Quantum secure communication using hybrid post-quantum cryptography and quantum key distribution

2024 24th International Conference on Transparent Optical Networks (ICTON) | 979-8-3503-7732-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICTON62926.2024.10648124

Quantum secure communication using hybrid post-quantum cryptography and quantum key distribution

Nick Aquina
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, The Netherlands
n.aquina@tue.nl

Simon Rommel
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, The Netherlands
s.rommel@tue.nl

Idelfonso Tafur Monroy
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, The Netherlands
i.tafur.monroy@tue.nl

***Abstract*—Once a cryptographically relevant quantum computer is built, confidential information that is communicated today can be decrypted and exposed. Key exchange methods currently used can then be broken and the key used to protect the data can be recovered. Depending on the duration for which information needs to remain confidential, quantum-safe solutions have to be used now to protect data transmitted today. In this paper, we review recent advances in integrating the combination of two different quantum-safe solutions, Quantum Key Distribution (QKD) and Post-Quantum Cryptography (PQC), in existing protocols. We also combine classical cryptography, PQC and QKD to securely exchange a key that remains secure even if only one of the three key exchange methods is secure. This is one of the first steps to update existing communication methods such that they remain secure even if only one of PQC, QKD or classical cryptography is secure.**

***Index Terms*—QKD, PQC, quantum-secure communications, post-quantum cryptography, KEM combiners**

I. Introduction

Quantum Computers with enough qubits will be able to use Shor's algorithm to break widely deployed public-key cryptography schemes such as RSA and Elliptic-Curve Diffie-Hellman (ECDH) [1]. This makes it possible to retroactively recover keys that were exchanged in the past. These keys can then be used to decrypt confidential data. This attack is also known as the "harvest now, decrypt later" attack [2]. To protect confidential data from being exposed, new key exchange methods which cannot be broken by quantum computers, such as post-quantum cryptography (PQC) and Quantum Key Distribution (QKD), will have to be deployed. New key exchange methods have seen less scrutiny from experts and might risk being broken by algorithms that can run on a classical computer [3]. Therefore, until confidence has been gained in the new cryptography, existing key exchange methods should still be used and quantum-secure methods have to be used alongside these classical key exchange methods. Classical and new methods must be combined in a way that guarantees security if only one of the methods is secure [4]. Even if the adversary can maliciously choose two of the three keys that will be combined, the final key should be secure.

The authors would like to acknowledge the funding received under the FIQCS project (NWA.1436.20.005) and the EU funding received under the ALLEGRO project (grant No. 101092766).

A KEM (Key Encapsulation Mechanism) is a key exchange method that exchanges a key using a public key. When using multiple KEMs at the same time to enhance security, the keys from the KEMs have to be combined. This can be done using a KEM combiner.

In this paper we show how KEM combiners can be used with Diffie-Hellman (DH) and QKD, extending the use of KEM combiners beyond PQC. We also demonstrate the use of a KEM combiner with DH, PQC and QKD in a protocol such that security is guaranteed even if only one of the key exchange methods is secure.

In section II we describe the security notions used for KEMs. In section III we discuss related work on key combiners, which key combiners have been used with QKD and PQC and we discuss decapsulation oracles. In section IV we propose how KEM combiners can be used with QKD and we describe a specific KEM combiner that has been proven to be secure. We also explain a protocol which uses the previously described KEM combiner to combine keys that were exchanged using classical cryptography, PQC and QKD. Finally, in section V we conclude the paper.

II. Key Encapsulation Mechanisms and Combiners

A KEM consists of three algorithms [5]:

1) $(pk, sk) \leftarrow_{\$} KeyGen()$, which generates a public key pk and private/secret key sk.
2) $(k, c) \leftarrow_{\$} Encapsulate(pk)$, which generates a key k and a ciphertext c.
3) $k \leftarrow Decapsulate(sk, c)$, which calculates the key using the secret key and the ciphertext.

Usually, $KeyGen$ is run by Alice, who then sends the public key pk to Bob. Bob encapsulates the public key pk using $Encapsulate$ to generate a key k and a ciphertext c. He then sends the ciphertext c to Alice. Alice then decapsulates the ciphertext c with the secret key sk using $Decapsulate$ to

get the key k. A KEM combiner is a function that securely combines the output of multiple KEMs into one key.

A KEM combiner can be indistinguishable under chosen plaintext attack (IND-CPA), indistinguishable under chosen ciphertext attack (IND-CCA) and split-key pseudorandom. These properties are formalized as games [5]. In all three games the attacker cannot distinguish the final key from a random string of bits, i.e., the attacker has no information about the final key. The difference between the games is the capabilities that the attacker is given. In the IND-CPA game, the attacker has access to all public information and all but one key that is fed into the KEM combiner. In case one of the KEMs is secure, an IND-CPA secure KEM combiner would be sufficient to protect against the "harvest now, decrypt later" attack. In the IND-CCA game, the attacker has all the capabilities that he has in the in the IND-CPA game, but also has access to a decapsulation oracle of the KEM combiner. This decapsulation oracle will give the attacker the final key when the attacker inputs all the public information necessary to calculate this key. The decapsulation oracle itself has access to all the secrets required to calculate the final key. The attacker, however, is not allowed to use the decapsulation oracle on the public information that has been used to calculate the final key. The attacker can, however, try to send an altered version of the public information to the decapsulation oracle to trick the decapsulation oracle such that it provides the attacker information about the final key. In the split-key pseudorandom game, the attacker can also choose the keys that will be used in the KEM combiner except for the key of one KEM.

Security of a KEM combiner for a certain game can be proven in different models. Security proofs given in models that use fewer assumptions than other models are considered more secure. When proving the security of a KEM in the random oracle model, the security relies on the assumption that a random oracle is used in the KEM combiner [6]. If it turns out that the hash function used in the KEM combiner does not behave like a random oracle, the security proof does not apply anymore and the KEM might turn out to be insecure. When the security of a KEM is proven in the standard model, the security proof of the KEM combiner does not require a random oracle.

We consider a KEM combiner secure if the attacker cannot learn any information about the final key even if the attacker has full control over the input produced by all but one KEM and the attacker has access to a decapsulation oracle that uses this KEM combiner. This means that a KEM combiner should be split-key pseudorandom.

III. Related work

A. KEM combiners

Giacon *et al.* [5] investigated the security of different KEM combiners and discovered that KEM combiners require the ciphertext produced by the encapsulation of the KEM to be used in the key derivation to retain IND-CCA security. They also propose a split-key pseudorandom KEM combiner, the PRF-then-XOR split-key combiner.

Petcher and Campagna [7] introduced CtKDF, a KEM combiner proven secure in the random oracle model. CtKDF is a variant of CatKDF which besides the key, also combines the public information produced by the KEM. When using HKDF as the Key Derivation Function (KDF), CtKDF requires hashing almost three times fewer bytes than the PRF-then-XOR split-key combiner when used in the protocol presented in section IV. However, it requires to exchange a random label to salt the KDF and IND-CCA security has not been proven in the standard model.

ETSI has standardized two key combiners [8]: concatenation and cascading. However, neither concatenation nor cascading has been proven to be secure when the attacker can control one of the keys used. The standard states that IND-CPA is sufficient and stronger security notions are unnecessary, but does not argue why these would be unnecessary. These combiners are not secure according to our definition and we will thus not use these combiners.

B. Key combiners used with QKD

Rubio García *et al.* [9] modified a TLS 1.2 implementation such that it uses PQC instead of a classical key exchange and it also retrieves and uses a QKD key. They analyzed the performance of different key-exchange methods in the modified implementation and evaluated the results. They show that QKD can be used in an existing protocol. The work removes the classical key exchange, which risks losing security in case the post-quantum systems turn out to be insecure. The security of post-quantum systems are less understood than the classical key exchange methods, which is why the French Cybersecurity Agency recommends combining classical algorithms with post-quantum algorithms [4]. The authors propose two methods to combine the PQC key and the QKD key. The first proposed method is to concatenate both keys and then derive a new key using a pseudorandom function (PRF). The second proposed approach is to XOR the keys and then derive a key using a PRF. Both key combiners are secure when the attacker learns one of the two keys, IND-CPA, but both methods were not proven to retain security when the attacker has access to a decapsulation oracle [5], [7].

Rubio García *et al.* [10] modified a TLS 1.3 implementation such that it combines classical cryptography, PQC and QKD. They tested the solution in a practical scenario, analyzed the performance of different key-exchange methods and studied the implications of the modified implementation. The keys were combined by using a PRF, in this case HKDF, on the concatenation of the keys. Aviram *et al.* [11] note that this construction was never proven secure in case the attacker can choose part of the input to HKDF. In case the classical and post-quantum digital signature algorithm used are broken, the attacker can choose part of the input to the key derivation. This construction thus requires a split-key pseudorandom key combiner to retain security in case only QKD is secure.

Ricci *et al.* modified a dual-PRF [12] used by Aviram *et al.* as a key combiner [11]. They proof that their key combiner

is a IND-CCA secure dual-PRF and use it to combine Diffie-Hellman, Kyber and QKD.

C. Decapsulation oracles

The KEM combiners proposed by [5] and [7] were designed to combine keys from KEMs. Both Diffie-Hellman and QKD are however not a KEM. The construction from both authors require the ciphertext produced by the KEM as input to the key derivation. Neither DH nor QKD produce a ciphertext. The reason that the ciphertext was included in the key derivation is to achieve IND-CCA security. In the IND-CCA game, the attacker is given access to a decapsulation oracle that will decapsulate any ciphertext that is not the ciphertext that is used to determine the final key. To determine what should be used instead of the ciphertext for both QKD and Diffie-Hellman we can look at the what the attacker can input into the decryption oracles of Diffie-Hellman and QKD.

Brendel *et al.* [13] describe the decryption oracle for Diffie-Hellman: $PRF((pk_{adv})^{sk}, x_{adv})$. In the IND-CCA game, the adversary has access to the result produced by this function and can choose pk_{adv} and x_{adv}. The adversary is, however, not allowed to use the public key that was used to derive the key. Since both sides in the key exchange might expose a decryption oracle to the adversary, the final key should depend on the public keys of both parties. Diffie-Hellman can also be transformed into a KEM as done by Bindel *et al.* [14]. In this construction, the ciphertext of the KEM is the DH public key, which confirms that the DH public key should be used in the key derivation.

Chen *et al.* [15] integrate QKD into WireGuard and in their QKD based KEM they describe a QKD decapsulation oracle that has no access to the QKD key.

IV. Hybrid Authenticated Key Exchange

We propose a decapsulation oracle for QKD, differing from the one proposed in [15]: the Key Management Systems (KMS) used by the applications to retrieve the QKD key. In contrast to the decapsulation oracle proposed by [15], our decapsulation oracle could be used by the adversary to retrieve the QDK key. As standardized by ETSI [16], a QKD KMS has two methods to retrieve a key (ignoring authentication): $getKey()$, which returns a key and a Universally Unique IDentifier (UUID), and $getKeyWithKeyID(uuid)$ which returns the key for the specified UUID. In a KMS which does not properly handle authentication, $getKeyWithKeyID(uuid)$ could be abused by an adversary as a decapsulation oracle. In case Alice requests a key using $getKey()$ and already encrypts and sends data using this key without having checked that Bob instead of Eve requested $getKeyWithKeyID(uuid)$, the data that Alice encrypted using this key can be decrypted by Eve. The input of the attacker to the decapsulation oracle is the UUID. Since the input of the decapsulation oracle has to be used to achieve IND-CCA [5], the UUID should be used in the key derivation. We can also model QKD key retrieval from a KMS as a KEM to reach the same conclusion. We construct this KEM Q, such that public and secret keys play no role. Q.Encapsulate() produces a UUID and a key, Q.Decapsulate(uuid) produces a key. Q.Encapsulate() and Q.Decapsulate(uuid) are respectively $getKey()$ and $getKeyWithKeyID(uuid)$. The ciphertext of the KEM in this construction is the UUID, which should thus be used in the key derivation.

Now that we have determined the input to the decryption oracles of DH, PQC and QKD, we know all the information that should influence the final key such that the adversary cannot trick the decryption oracle into giving the adversary information about the key.

Giacon *et al.* [5] proposed several KEM combiners. Only one proposed combiner is split-key pseudorandom and proven secure in the standard model. This is the PRF-then-XOR split-key combiner. Modified for combining DH, PQC and QKD this combiner would become:

$$v \leftarrow dhpk_A \parallel dhpk_B \parallel c \parallel uuid \tag{1}$$

$$K \leftarrow PRF(k_{dh}, v) \oplus PRF(k_{pq}, v) \oplus PRF(k_{qkd}, v) \tag{2}$$

When using HMAC as PRF and Kyber-1024 as KEM, this combiner requires to hash 5196 bytes, while the key combiner from [12] requires to hash 7035 bytes.

To achieve secure communication we use the PRF-then-XOR split-key combiner to combine classical cryptography (X25519 and Ed25519), PQC (Kyber-1024 and Dilithium5) and QKD using the protocol shown in Fig. 1. The ECDH (X25519) public key, Kyber public key and the QKD UUID are communicated from Alice to Bob. The message by Alice is signed by Alice using EdDSA (Ed25519) and Dilithium5 and also authenticated using Poly1305 with keys exchanged using QKD. We assume that Alice and Bob have previously communicated their EdDSA and Dilithium public keys or their public keys are signed by a certificate authority that they trust. In the latter case, Alice and Bob also have to communicate the certificate alongside the signature. We also assume that Alice and Bob have previously communicated their QKD SAE ID to each other. The KMS can then make sure that the QKD key they share is only known to Alice and Bob and a part of the key material from QKD can then be used for authentication. After Bob receives the message from Alice and after verifying the signatures, Bob calculates the ECDH key, Kyber key and retrieves the QKD key. At this point, Bob can combine the three keys into the final key using the PRF-then-XOR split-key combiner. He then sends his ECDH public key, the Kyber encapsulation and a hash of Alice's message to Alice. He signs this message using EdDSA and Dilithium and also authenticates the message using a part of the key material retrieved from QKD. When Alice receives Bob's response and after verifying Bob's signatures and MAC, she can calculate the ECDH key and the Kyber key. She then combines the three keys into the final key using the same method as Bob, using the PRF-then-XOR split-key combiner. This key can then be used to derive more keys which can be used for encryption and authentication.

The hash of the message that Alice sends is signed by Bob to prevent an attacker from being able to replay messages from

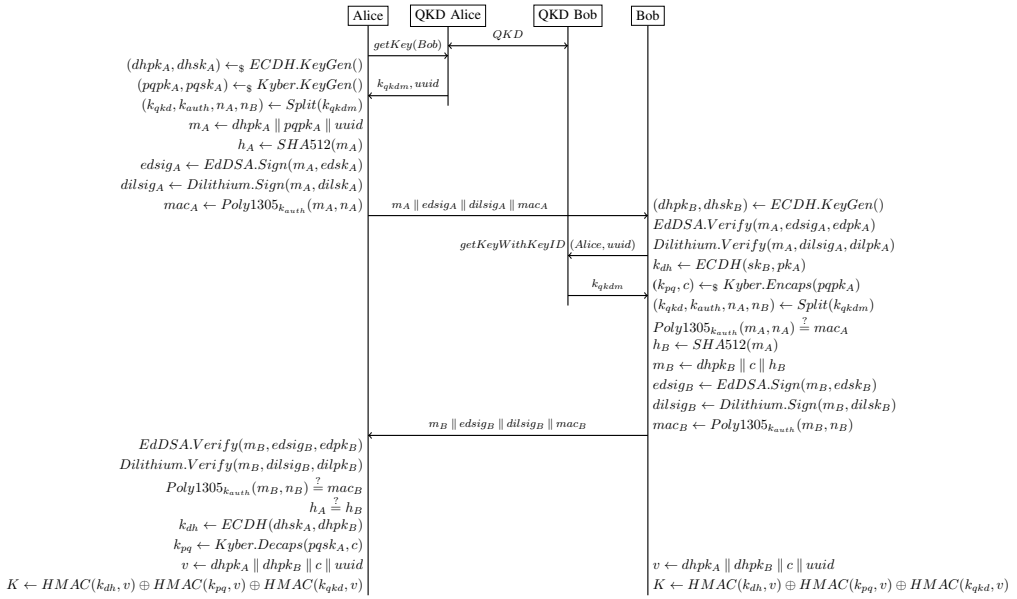

Fig. 1. Authenticated Key Exchange protocol used to agree on a key using ECDH, PQC and QKD

Bob. Messages from Alice can still be replayed. A replayed message from Alice will cause Bob to waste resources on key derivation and key retrieval from QKD. Assuming that the KMS of Bob deletes the key after the first time Bob retrieves this key, the key retrieval from QKD for a replayed message will fail and the attacker will not learn anything about the keys used. This replay attack could be prevented by using a timestamp in the message from Alice to Bob which Bob remembers, but this is not implemented in this protocol and left as future work.

Since a split-key pseudorandom KEM combiner was used in the protocol, the Poly1305 authentication with QKD keys could be removed to lower the required key rate from QKD from 640 bits to 256 bits per key exchange. In case both EdDSA, Dilithium and ECDH are broken, the attacker might be able to decide the key used for ECDH, but the attacker cannot decide the QKD key used. The attacker would not learn anything about the final key as the QKD key is random and a split-key pseudorandom KEM combiner is used.

Although not visible in Fig. 1, in our experimental setup, the QKD key was transmitted using the BB84 protocol using the Toshiba MU Multiplexed to an intermediate trusted node. This key was then relayed from this node to the KMS of Bob by encrypting and authenticating it using a QKD key that was transmitted from the intermediate node to the QKD device of Bob. This key was transmitted using the COW protocol using the ID Quantique Cerberis XGR.

The final key derived after running the protocol will be unknown to an attacker even if two out of three from classical cryptography (ECDH and EdDSA), PQC (Kyber and Dilithium) and QKD are broken, which is not the case for all protocols presented in section III-B.

V. Conclusions

Although QKD has been added to several protocols to retain security in case classical and post-quantum cryptography can be broken, the constructions used do not always retain security in case only QKD is secure. We have shown how KEM combiners can be used with QKD and propose the usage of the PRF-then-XOR split-key combiner with QKD. We use this combiner in a protocol that combines ECDH, the post-quantum KEM Kyber and QKD and we have implemented this protocol in a QKD testbed.

References

[1] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[2] General Intelligence and Security Service, *Prepare for the threat of quantum computers*, Feb. 17, 2022. [Online]. Available: https://english.aivd.nl/binaries/aivd-en/documenten/publications/2022/01/18/prepare-for-the-threat-of-quantumcomputers/Prepare+for+the+threat+of+quantumcomputers.pdf.

[3] W. Castryck and T. Decru, *An efficient key recovery attack on SIDH*, Publication info: Published by the IACR in EUROCRYPT 2023. [Online]. Available: https://eprint.iacr.org/2022/975.

[4] French National Agency for the Security of Information Systems. "ANSSI views on the post-quantum cryptography transition — ANSSI," Agence nationale de la sécurité des systèmes d'information. (Jan. 4, 2022), [Online]. Available: https://cyber.gouv.fr/en/publications/anssi-views-post-quantum-cryptography-transition (visited on 03/20/2024).

[5] F. Giacon, F. Heuer, and B. Poettering, *KEM combiners*, 2018. [Online]. Available: https://eprint.iacr.org/2018/024.

[6] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security*, ser. CCS '93, New York, NY, USA: Association for Computing Machinery, Dec. 1, 1993, pp. 62–73, ISBN: 978-0-89791-629-5. DOI: 10.1145/168588.168596.

[7] A. Petcher and M. Campagna, *Security of hybrid key establishment using concatenation*, Publication info: Preprint., 2023. [Online]. Available: https://eprint.iacr.org/2023/972.

[8] ETSI, *Quantum-safe hybrid key exchanges*, version 1.1.1, Dec. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103700_103799/103744/01.01.01_60/ts_103744v010101p.pdf.

[9] C. Rubio García, S. Rommel, S. Takarabt, *et al.*, "Quantum-resistant transport layer security," *Computer Communications*, vol. 213, pp. 345–358, Jan. 2024, ISSN: 01403664. DOI: 10.1016/j.comcom.2023.11.010.

[10] C. Rubio García, A. C. Aguilera, J. J. V. Olmos, I. T. Monroy, and S. Rommel, "Quantum-resistant TLS 1.3: A hybrid solution combining classical, quantum and post-quantum cryptography," in *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, ISSN: 2378-4873, Nov. 2023, pp. 246–251. DOI: 10.1109/CAMAD59638.2023.10478407.

[11] N. Aviram, B. Dowling, I. Komargodski, K. G. Paterson, E. Ronen, and E. Yogev, *Practical (post-quantum) key combiners from one-wayness and applications to TLS*, Publication info: Preprint., 2022. [Online]. Available: https://eprint.iacr.org/2022/065.

[12] S. Ricci, P. Dobias, L. Malina, J. Hajny, and P. Jedlicka, "Hybrid keys in practice: Combining classical, quantum and post-quantum cryptography," *IEEE Access*, vol. 12, pp. 23 206–23 219, 2024, Conference Name: IEEE Access, ISSN: 2169-3536. [Online]. Available: https://ieeexplore.ieee.org/document/10430098.

[13] J. Brendel, M. Fischlin, F. Günther, and C. Janson, *PRF-ODH: Relations, instantiations, and impossibility results*, Publication info: A major revision of an IACR publication in CRYPTO 2017, 2017. [Online]. Available: https://eprint.iacr.org/2017/517.

[14] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, *Hybrid key encapsulation mechanisms and authenticated key exchange*, Publication info: Published elsewhere. Major revision. 10th International Workshop on Post-Quantum Cryptography (PQCrypto 2019), 2018. [Online]. Available: https://eprint.iacr.org/2018/903.

[15] L. Chen, X. Kaping, J. Li, Z. Li, and N. Yu, "Security-enhanced WireGuard protocol design using quantum key distribution," presented at the International Conference on Computing, Networking and Communications, 2024. [Online]. Available: http://www.conf-icnc.org/2024/papers/p718-chen.pdf.

[16] *Protocol and data format of REST-based key delivery API*, version 1.1.1, Feb. 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf.