Master's thesis proposal: Post-quantum Key Exchange Over COSE

Cuong Nguyen
cuong.c.nguyen@aalto.fi

Advisor: Sampo Sovio
sampo.sovio@huawei.com

February 13, 2025

Contents

1 Introduction

Ephemeral Diffie-Hellman Over COSE (EDHOC) is authenticated key exchange protocol which is designed for low-power devices in the IoT. This kind of communicating environment requires highly restricted bandwidth as well as limited power consumption, leading to the priorities of small message size and high-efficiency resource consumption. In addition, given the significantly increasing number of IoT devices, security is an important concern.

Given the foreseeable threat of Cryptographically Relevant Quantum Computer (CRQC), any cryptographic schemes based on the hardness of integer factoring, e.g. RSA, or discrete logarithm, e.g. Diffie-Hellman, are breakable, leading to "Harvest Now, Decrypt Later" attack strategy [26, 25]. Not being out of this threat, EDHOC relies on the security of Diffie-Hellman model, which is not quantum-resistant. Though EDHOC was designed with post-quantum cryptography (PQC) considerations, there have not been any corresponding specifications regarding this migration, which is stated in the Section 9.4 of RFC 9528 [31]:

> " EDHOC supports all signature algorithms defined by COSE, including PQC signature algorithms such as HSS-LMS. EDHOC is currently only specified for use with key exchange algorithms of type ECDH curves, but any Key Encapsulation Method (KEM), including PQC KEMs, can be used in method 0. While the key exchange in method 0 is specified with the terms of the Diffie-Hellman protocol, the key exchange adheres to a KEM interface: G_X is then the public key of the Initiator, G_Y is the encapsulation, and G_XY is the shared secret. Use of PQC KEMs to replace static DH authentication would likely require a specification updating EDHOC with new methods. "

Furthermore, among totally 4 authentication methods (Table 1), only the first method, which requires rounds of interactivity, is seemingly able to be migrated to PQC through the Key Encapsulation Mechanism (KEM) adaptation. KEM replacement to DH is a straight-forward PQC-migration strategy that has been well-studied in widely-deployed secure protocols and frameworks, such as TLS [23, 30], WireGuard [21], Noise [4], and Split-Key STS [1]. However, when authentication methods based on static DH keys are utilized in EDHOC, this KEM-based approach is doomed to fail; this issue is even raised by John Mattsson, one of the authors of EDHOC [2, 27]. In those static-DH-key-based settings, a non-interactive key exchange (NIKE) protocol is needed, which is somehow reflected in this dicussion among researchers.

Scheme (variant)	Security	Assumption	Non-interactive	Post-quantum	Size (bytes)		Cycles	
					ct	pk	Gen	Encaps + Decaps or SdK
CRYSTALS-Kyber (`Kyber-768` [71])	IND-CCA2	M-LWE	✗	✓	1088	1184	200 302	539 108 (251 384 + 287 724)
Classic McEliece (`mceliece348864` [1])	IND-CCA2	Binary Goppa Codes	✗	✓	96	261 120	46 715 060	143 178 (31 000 + 112 178)
ECDH (`X25519` [13])	HKR-CKS* [5]	CDH	✓	✗	—	32	28 187	87 942
CTIDH (`CTIDH-1024` [10])	HKR-CKS* [6]	CSIDH	✓	✓	—	128	469 520 000	511 190 000
This work (Passive-SWOOSH)	HKR-CKS	M-LWE	✓	✓	—	221184	146 920 890	10 612 666

Figure 1: Comparison of select post-quantum KEMs and NIKEs[18].

Regarding this, CTIDH [5] and SWOOSH [18], which are isogeny-based and lattice-based respectively, appear to be only two post-quantum NIKE currently. As presented in Figure 1, the key size of SWOOSH is prohibitively large, which is unacceptable in constrained environments. On the other hand, CTIDH shows an impressively small key, though its computational time of the shared key is around 50 times slower than that of Passive-SWOOSH. Given the polarized advantages and disadvantages of CTIDH and SWOOSH along with the priority of minimal message size stated in [31], it seems that CTIDH is a more suitable choice for communicating settings where EDHOC is deployed due to its compact key size. Furthermore, a faster and deterministic version of CTIDH, namely dCTIDH [8], recently sheds a light to the potentiality of ECDH replacement in certain applications.

