall outside the scope of this work.

NIST announced the deprecation of classical asymmetric algorithms by 2030 and entire prohibition of their use by 2035 [15]. Extending our protocols to post-quantum cryptography—or to hybrid schemes that combine a classical algorithm (e.g., X25519) with a post-quantum algorithm (e.g., ML-KEM) so that the construction remains secure even if one of the two is broken—is therefore a clear priority to ensure long-term cryptographic resilience.

To date, NIST has selected five post-quantum algorithms for standardization. Three of them are already standardized: the signature schemes *ML-DSA* (formerly CRYSTALS-Dilithium) [16] and *SLH-DSA* (formerly SPHINCS+) [17], and KEM *ML-KEM* (formerly CRYSTALS-Kyber) [18]. The standards for the remaining two candidates, the signature scheme *Falcon* [19] and the KEM *HQC* [20], are still in preparation. Since Diffie–Hellman key exchange has no direct post-quantum counterpart, our 2DH and 3DH protocols must be adapted by replacing the DH operations with KEM-based constructions.

A prominent approach is the work proposed by Fujioka et al. [21], which shows how to build an *Authenticated Key Exchange* (AKE) from IND-CCA-secure KEMs. Additionally, Del Pino, Lyubashevsky, and Pointcheval [22] presented another method to construct an AKE by combining a KEM algorithm with a signature scheme. This line of work was further explored by [6], where multiple constructions are proposed, mainly dual-KEM exchanges and triple-KEM exchanges, which achieve roughly the same security properties as our 2DH and 3DH protocols, respectively.

Our choice of candidate algorithms is not ad hoc but guided by space-specific constraints. Among the standardized options, we identified *ML-KEM* as the most suitable KEM for replacing Diffie-Hellman in this context. Should explicit signatures be required, *ML-DSA* stands out as the preferred choice, providing lattice-based post-quantum signatures that align well with the security level of *ML-KEM*. We also considered *Falcon*, but its reliance on floating-point computations is a significant implementation challenge on many embedded systems, especially on space-grade processors.

VIII. Conclusion and further work

This paper has adapted and specified the integration of two DH-based rekeying protocols for SDLS: 2DH and 3DH. In 2DH, only the ground station contributes an ephemeral key, keeping onboard computation and randomness requirements to a minimum but providing partial forward secrecy. The 3DH scheme lets both parties generate fresh ephemeral keys, thereby achieving perfect forward secrecy at the expense of an additional round-trip and an extra computation on the spacecraft. It also ensures anti-replay protection.

The 2DH scheme variant begins with a static–static Diffie–Hellman bootstrap that installs a SDLS *Secure Association*, so replay protection is inherited from the existing SA counter mechanism without altering the frame format.

In summary, 2DH and 3DH demonstrate that replay-safe, forward-secret key refresh can be achieved within the tight bandwidth, storage, and power budgets typical of modern small satellites, while preserving full compatibility with existing SDLS link-layer standards. As shown in our discussion on post-quantum cryptography, it is also possible to adapt these constructions to use post-quantum KEMs or hybrid schemes, thereby future-proofing satellite OTAR mechanisms against upcoming cryptanalytic threats. This opens the door for secure, efficient, and long-lived missions even in a post-2035 cryptographic landscape.

Acknowledgment

The authors would like to thank the CYSEC team for their collaboration and support. This work was carried out as part of a project funded by Innosuisse, in partnership between the School of Engineering and Management Vaud and CYSEC. The authors also wish to express their gratitude to Yann Thoma and his team for their valuable discussions and insightful questions, which helped the work progress significantly.

References

[1] Consultative Committee for Space Data Systems, “Space Data Link Security Protocol, Recommended Standard,” National Aeronautics and Space Administration, Tech. Rep. CCSDS 355.0-B-2, July 2022.

[2] ——, “Cryptographic Algorithms, Recommended Standard,” National Aeronautics and Space Administration, Tech. Rep. CCSDS 352.0-B-2, August 2019.

[3] T. Perrin, “The noise protocol framework,” *noiseprotocol, Protocol Revision*, vol. 34, 2018.

[4] C. Kudla and K. G. Paterson, “Modular security proofs for key agreement protocols,” in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 3788. Springer, 2005, pp. 549–565.

[5] Consultative Committee for Space Data Systems, “Space Data Link Security Protocol – Extended Procedures,” National Aeronautics and Space Administration, Tech. Rep. CCSDS 355.1-B-1, February 2020.

[6] F. J. Weber, “Cryptographic protocols in a post-quantum world,” Ph.D. dissertation, Eindhoven University of Technology, 2025.

[7] J. Smailes, S. Köhler, S. Birnbach, M. Strohmeier, and I. Martinovic, “Keyspace: Public key infrastructure considerations in interplanetary networks,” *arXiv preprint arXiv:2408.10963*, 2024.

[8] R. Housley and S. Turner, “Use of the RSA-KEM Algorithm in the Cryptographic Message Syntax (CMS),” *RFC*, vol. 9690, pp. 1–30, 2025.

[9] M. Bellare and P. Rogaway, “Optimal asymmetric encryption,” in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 950. Springer, 1994, pp. 92–111.

[10] “IEEE standard specifications for public-key cryptography - amendment 1: Additional techniques,” pp. 1–167, 2004.

[11] M. Sonmez, E. Barker, J. Kelsey, K. McKay, M. Baish, and M. Boyle, “Recommendation for the entropy sources used for random bit generation,” Tech. Rep., 2018-01-10 2018.

[12] A. Langley, M. Hamburg, and S. Turner, “Elliptic curves for security,” *RFC*, vol. 7748, pp. 1–22, 2016.

[13] D. A. McGrew, K. M. Igoe, and M. Salter, “Fundamental elliptic curve cryptography algorithms,” *RFC*, vol. 6090, pp. 1–34, 2011.

[14] H. Krawczyk and P. Eronen, “Hmac-based extract-and-expand key derivation function (HKDF),” *RFC*, vol. 5869, pp. 1–14, 2010.

[15] D. Moody, R. Perlner, A. Regenscheid, A. Robinson, and D. Cooper, “Transition to post-quantum cryptography standards,” National Institute of Standards and Technology, Tech. Rep., 2024.

[16] N. I. of Standards and Technology, “Module-lattice-based digital signature standard,” U.S. Department of Commerce, Washington, D.C., Federal Information Processing Standards Publication NIST FIPS 204, 2024. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.204

[17] ——, “Stateless hash-based digital signature standard,” U.S. Department of Commerce, Washington, D.C., Federal Information Processing Standards Publication NIST FIPS 205, 2024. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.205

[18] ——, “Module-lattice-based key-encapsulation mechanism standard,” U.S. Department of Commerce, Washington, D.C., Federal Information Processing Standards Publication NIST FIPS 203, 2024. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.203

[19] Falcon website, https://falcon-sign.info/.

[20] HQC website, https://pqc-hqc.org/.

[21] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama, “Strongly secure authenticated key exchange from factoring, codes, and lattices,” *Designs, Codes and Cryptography*, vol. 76, pp. 469–504, 2015.

[22] R. Del Pino, V. Lyubashevsky, and D. Pointcheval, “The whole is less than the sum of its parts: Constructing more efficient lattice-based akes,” in *International Conference on Security and Cryptography for Networks*. Springer, 2016, pp. 273–291.