This project aims for migrating EDHOC to PQC in all authentication methods; in other words, designing, implementing, and testing post-quantum variants of EDHOC with all authentication methods. While KEM-based migration is straight-forward for the first authentication method, a PQC NIKE comes as a drop-in replacement of ECDH in the remaining methods. Additionally, we plan to implement EDHOC protocol under the support of Trusted Execution Environment (TEE) which ensures that secret key materials and sensitive cryptographic services are stored and executed in a secure manner.

The proposal is structured as follows. Section 2 introduces premilinaries and technical background. Particular research questions and research methods are presented in Section 3 and Section 4, respectively. Section 5 describes how to conduct experimental settings.

Method Type Value	Initiator Authentication Key	Responder Authentication Key
0	Signature Key	Signature Key
1	Signature Key	Static DH Key
2	Static DH Key	Signature Key
3	Static DH Key	Static DH Key

Table 1: Authentication Keys for EDHOC Method Types [31].

2 Background

2.1 EDHOC

EDHOC is a lightweight authenticated key exchange protocol for low-power environments such as Cellular IoT and Low-Power Radio Wide Area Network (LoRaWAN). It was especially designed to be integrated seamlessly to security framework OSCORE, in which EDHOC provides authentication and key agreement for OSCORE-protected communications over CoAP. By leveraging CBOR for object encoding and COSE for object-oriented cryptographic operations, which are the same primitives deployed under OSCORE, EDHOC introduces minimal extra cost. Compared to other widely-deployed authenticated key agreement protocols, such as TLS Handshake, or the compact version DTLS, and IKEv2, EDHOC outperforms in terms of computational efficiency and message size while providing the equivalent security level.

2.1.1 Overview of EDHOC

EDHOC was built upon a variant of SIGn-and-MAc (SIGMA) family of theoretical protocols, SIGMA-I, which is also the building block of (D) TLS Handshake and IKEv2. Specifically, EDHOC is a MAC-then-Sign Diffie-Hellman based key exchange protocol. Two involved parties, an initiator I and a responder R, are able to mutually authenticate by using their long-term public keys: public signature key or static DH key, resulting in the 4 methods presented in Table 1.

EDHOC includes three mandatory messages, namely *message_1, message_2*, and *message_3*, an optional message for providing addtional key confirmation, *message_4*, and an error message if applicable [31]. If there is no error occured, the shared session key is established for application data protection after the *message_3*. In general, EDHOC protocol could be described as the

following steps:

- *message_1*: I sends setup information along with its identifier and public component of key share
- *message_2*: R computes the shared session key based on I's public component and its own private component. Then, R authenticates its credential and responds with its own identifier, public credentials, all encrypted using the shared key, along with its public component of key share.
- *message_3*: I computes the shared session key based on R's public component and its own private component. Then, I decrypts the encrypted part of *message_2* and authenticates its own credentials.
- *message_4*: Optionally, R explicitly confirms the shared session key.

, which is illustrated in Figure 2. After the shared secret key is exchanged, supsequent data packets are encrypted under OSCORE framework.

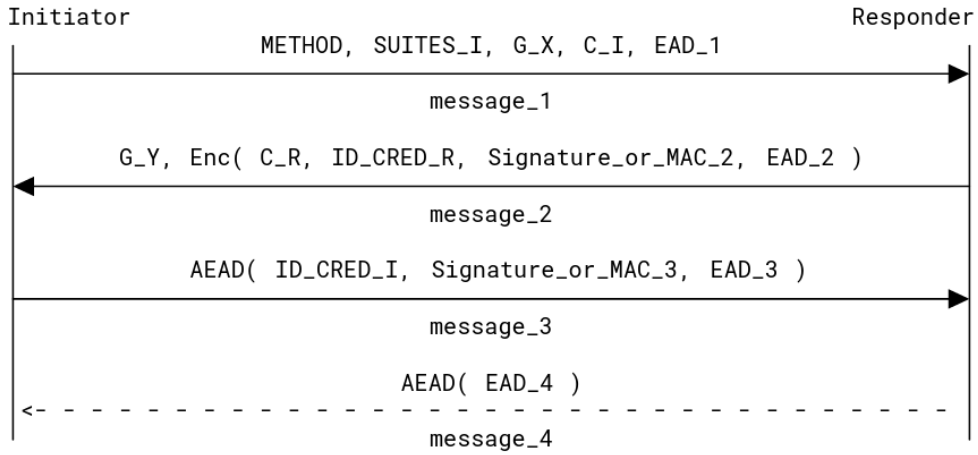

Figure 2: EDHOC message flow (message_4 is optional) [31].

2.1.2 Security properties

According to [27], the security goals of the protocol could be summarized as follows:

- **Confidentiality**: the shared secret key at the end is only known to the two authenticated-communicating parties. Moreover, even if an active attacker compromises either one of peer's private key, they would not be able to compromise past session keys (Forward Secrecy).
- **Mutual authentication**: Each peer should re-authenticate the other at the end of the session, and both must agree on a new sessionidentifier, roles, and credentials. Critically, in the given session, the compromomise of the long-term secretof one party should not allow an attacker to impersonate the other party (Key Compromise Impersonate resistance). Furthermore, identity misbinding protection should be provided by the protocol.
- **Identity protection**: The protocol must protect one peer's identity from active attacks and the other peer's identity from passive attacks. Identity is a unique identifier which could be a cryptographic certificate, public keys, or MAC addresses or any other unique identifiers exchanged during the protocol.
- **Cryptographic strength**: The target security level of the protocol's key is at least 128 bits. This requirement applies to the authentication, shared session keys and negotiation for all cryptographic parameters.
- **Protection of external security data**: External Authorization Data (EAD) should have the same level of protection as the protocol message.
- **Downgrade protection**: To deal with long-term deployment, cryptographic agility must be provided to support different cryptographic primitives, resulting in a mutually agreed cipher suit after a round of negotiation. Downgrade protection is essential to prevent attackers from forcing the use of weaker cryptography. This includes preventing cipher suite downgrade attacks (forcing less secure cipher suites) and key material downgrade attacks (manipulating key derivation to use weaker keys).

In order to guarantee that the protocol satisfies the aforementioned high-level security goals, researchers have formally verified, using both symbolic and computational models, certain drafts (12,15,17,23) of EDHOC with all authentication methods [22, 12, 19, 16]. Given those security analyses, a security analysis of the proposed post-quantum variant could be built upon them.

2.2 Key Encapsulation Mechanism (KEM)

A KEM is a public encryption scheme that establishes a shared secret key, which can be used for symmetric encryption, between two communicating parties. According to [13, 14], a KEM consists of three probabilistic polynomial-time (PTT) algorithms:

- **KEM.KeyGen**: a probabilistic key generation algorithm that takes security parameters 1^λ as input and produces a public/secret key pair (PK, SK)
- **Encap**: a probabilistic key encapsulation, also known as polynomial-time encryption, algorithm that takes a public key PK as input and produces an encapsulated key pair (K, C), sometimes C is also known as a ciphertext or an encapsulation of the key K.
- **Decap**: a probabilistic key decapsulation, also known as polynomial-time decryption, algorithm that takes a ciphertext C and a secret key SK as input, and recover the key K.

As a result, the secure KEM key K, the output of **Encap** and **Decap**, is shared between two communicating parties.

2.2.1 Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM)

At the time of writing, among several proposed KEMs, only CRYSTALS-KYBER [6], whose name is later changed to ML-KEM, is standardized by NIST [32]. The security of ML-KEM is based on the hard problem called Module Learning with Errors (MLWE) [24], which is a more general version of the Learning With Errors (LWE) [28].

ML-KEM guarantees IND-CCA2 security [3, 6].

2.3 Non-Interative Key Exchange (NIKE)

Following [17, 10], NIKE is defined as a tuple of three PPT algorithms: **Setup**, **NIKE.KeyGen**, and **SharedKey** accompanying an identity space IDS and a shared key space SHK.

- **Setup**: a common setup algorithm that takes security parameter 1^λ as input and outputs a set of system parameters *params*.
- **NIKE.KeyGen**: a key generation algorithm that takes *params* and an identity $ID \in IDS$ as input and produces a public/secret key pair

(PK, SK).

- **SharedKey**: a shared key establishment algorithm that takes as input an identity $ID_1 \in IDS$ and a public key pk_1 together with another identity $ID_2 \in IDS$ and its corresponding secret key sk_2 and returns either a shared key $k \in SHK$, or a failure symbol $\perp$. It is assumed that **SharedKey** always outputs $\perp$ if $ID_1 = ID_2$

Regarding the correctness, given any pair of identities ID_1, ID_2 along with their corresponding public/secret key pairs (pk_1, sk_1) and (pk_2, sk_2), the algorithm **SharedKey** holds the following constraint:

$$\textbf{SharedKey}(ID_1, pk_1, ID_2, sk_2) = \textbf{SharedKey}(ID_2, pk_2, ID_1, sk_1)$$

2.3.1 dCTIDH

Commutative Supersingular Isogeny Diffie-Hellman (CSIDH) [11] is an isogeny-based NIKE that enables parties to establish a shared key using endomorphisms of supersingular elliptic curves. Its security is based on the hardness of computing endomorphism rings of supersingular elliptic curves, a problem believed to be hard even for quantum computers. However, it introduces a high computational cost, which is especially not desired in resource-constrained environments, due to large prime fields and group actions. In addittion, as a consequence of its probabilistic nature, CSIDH is difficult to implement in constant-time [11]. To make CSIDH constant-time, many approaches rely on dummy operations, operations that do not affect the result depending on values in secret key, which leads to fault injection attacks [9, 7]. Depending on the specific use case, this type of physical attack is considered whether problematic or not [8].

Compressed Torsion-point Isogeny Diffie-Hellman (CTIDH) [5] is an optimized version of CSIDH. CTIDH has solved the issues of constant-time by introducing dummy operations, therefore it is a dummy-based implementation. By leveraging a different way of specifying the key space and some algorithmic modifications, CTIDH, to date, is the most efficient constant-time implementation of CSIDH in terms of both key size and computaitonal cost, even better than time-variant CSIDH.

dCTIDH is a deterministic version of CTIDH. Deterministics implementations do not require a high-quality source of entropy during computation. The results from [8] shows that dCTIDH not only outperforms the deterministic CSIDH (dCSIDH), but also non-deterministic CTIDH. At the time of writing, dCTIDH is the most efficient constant-time, deterministic implementation.

2.4 Trusted Execution Environment (TEE)

TEE is a secure, tamper-resistant processing environment providing code executation, memory and storage capabilities on a separated kernel from a rich execution environment (REE) [15]. It guarantees the authentication of executed code, the integrity of runtime states, and the confidentiality of code, data, and runtime states stored on a persistent memory [29]. By isolating the REE, also known as the "normal" execution environment, from highly secure environments so that sensitive operations and data are protected and sandboxed inside TEE, TEE enhances the security level of sensitive services even if they interact with a non-secure world.

In the cryptography context, TEE is a sandboxed environment that securely stores and handles functions associated to secret key materials, preventing the leakage of private keys.

3 Research questions

- Is it possible, if yes then how to integrate quantum-resistant cryptographic schemes into EDHOC without any major changes in its design and specification?
- What are trade-offs in terms of message size, cpu cycle, and latency given that EDHOC was designed specifically for highly restricted settings? According to EDHOC RFC [31], minimal message size is the top priority. Moreover, in terms of time latency, the time distribution needs to be considered during evaluation besides the average value which could be drastically affected by the rarely-appeared outliers.
- Does the PQC-adopted design still meets security goals of the original protocol as well as the PQC security level standardized by NIST?
- Given TEE-protected EDHOC implementation [20], is there any significant overheads introduced by the PQC-variant of EDHOC? If yes, does the security enhancement provided by TEE outweight those extra costs?

4 Research methods

- Modifies EDHOC design to adapt KEM for the method 0, and quantum-resistant NIKE, SWOOSH or CTIDH, for the remaining methods.
- Formally verifies the PQC-variant designs, using automated tool such as ProVerif.
- Implements PQC-variant EDHOC in the manner of TEE isolation. Leveraging the existing work of µOSCORE-µEDHOC [20], which is a TEE-supported implementations of the original (non-quantum-resistant) EDHOC.
- Measures the performance of PQC-variant EDHOC in terms of message size, cpu cycle, and latency, as well as demystify the time distribution. Additonally, compare its performance to that of DTLS, a widely used protocol in IoT environments, and the original (non quantum-safe) EDHOC.

5 Experimental setups

Since the firmware μOSCORE-μEDHOC would be used as the baseline for the post-quantum variant implementation, I would follow their setups for reproducability and comparability.

5.1 EDHOC between two microcontrollers

Requirements:

- Zephyr OS is needed for constrained devices.
- Two microcontrollers (e.g. nRF52832, nRF52840).
- Two Raspberry Pi act as IPv6 over BLE routers for nrf52832. If nRF52840 is utilized, there is no need for these external routing devices as nRF52840 has built-in BLE.

References

[1] Gizem Akman et al. "Split keys for station-to-station (STS) protocols". In: *Journal of Surveillance, Security and Safety* 4.3 (2023). ISSN: 2694-1015. DOI: 10.20517/jsss.2023.16. URL: https://www.oaepublish.com/articles/jsss.2023.16.

[2] John Mattsson et al. *Non-Interactive Key Exchange, CSIDH, and Swoosh.* 2025. URL: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/y_nQ3xMDRy8/m/uI1nU5iuAAAJ (visited on 02/10/2025).

[3] José Bacelar Almeida et al. *Formally verifying Kyber Episode V: Machine-checked IND-CCA security and correctness of ML-KEM in EasyCrypt.* Cryptology ePrint Archive, Paper 2024/843. 2024. URL: https://eprint.iacr.org/2024/843.

[4] Yawning Angel et al. "Post Quantum Noise". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* CCS '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 97–109. ISBN: 9781450394505. DOI: 10.1145/3548606.3560577. URL: https://doi.org/10.1145/3548606.3560577.

[5] Gustavo Banegas et al. "CTIDH: faster constant-time CSIDH". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.4 (Aug. 2021). Artifact available at https://artifacts.iacr.org/tches/2021/a20, pp. 351–387. DOI: 10.46586/tches.v2021.i4.351-387.

[6] Joppe Bos et al. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P).* 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032.

[7] Fabio Campos, Juliane Krämer, and Marcel Müller. "Safe-Error Attacks on SIKE and CSIDH". In: *Security, Privacy, and Applied Cryptography Engineering: 11th International Conference, SPACE 2021, Kolkata, India, December 10–13, 2021, Proceedings.* Kolkata, India: Springer-Verlag, 2021, pp. 104–125. ISBN: 978-3-030-95084-2. DOI: 10.1007/978-3-030-95085-9_6. URL: https://doi.org/10.1007/978-3-030-95085-9_6.

[8] Fabio Campos et al. *dCTIDH: Fast & Deterministic CTIDH.* Cryptology ePrint Archive, Paper 2025/107. 2025. URL: https://eprint.iacr.org/2025/107.

[9] Fabio Campos et al. "Trouble at the CSIDH: Protecting CSIDH with Dummy-Operations Against Fault Injection Attacks". In: *2020 Workshop on Fault Detection and Tolerance in Cryptography (FDTC).* 2020, pp. 57–65. DOI: 10.1109/FDTC51366.2020.00015.

[10] David Cash, Eike Kiltz, and Victor Shoup. "The Twin Diffie-Hellman Problem and Applications". In: *Advances in Cryptology – EUROCRYPT 2008.* Ed. by Nigel Smart. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 127–145. ISBN: 978-3-540-78967-3.

[11] Wouter Castryck et al. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: *Advances in Cryptology – ASIACRYPT 2018*. Vol. 11274. Lecture Notes in Computer Science. Springer, 2018, pp. 395–427. DOI: 10.1007/978-3-030-03332-3_15.

[12] Baptiste Cottier and David Pointcheval. "Security Analysis of Improved EDHOC Protocol". In: *Foundations and Practice of Security*. Ed. by Guy-Vincent Jourdan et al. Cham: Springer Nature Switzerland, 2023, pp. 3–18. ISBN: 978-3-031-30122-3.

[13] Ronald Cramer and Victor Shoup. "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack". In: *SIAM Journal on Computing* 33.1 (2003), pp. 167–226. DOI: 10.1137/S0097539702403773. eprint: https://doi.org/10.1137/S0097539702403773. URL: https://doi.org/10.1137/S0097539702403773.

[14] Alexander W. Dent. "A Designer's Guide to KEMs". In: *Cryptography and Coding*. Ed. by Kenneth G. Paterson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 133–151. ISBN: 978-3-540-40974-8.

[15] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. "Trusted execution environments on mobile devices". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: Association for Computing Machinery, 2013, pp. 1497–1498. ISBN: 9781450324779. DOI: 10.1145/2508859.2516758. URL: https://doi.org/10.1145/2508859.2516758.

[16] Loïc Ferreira. "Computational Security Analysis of the Full EDHOC Protocol". In: *Topics in Cryptology – CT-RSA 2024*. Ed. by Elisabeth Oswald. Cham: Springer Nature Switzerland, 2024, pp. 25–48. ISBN: 978-3-031-58868-6.

[17] Eduarda S. V. Freire et al. "Non-Interactive Key Exchange". In: *Public-Key Cryptography – PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 254–271. ISBN: 978-3-642-36362-7.

[18] Phillip Gajland et al. "SWOOSH: Efficient Lattice-Based Non-Interactive Key Exchange". In: *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 487–504. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/gajland.

[19] Felix Günther and Marc Ilunga Tshibumbu Mukendi. "Careful with MAc-then-SIGn: A Computational Analysis of the EDHOC Lightweight Authenticated Key Exchange Protocol". In: *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. 2023, pp. 773–796. DOI: 10.1109/EuroSP57164.2023.00051.

[20] Stefan Hristozov et al. "The Cost of OSCORE and EDHOC for Constrained Devices". In: *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*. CODASPY '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 245–250. ISBN: 9781450381437. DOI: 10.1145/3422337.3447834. URL: https://doi.org/10.1145/3422337.3447834.

[21] Andreas Hülsing et al. "Post-quantum WireGuard". In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 304–321. DOI: 10.1109/SP40001.2021.00030.

[22] Charlie Jacomme et al. "A comprehensive, formal and automated analysis of the EDHOC protocol". In: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 5881–5898. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/jacomme.

[23] Hugo Krawczyk and Hoeteck Wee. "The OPTLS Protocol and TLS 1.3". In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2016, pp. 81–96. DOI: 10.1109/EuroSP.2016.18.

[24] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *Designs, Codes and Cryptography* 75.3 (June 2015), pp. 565–599. ISSN: 1573-7586. DOI: 10.1007/s10623-014-9938-4. URL: https://doi.org/10.1007/s10623-014-9938-4.

[25] Lukasz Olejnik, Robert Riemann, and Thomas Zerdick. *EDPS TechDispatch on Quantum Computing and Cryptography*. Tech. rep. Technology and Privacy understandingnit of the European Data Protection Supervisor (EDPS), 2020. URL: https://www.edps.europa.eu/sites/default/files/publication/06-08-2020_techdispatch_quantum_computing_en.pdf.

[26] David Ott, Christopher Peikert, and other workshop participants. *Identifying Research Challenges in Post Quantum Cryptography Migration and Cryptographic Agility*. 2019. arXiv: 1909.07353 [cs.CY]. URL: https://arxiv.org/abs/1909.07353.

[27] Elsa López Pérez et al. "EDHOC Is a New Security Handshake Standard: An Overview of Security Analysis". In: *Computer* 57.9 (2024), pp. 101–110. DOI: 10.1109/MC.2024.3415908.

[28] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93. ISBN: 1581139608. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603.

[29] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. "Trusted Execution Environment: What It is, and What It is Not". In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. 2015, pp. 57–64. DOI: 10.1109/Trustcom.2015.357.

[30] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1461–1480. ISBN: 9781450370899. DOI: 10.1145/3372297.3423350. URL: https://kemtls.org/publication/kemtls/.

[31] Göran Selander, John Preuß Mattsson, and Francesca Palombini. *Ephemeral Diffie-Hellman Over COSE (EDHOC)*. RFC 9528. Mar. 2024. DOI: 10.17487/RFC9528. URL: https://www.rfc-editor.org/info/rfc9528.

[32] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Tech. rep. Washington, D.C.: U.S. Department of Commerce, 2024. DOI: 10.6028/NIST.FIPS.203. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf.