IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 1, 22 pages.

https://doi.org/10.62056/a3qj89n4e

X-Wing

The Hybrid KEM You've Been Looking For

Manuel Barbosa[1,2,3], Deirdre Connolly[6], João Diogo Duarte[1,2], Aaron Kaiser[3], Peter Schwabe[3,4], Karolin Varner[3,5] and Bas Westerbaan[7]

[1] University of Porto, Portugal
[2] INESC TEC, Portugal
[3] Max Planck Institute for Security and Privacy, Germany
[4] Radboud University, The Netherlands
[5] Rosenpass e.V., Germany
[6] SandboxAQ, USA
[7] Cloudflare, The Netherlands

Abstract. X-Wing is a hybrid key-encapsulation mechanism based on X25519 and ML-KEM-768. It is designed to be the sensible choice for most applications. The concrete choice of X25519 and ML-KEM-768 allows X-Wing to achieve improved efficiency compared to using a generic KEM combiner. In this paper, we introduce the X-Wing hybrid KEM construction and provide a proof of security. We show (1) that X-Wing is a classically IND-CCA secure KEM if the strong Diffie-Hellman assumption holds in the X25519 nominal group, and (2) that X-Wing is a post-quantum IND-CCA secure KEM if ML-KEM-768 is itself an IND-CCA secure KEM and SHA3-256 is secure when used as a pseudorandom function. The first result is proved in the ROM, whereas the second one holds in the standard model. Loosely speaking, this means X-Wing is secure if either X25519 or ML-KEM-768 is secure. We stress that these security guarantees and optimizations are only possible due to the concrete choices that were made, and it may not apply in the general case.

Keywords: Hybrid KEM · Post-Quantum Cryptography · Public-Key Cryptography · X-Wing

1 Introduction

To counter the potential threat of store-now/decrypt-later attacks using quantum computers, industry has started to deploy post-quantum key-encapsulation mechanisms (KEMs) for key agreement (cf. [O'B23, WR22]). The post-quantum cryptography that is now being considered for widespread adoption is relatively young compared to the public-key cryptography that has protected applications until now. For example, RSA and (elliptic-curve) Diffie-Hellman might not be secure against quantum-computers, but these schemes have undergone decades of cryptographic analysis. For this reason, many opt to deploy a *hybrid* of traditional and post-quantum schemes: if the post-quantum component turns out to be weak, security falls back to the (non-post-quantum) security of traditional constructions.

Warning. X-Wing depends on the NIST standard ML-KEM, which has not been finalized yet. Thus, X-Wing is not final yet.

E-mail: mbb@fc.up.pt (Manuel Barbosa), durumcrustulum@gmail.com (Deirdre Connolly), joao@diogoduarte.pt (João Diogo Duarte), aaron.kaiser@mpi-sp.org (Aaron Kaiser), peter@cryptojedi.org (Peter Schwabe), karo@cupdev.net (Karolin Varner), bas@westerbaan.name (Bas Westerbaan)

Received: 2024-01-09 Accepted: 2024-03-05

Conversely, when the traditional component becomes weak due to practical quantum attacks, security falls back to the (post-quantum) security of the post-quantum scheme.

As pointed out by Giacon, Heuer, and Poettering [GHP18], under the standard IND-CCA security definition for KEMs[1], creating a generic KEM combiner is more subtle than one might expect. They show that the obvious combiner $\mathsf{KDF}(k_1 \| k_2)$ is not IND-CCA secure in all cases, but that such a generic combiner can be made secure by mixing in the ciphertexts into the KDF input, e.g., as $\mathsf{KDF}(k_1 \| k_2 \| c_1 \| c_2)$. As typical post-quantum KEMs have large ciphertexts, this second combiner comes with a noticeable performance penalty.

Despite this, in specific applications such as TLS 1.3, the simpler combiner is taken to be secure because the ciphertexts are mixed into the *transcript hash*. This has led to the current situation where there are two different hybrid KEMs, both called X25519Kyber768Draft00 [WS23, WW23]: one for HPKE where the ciphertext is used in the key derivation input, and one for use in TLS where security instead relies on the *transcript hash* outside the combiner. This is an undesirable situation.

In this paper, we show that there is a class of KEMs where the best of both worlds can be achieved: For these KEMs, the simple, more efficient combiner yields an IND-CCA secure hybrid KEM despite not using the KEM ciphertext during key derivation. We call these KEMs *ciphertext second preimage resistant (C2PRI)* and show that ML-KEM [NIS23] provides ciphertext second preimage resistance. We use this observation to design an efficient, hybrid KEM suitable for a broad range of applications.

Furthermore, besides having to choose the details of the combiner (e.g., which PRF to use or which design to follow), the KEMs used in the combiner have to be decided upon as well, which involves several crucial choices to start the scheme. Then, the parameters and security levels have to be fixed. There might be further details to consider. For instance, for ECDH, designers need to decide upon how a KEM is to be created from the Diffie–Hellman key exchange like X25519 [Ber06, LHT16].

Although there is value in having a standardized recipe to create bespoke hybrid KEMs using a black-box combiner from existing KEMs [OWK23], for most use cases a single choice is beneficial for increased interoperability, and reduced engineering efforts. With X-Wing we are making concrete choices, and we provide a proof of security for these specific choices. This simplifies our proof and yields a performance level unmet by previous constructions at the same security level.

X-Wing is a concrete KEM and not a generic combiner. Its simplicity and performance ensures it is a good choice in most use cases, including TLS and HPKE. X-Wing targets 128-bit security, and achieves this goal by combining X25519 and ML-KEM-768 using SHA3-256 as the key derivation function. ML-KEM-768 is chosen over ML-KEM-512 to hedge against advances in cryptanalysis and to match X25519Kyber768Draft00 [WS23], which already found real-world usage.

In this paper, we show that X-Wing achieves IND-CCA security based on either the security properties of either X25519 or ML-KEM. In the pre-quantum case, we model SHA3-256 as a random oracle and reduce the IND-CCA security of the scheme to the strong Diffie-Hellman problem in the X25519 nominal group. To achieve this result, we also show that ML-KEM-768 retains a ciphertext second preimage resistance, even if its secret key is known to the attacker. In the post-quantum case, we give a standard model reduction from the IND-CCA security of X-Wing to IND-CCA security of ML-KEM-768 while assuming SHA3-256 to be a PRF. Here we closely follow the proof idea for KEM combiners given by Giacon, Heuer, and Poettering [GHP18], while extending it to KEMs that may allow for a small decryption error probability.

[1] Here, IND-CCA means the standard notion of ciphertext indistinguishability under adaptive chosen-ciphertext attacks for KEMs.

Structure of this paper. To start, in Section 2 we describe the design of X-Wing, and explain the choices made. Before getting into the details of our construction, in Section 3, we give an intuitive explanation for the security of our scheme as well as a sketch of the proof. Notation and definitions are introduced Section 4. The QSF framework is introduced in Section 5; our generic construction – termed Quantum Superiority Fighter – constitutes a generalization of the specific X-Wing construction. In Section 6 we prove the security of QSFs in general, and finally, in Section 7 we show that X-Wing is a Quantum Superiority Fighter and provide benchmarks for an optimized implementation of X-Wing.

2 Design

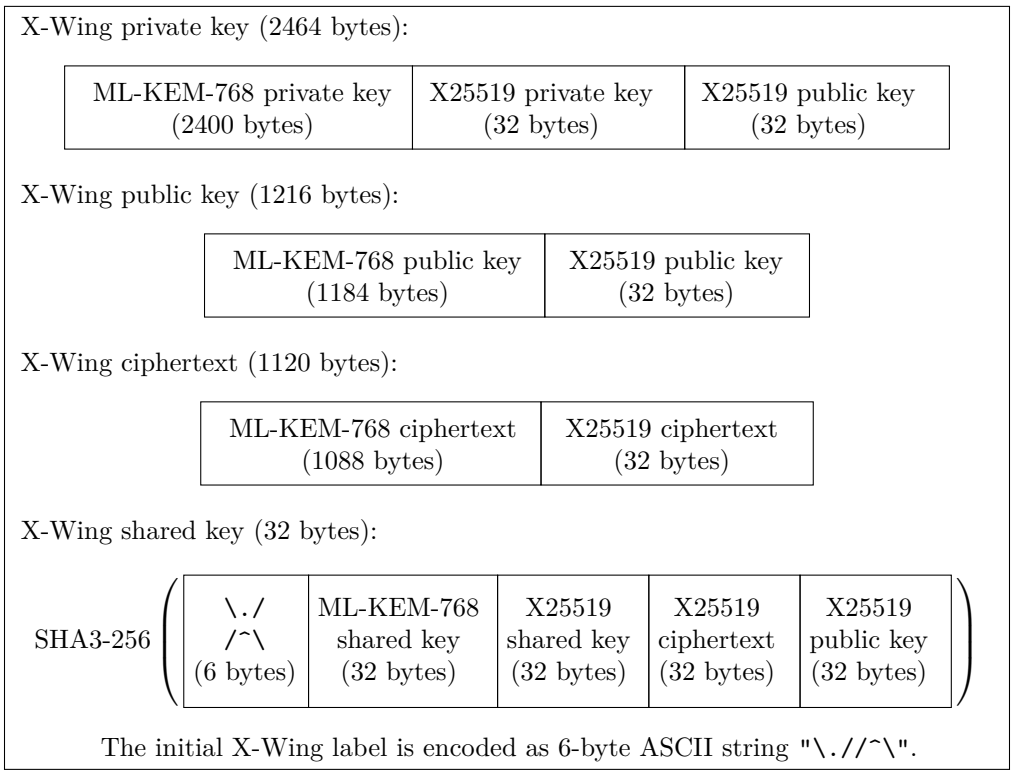


Figure 1: The X-Wing KEM private key, public key, ciphertext, and shared key.

The primary goal of X-Wing is to be usable and the go-to solution in most applications. There are many aspects to being 'usable', which can be in conflict.

First, there are the security guarantees and performance, which we already covered in the introduction; by targeting IND-CCA security at 128 bits with extra margin for ML-KEM, we have a solid security guarantee with comfortable margin, while retaining performance.

Secondly, there is implementation simplicity. We designed X-Wing so that it is straightforward to implement with X25519 and ML-KEM-768 as black boxes. In particular, we opted not to use the DHKEM(X25519) construction from HPKE [BBLW22, ABH+21] that turns X25519 into a KEM. Also, these considerations steered us away from utilizing a standard combiner. Compared to a scheme based on DHKEM and the GHP-combiner [GHP18], we achieve the same security at a lower computational cost and implementation complexity. We stress that this optimization is possible, because of the concrete choices we made, and it may not apply in general.

We chose X25519 as it is currently the de-facto standard traditional key-agreement with excellent performance. ML-KEM is currently NIST's only choice for a post-quantum KEM. More importantly, these choices closely match with the choices made in the Internet-Draft X25519Kyber768Draft00 [WS23], which is used in the current wide-scale early deployment of post-quantum cryptography in TLS by Google and Cloudflare [O'B23, WR22]. A summary of the X-Wing design is depicted in Figure 1.

While we could further improve performance, by opening up ML-KEM and merging the key derivation phase of ML-KEM with that of X-Wing, we decided against this optimization since it would require the implementer to open up the ML-KEM implementation abstraction, which might not always be possible or desirable.

The final key-derivation includes the X25519 public key and ciphertext (i.e., both the DH long-term and ephemeral public keys). The first is added as a measure of security against multi-target attacks, similarly to what is done in the ML-KEM design.[2] Removing the X25519 ciphertext from the final key-derivation step is not possible, as X25519, if seen as a KEM, is not ciphertext second preimage resistant. Removing either or both tokens would not improve performance by much, since in the KDF call, we are already processing only a single SHA3-256 input block.

3 Security Intuition

Second Preimage attacks against KEM Combiners. Hashing public keys, shared keys, and ciphertexts are natural steps in creating a generic KEM combiner, but one might think that the key-derivation stage could be as uncomplicated as concatenating both shared keys before applying a key-derivation function. Let us call this the *pedestrian* combiner. Giacon, Heuer, and Poettering [GHP18] showed that such a combiner would not *robustly* provide IND-CCA security. Instead, the GHP-combiner [GHP18] that is proved secure, additionally mixes both ciphertexts into the key-derivation step.

The paper provides an in-depth explanation of the attack that can be performed if the ciphertexts are omitted. To briefly illustrate the attack, recall that a robust combiner is meant to combine two or more schemes so that security is preserved if all but one schemes are replaced by an arbitrarily bad scheme. Consider a hypothetical bad scheme, which is broken in the following sense: given a challenge ciphertext c_1, it is easy to find another ciphertext $c_1' \neq c_1$ that decapsulates to the same shared key k_1. Then, an attacker could win the IND-CCA game against the KEM combiner as follows: Given (c_1, c_2) as the challenge ciphertext, where c_2 is the ciphertext for a secure KEM, the adversary knows this will decapsulate to $\mathsf{KDF}(k_1, k_2)$ for unknown k_2. However, the attacker can simply call its decapsulation oracle on (c_1', c_2), which is a legitimate query, and obtain the correct shared key.

Second Preimage Resistant KEMs. The fact that the previous attack works for a degenerate insecure KEM does not mean that omitting a corresponding ciphertext from the KDF input always leads to an attack. Indeed in this paper we show that, for X-Wing, this is *not* the case: we can omit the (large) ML-KEM-768 ciphertext from the KDF input, and still prove that X-Wing is an IND-CCA secure KEM. Intuitively, this is because ML-KEM-768 has the following property: even if ML-KEM-768 is broken as a KEM, we show that its internal structure—based on the Fujisaki-Okamoto transform—guarantees that it is impossible to find a second ciphertext as described above. We call this notion *ciphertext second preimage resistance* for KEMs.

[2] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/C0D3W1KoINY/m/99kIvydoAwAJ

Inlining the Diffie-Hellman based KEM. The X-Wing construction is not a generic KEM combiner because we do not treat the DH component as a black-box KEM. We instead take the direct route, reducing the X25519-based security of our construction to the hardness of breaking the Strong Diffie-Hellman [ABR01] (SDH) problem in a nominal group. Intuitively, this means that we prove security based on a computational-Diffie-Hellman-like problem, but no assumption is made about the format of the generated group element. In particular, no assumption is made that the shared group element is indistinguishable from random bytes. Indistinguishability of the final shared secret from a random key is established by modeling the key-derivation function as a random oracle.

Proof-strategy. The proof is split into two cases: one covering the possibility of quantum computers becoming a reality (and thus breaking X25519), and the other one covering the possibility of post-quantum cryptography, specifically ML-KEM-768, being classically broken.

Pre-Quantum Security: This the case case where X25519 is assumed to be secure but ML-KEM-768 may not offer any security. This proof is itself presented in two steps and, in both cases, the attacker is assumed to be classical. We first prove that X-Wing is IND-CCA secure when:

1. The SHA3-256 KDF is modeled as a Random Oracle;
2. X25519 is modeled as a nominal group in which the strong Diffie-Hellman assumption holds; and
3. ML-KEM-768 is modeled as a possibly broken KEM that provides no security beyond ciphertext second preimage resistance.

We then show that ML-KEM-768 satisfies the ciphertext second preimage resistance property, again in the ROM.

Post-Quantum Security: This is the case where ML-KEM-768 is assumed to be post-quantum secure, but X25519 offers no security. This proof is itself presented in two steps and, in both cases, the attacker is assumed to be able to perform quantum computations, but it interacts with X-Wing classically. We prove in the standard model that X-Wing is secure when:

1. The SHA3-256 KDF is modeled as a (post-quantum) PRF;
2. There is no assumption on the X25519 component; and
3. ML-KEM-768 is assumed to provide (post-quantum) IND-CCA security.

We present the above proofs in a modular way, by first justifying the QSF design, and then arguing that SHA3-356, X25519 and ML-KEM-768 are good instantiations.

4 Preliminaries

4.1 Notation and conventions

For an integer n, we denote by $\mathbb{Z}_n$ the residual ring $\mathbb{Z}/n\mathbb{Z}$. $a \leftarrow\!\$\ A$ denotes sampling a uniformly at random from a non-empty finite set A. $\leftarrow$ denotes a deterministic assignment of a variable. $\{0,1\}^n$ is the set of all bitstrings of length n. (x, y) denotes a tuple of two elements x and y. $\mathbf{X}[y]$ denotes access into the table $\mathbf{X}$ at position y. Tables are denoted with bold uppercase variable names or $\sum$. An uninitialized position in a table is denoted with the bottom symbol $\perp$. $\mathbf{X}[\cdot] \leftarrow y$ sets all positions of table $\mathbf{X}$ to y. $o \leftarrow\!\$\ \mathcal{A}(I)$ denotes

running the algorithm $\mathcal{A}$ with input I with uniform random coins and o describing its output. If $\mathcal{A}$ has additionally access to an oracle O, this is denoted as $o \leftarrow\!\$\ \mathcal{A}^{O(\cdot)}(I)$. A security game consists of a main procedure and optionally some oracle procedures. When a game is played, the main procedure is run and adversary $\mathcal{A}$ is given some inputs and access to the oracle procedures. Based on the output of the adversary $\mathcal{A}$ and its oracle calls, the main procedure outputs 1 or 0 depending on whether the adversary $\mathcal{A}$ won the game. If a game aborts at any time, it means that the adversary has no advantage in winning this game. In case of a decision game, this means that the game returns a random bit indicating whether the adversary has won or not. Whenever an adversary algorithm executes "stop with x", it halts returning x to its challenger. We denote an adversary $\mathcal{A}$ playing game G as $G_{\mathcal{A}}$. The probability of the game returning a 1 when played by adversary $\mathcal{A}$ is written as $\Pr[G_{\mathcal{A}} \Rightarrow 1]$. Our security analyses are concrete, which means that we prove that the advantage of an attacker against a construction with fixed parameters is bounded by a real value that is argued to be small. Here, *small* means a summation of statistical terms and advantage terms, the former representing worst-case probabilities below 2^{-128} and the latter representing attacks on lower-level primitives that are assumed to require at least 2^{128} steps to break.

4.2 Key-Encapsulation Mechanisms

4.2.1 Syntax

Definition 1 (Key Encapsulation Mechanism (KEM))**.** A key encapsulation mechanism is a triple of algorithms KEM = $\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ with public keyspace $\mathcal{PK}$, private keyspace $\mathcal{SK}$, ciphertext space $\mathcal{C}$, and shared keyspace $\mathcal{K}$. The triple of algorithms is defined as:

- KEM.$\mathsf{KeyGen}(\,) \$\!\!\mapsto (sk, pk)$ Randomized algorithm that outputs a secret (private) key $sk \in \mathcal{SK}$, and a public key $pk \in \mathcal{PK}$.
- KEM.$\mathsf{Enc}(pk) \$\!\!\mapsto (k, c)$ Randomized algorithm that, given a public key $pk \in \mathcal{PK}$, outputs a shared key $k \in \mathcal{K}$, and a ciphertext $c \in \mathcal{C}$.
- KEM.$\mathsf{Dec}(c, sk) \rightarrow y \in \{k, \perp\}$ Deterministic algorithm that, given a secret key, $sk \in \mathcal{SK}$ and a ciphertext $c \in \mathcal{C}$, returns the shared key $k \in \mathcal{K}$. In case of rejection, this algorithm returns $\perp$.

4.2.2 Correctness

The correctness of a KEM imposes that, except with small probability drawn over the random coin space of KEM.KeyGen and KEM.Enc, we have that KEM.Dec correctly recovers the shared key produced by KEM.Enc. Formally, we say that a KEM is δ-correct if:

$$\Pr\left[k \neq k' \;\middle|\; \begin{smallmatrix} (sk,pk) \leftarrow\!\$ \mathsf{KeyGen}(\,) \\ (k,c) \leftarrow\!\$ \mathsf{Enc}(pk) \\ k' \leftarrow \mathsf{Dec}(c,sk) \end{smallmatrix} \right] \leq \delta.$$

4.2.3 Security

The IND-CCA security game for KEMs is denoted as $\text{IND-CCA}^{b}_{\text{KEM},\mathcal{A}}$ and is shown in Figure 2.

Definition 2 (IND-CCA advantage for KEMs)**.** The advantage of $\mathcal{A}$ in breaking the IND-CCA security of KEM is defined as

$$\text{Adv}^{\text{KEM}}_{\text{IND-CCA},\mathcal{A}} = \left|\Pr[\text{IND-CCA}^{0}_{\text{KEM},\mathcal{A}} \Rightarrow 1] - \Pr[\text{IND-CCA}^{1}_{\text{KEM},\mathcal{A}} \Rightarrow 1]\right|.$$

Game $\text{IND-CCA}^b_{\text{KEM},\mathcal{A}}$
$(sk, pk) \leftarrow\$ \text{KEM.KeyGen}()$
$(k_0, c^*) \leftarrow\$ \text{KEM.Enc}(pk)$
$k_1 \leftarrow\$ \mathcal{K}$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{Dec}(\cdot)}(pk, c^*, k_b)$
return b'

Oracle $\mathsf{Dec}(c)$
if $c = c^*$ **then**
  return $\perp$
$k \leftarrow \text{KEM.Dec}(c, sk)$
return k

Figure 2: IND-CCA security game for KEMs.

4.3 Pseudorandom function

A pseudorandom function (PRF) is a keyed deterministic function that cannot be distinguished from a truly random function.

Definition 3 (Pseudorandom Function (PRF))**.** For a finite keyspace $\mathcal{K}$, input space $\mathcal{X}$, finite output space $\mathcal{Y}$, and an efficiently computable function $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we define the PRF advantage of adversary $\mathcal{A}$ based on the experiments in Figure 3 as

$$\text{Adv}^f_{\text{PRF},\mathcal{A}} = \left|\Pr[\text{PRF}^0_{f,\mathcal{A}} \Rightarrow 1] - \Pr[\text{PRF}^1_{f,\mathcal{A}} \Rightarrow 1]\right|.$$

Game $\text{PRF}^b_{f,\mathcal{A}}$
$\mathbf{T}[\cdot] \leftarrow \perp$
$k \leftarrow\$ \mathcal{K}$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{Eval}(\cdot)}$
return b'

Oracle $\mathsf{Eval}_f(x)$
if $x \in \mathbf{T}$ **then**
  return $\mathbf{T}[x]$
$y_0 \leftarrow f(k, x)$; $y_1 \leftarrow\$ \mathcal{Y}$
$\mathbf{T}[x] \leftarrow y_b$
return y_b

Figure 3: PRF security games.

4.4 Nominal Group

The construction that we introduce in this paper uses an elliptic curve. In our security proofs, we abstract away the elliptic curve and use the notion of a nominal group. In this section, we recall the definition of a nominal group, as of [ABH+21].

Definition 4 (Nominal Group [ABH+21])**.** A nominal group $\mathcal{N} = (\mathcal{G}, g, p, \varepsilon_h, \varepsilon_u, \mathsf{exp})$ consists of an efficiently recognizable finite set of elements $\mathcal{G}$ (also called "group elements"), a base element $g \in \mathcal{G}$, a prime p, a finite set of honest exponents $\varepsilon_h \subset \mathbb{Z}$, a finite set of exponents $\varepsilon_u \subset \mathbb{Z}/p\mathbb{Z}$, and an efficiently computable exponentiation function $\mathsf{exp} : \mathcal{G} \times \mathbb{Z} \to \mathcal{G}$, where we write X^y for $\mathsf{exp}(X, y)$. The exponentiation function is required to have the following properties:

1. $(X^y)^z = X^{yz}$ for all $X \in \mathcal{G}, y, z \in \mathbb{Z}$.
2. The function ϕ defined by $\phi(x) = g^x$ is a bijection from ε_u to $\{g^x | x \in [1, p-1]\}$.

As done in [ABH+21] for a nominal group $\mathcal{N} = (\mathcal{G}, g, p, \varepsilon_h, \varepsilon_u, \mathsf{exp})$, we define D_H to be the uniform distribution of honestly generated exponents ε_h and D_U to be the uniform distribution on ε_u. We also recall the definition for the statistical distance between these distributions: $\Delta_{\mathcal{N}} = \Delta[D_H, D_U] = \frac{1}{2} \sum_{x \in \mathbb{Z}} |\underset{D_U}{Pr}(x) - \underset{D_H}{Pr}(x)|$ [ABH+21].

4.5 Strong Diffie-Hellman Problem

Definition 5 (Strong Diffie-Hellman (SDH) Problem)**.** We define the advantage function of an adversary $\mathcal{A}$ against the Strong Diffie-Hellman problem over the nominal group $\mathcal{N}$ as

$$\mathrm{Adv}^{\mathrm{SDH}}_{\mathcal{N},\mathcal{A}} = \Pr\left[Z = g^{xy} \,\middle|\, x, y \leftarrow \varepsilon_u; Z \leftarrow \mathcal{A}^{DH(\cdot,\cdot)}(g^x, g^y)\right].$$

where DH is a decision oracle that on input (Y, Z) with $Y, Z \in \mathcal{G}$, returns 1 iff $Y^x = Z$ and 0 otherwise.

5 Introducing QSF

In this section, we introduce a framework to build a robust hybrid KEM based on two building blocks, a nominal group and another KEM. We show that the resulting KEM is secure even if one of the building blocks loses its security properties. We call this QSF.

Definition 6 (QSF)**.** Let $\mathcal{N} = (\mathcal{G}, g, p, \varepsilon_h, \varepsilon_u, \mathsf{exp})$ be a nominal group, let KEM be a key encapsulation mechanism and let H be a hash function. We define the QSF KEM as depicted in Figure 4.

Algorithm $\mathsf{KeyGen}()$
$(sk_1, pk_1) \leftarrow\!\$\ \mathrm{KEM.KeyGen}()$
$sk_2 \leftarrow\!\$\ \varepsilon_h$
$pk_2 \leftarrow \mathsf{exp}(g, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
$sk \leftarrow (sk_1, sk_2, pk_2)$
return (sk, pk)

Algorithm $\mathsf{Enc}(pk)$
$(pk_1, pk_2) \leftarrow pk$
$k_1, c_1 \leftarrow\!\$\ \mathrm{KEM.Enc}(pk_1)$
$sk_e \leftarrow\!\$\ \varepsilon_h$
$c_2 \leftarrow \mathsf{exp}(g, sk_e)$
$k_2 \leftarrow \mathsf{exp}(pk_2, sk_e)$
$k \leftarrow H(\mathrm{label} \| k_1 \| k_2 \| c_2 \| pk_2)$
$c \leftarrow (c_1, c_2)$
return (k, c)

Algorithm $\mathsf{Dec}(c, sk)$
$(sk_1, sk_2, pk_2) \leftarrow sk$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \mathrm{KEM.Dec}(c_1, sk_1)$
$k_2 \leftarrow \mathsf{exp}(c_2, sk_2)$
if $k_1 = \perp$ **then**
  return $\perp$
$k \leftarrow H(\mathrm{label} \| k_1 \| k_2 \| c_2 \| pk_2)$
return k

Figure 4: QSF Framework.

5.1 Correctness

The correctness of the QSF framework follows from the correctness of the KEM used to instantiate QSF. Formally, if QSF is instantiated with a δ-correct KEM, then the correctness of QSF is defined as follows:

$$\Pr\left[k \neq k' \;\middle|\; \begin{smallmatrix}(sk,pk)\leftarrow\!\$\mathsf{KeyGen}(\,)\\(k,c)\leftarrow\!\$\mathsf{Enc}(pk)\\k'\leftarrow\mathsf{Dec}(c,sk)\end{smallmatrix}\right] \leq \delta.$$

6 Security of QSF

In this section, we will analyze the security of the QSF framework. We will consider two cases. In the first case, we will analyze the security of QSF relative to the security of the nominal group, while in the second case, we analyze the security relative to the security of the underlying KEM.

6.1 Reduction to SDH and C2PRI in the ROM

As mentioned in the introduction, we introduce ciphertext second preimage resistance for KEMs (C2PRI) and show that this notion is sufficient, together with the SDH security of the nominal group, to make QSF an IND-CCA secure KEM.

Definition 7 (C2PRI)**.** We define the advantage function of an adversary $\mathcal{A}$ against KEM second preimage resistance as:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathrm{KEM},\mathcal{A}} = \Pr\left[\mathsf{Dec}(c, sk) = k^* \wedge c \neq c^* \;\middle|\; \begin{smallmatrix}(sk,pk)\leftarrow\$\mathsf{KeyGen}()\\ (k^*,c^*)\leftarrow\$\mathsf{Enc}(pk)\\ c\leftarrow\mathcal{A}(sk,pk,k^*,c^*)\end{smallmatrix}\right].$$

Note that the adversary also gets access to the secret key. This ensures that this notion will hold even if other security notions are broken, such as IND-CCA security. This allows us to prove the security of QSF relative to the strength of the nominal group for an arbitrarily bad KEM. With this definition, we can prove the IND-CCA security of QSF.

Note that this notion is related to the key-binding properties introduced by Cremers et al. in [CDM23] and Alwen et al. in [AHK+23, §5.3]. Instead of finding any collision, as for the properties introduced by Cremers et al. and Alwen et al., we opted for a second preimage resistance notion, as this best fits our requirements in the following proof. This is a strictly weaker notion, since the adversary is tasked to find a second preimage instead of any collision. Also this allows us to prove better concrete bounds on the achieved security level.

Theorem 1. *Let $\mathcal{N} = (\mathcal{G}, g, p, \varepsilon_h, \varepsilon_u, \mathsf{exp})$ be a nominal group and KEM be a key encapsulation mechanism, let H be a random oracle with an output size of n and let $\mathcal{A}$ be an adversary against the IND-CCA security of QSF in Definition 6 making at most q_h queries to the random oracles. Then there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ such that,*

$$Adv^{IND\text{-}CCA}_{QSF,\mathcal{A}} \leq 2\Delta_{\mathcal{N}} + Adv^{SDH}_{\mathcal{N},\mathcal{B}} + Adv^{C2PRI}_{KEM,\mathcal{C}}.$$

The run-times of $\mathcal{B}$ and $\mathcal{C}$ are roughly the same as of $\mathcal{A}$. $\mathcal{B}$ performs at most $2q_h$ queries to its own DH oracle.

Proof. Let us consider the following games,

Game 0 Let G_0 be defined as in Figure 5. In this game, Σ is used to model $H(m)$ as lazily sampled random oracle. Clearly, G_0 is the IND-CCA^0 game instantiated with QSF. By definition,

$$\Pr[\text{IND-CCA}^0_{\text{QSF},\mathcal{A}} \Rightarrow 1] = \Pr[G^{\mathcal{A}}_0 \Rightarrow 1].$$

Game 1 For G_1 we choose the secret keys sk_2 and sk_e from the set of exponents ε_u instead of the set of honest exponents ε_h. This allows us to use the SDH problem to bound the probability of the next game hop.

Claim 1

$$|\Pr[G^{\mathcal{A}}_0 \Rightarrow 1] - \Pr[G^{\mathcal{A}}_1 \Rightarrow 1]| \leq 2\Delta_{\mathcal{N}}.$$

Game $G_0/G_1/G_2/G_3/G_4$

$\sum[\cdot] \leftarrow \bot$
$(sk_1, pk_1) \leftarrow \text{KEM.KeyGen}()$
$sk_2 \leftarrow\$\ \varepsilon_h$
$sk_2 \leftarrow\$\ \varepsilon_u$ ▷ $G_1 - G_4$
$pk_2 \leftarrow \mathsf{exp}(g, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
$(k_1^*, c_1^*) \leftarrow \text{KEM.Enc}(pk_1)$
$sk_e \leftarrow\$\ \varepsilon_h$
$sk_e \leftarrow\$\ \varepsilon_u$ ▷ $G_1 - G_4$
$c_2^* \leftarrow \mathsf{exp}(g, sk_e)$
$k_2^* \leftarrow \mathsf{exp}(pk_2, sk_e)$
$c^* \leftarrow (c_1^*, c_2^*)$
$s \leftarrow \text{label} \| k_1^* \| k_2^* \| c_2^* \| pk_2$
if $\sum[s] = \bot$ **then**
  $\sum[s] \leftarrow\$\ \{0,1\}^n$
$k^* \leftarrow \sum[s]$
$k^* \leftarrow\$\ \{0,1\}^n$ ▷ G_4
$b' \leftarrow \mathcal{A}^{\mathsf{Dec}(\cdot), H(\cdot)}(pk, c^*, k^*)$
return b'

Oracle $\mathsf{Dec}(c)$

if $c = c^*$ **then**
  return $\bot$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \text{KEM.Dec}(c_1, sk_1)$
$k_2 \leftarrow \mathsf{exp}(c_2, sk_2)$
if $k_1 = \bot$ **then**
  return $\bot$
▷ $G_3 - G_4$ ◁
if $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then**
  $abort_2$
$s \leftarrow \text{label} \| k_1 \| k_2 \| c_2 \| pk_2$
if $\sum[s] = \bot$ **then**
  $\sum[s] \leftarrow\$\ \{0,1\}^n$
$k \leftarrow \sum[s]$
return k

$H(m)$

▷ $G_2 - G_4$ ◁
if $\text{label} \| k_1 \| k_2 \| c_2 \| pk_2 = m$ **then**
  if $k_2 = k_2^*$ **then**
    $abort_1$
if $\sum[m] = \bot$ **then**
  $\sum[m] \leftarrow\$\ \{0,1\}^n$
return $\sum[m]$

Figure 5: Game $G_0 - G_4$.

Proof With this game hop, we change the distribution from which these values are chosen. $\Delta_{\mathcal{N}}$ defines the bound on the probability of an adversary to distinguish the original distribution from the new one for one element. As we are replacing the distribution of two elements, we get $2\Delta_{\mathcal{N}}$ as an upper bound for $\mathcal{A}$ to detect this change.

Game 2 With the changes in G_2 we prevent the adversary from querying the random oracle containing the value k_2^* that was used to generate the challenge key k^*. This prevents $\mathcal{A}$ entirely from obtaining the shared challenge key k^* from the random oracle.

Claim 2

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq Pr[abort_1] = \text{Adv}^{\text{SDH}}_{\mathcal{N},\mathcal{B}}.$$

Proof To prove this claim, we show the existence of adversary $\mathcal{B}$ that wins the SDH game exactly when $\mathcal{A}$ would query H_2 containing the k_2^* used to generate k^*. Consider adversary $\mathcal{B}$ from Figure 6 playing the SDH game and simulating $\mathcal{A}$'s view on the IND-CCA game.

Adversary $\mathcal{B}$ uses the challenge group elements $X = \mathsf{exp}(g, x)$ as the static public key and $Y = \mathsf{exp}(g, y)$ as the ephemeral public key for the challenge ciphertext. Therefore, the shared challenge key k^* is supposed to be $H_2(label \| k_1^* \| \mathsf{exp}(Y, x) \| c_2^* \| pk_2)$. $\mathcal{B}$ is unable to compute $\mathsf{exp}(Y, x)$ and therefore sets k^* to a uniform random value. For $\mathcal{A}$ to win the IND-CCA game, it would have to query H_2 with $\mathsf{exp}(Y, x)$. $\mathcal{B}$ can use the DH oracle to detect when $\mathcal{A}$ queries H_2 with the result of $\mathsf{exp}(Y, x)$, since

Adversary $\mathcal{B}^{\mathsf{DH}(\cdot,\cdot)}(X, Y)$
$\sum[\cdot] \leftarrow \perp$
$\mathbf{E}[\cdot] \leftarrow \perp$
$(sk_1, pk_1) \leftarrow\$ \text{KEM.}\mathsf{KeyGen}()$
$pk_2 \leftarrow X$
$pk \leftarrow (pk_1, pk_2)$
$(k_1^*, c_1^*) \leftarrow\$ \text{KEM.}\mathsf{Enc}(pk_2)$
$c_2^* \leftarrow Y$
$c^* \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow\$ \{0,1\}^n$
$b' \leftarrow \mathcal{A}^{\mathsf{Dec}(\cdot),H(\cdot)}(pk, c^*, k^*)$

Oracle $\mathsf{Dec}(c)$
if $c = c^*$ **then**
  return $\perp$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \text{KEM.}\mathsf{Dec}(c_1, sk_1)$
if $k_1 = \perp$ **then**
  return $\perp$
if $\mathbf{E}[(c_2, k_1)] = \perp$ **then**
  $\mathbf{E}[(c_2, k_1)] \leftarrow\$ \{0,1\}^n$
return $\mathbf{E}[(c_2, k_1)]$

$H(m)$
if $\text{label}\|k_1\|k_2\|c_2\|pk_2 = m$ **then**
  if $\mathsf{DH}(Y, k_2) = 1$ **then**
    stop with k_2
  if $\mathsf{DH}(c_2, k_2) = 1$ **then**
    if $\mathbf{E}[(c_2, k_1)] = \perp$ **then**
      $\mathbf{E}[(c_2, k_1)] \leftarrow\$ \{0,1\}^n$
    return $\mathbf{E}[(c_2, k_1)]$
if $\sum[m] = \perp$ **then**
  $\sum[m] \leftarrow\$ \{0,1\}^n$
return $\sum[m]$

Figure 6: Adversary $\mathcal{B}$.

$\mathsf{DH}(Y, k_2^*) = 1$ iff $k_2^* = \mathsf{exp}(Y, x)$. Therefore, if this check in the H oracle evaluates to 1 we know that the k_2 provided has to equal k_2^*. When this happens, $\mathcal{A}$ has provided $\mathcal{B}$ with the solution to the SDH game. A similar approach can be used to simulate the Dec oracle. When calculating the shared key in the Dec oracle, the input for the random oracle has to have the form $label\|k_1\|\mathsf{exp}(c_2, x)\|c_2\|pk_2$ which $\mathcal{B}$ is not able to compute. $\mathcal{B}$ can use the same method as before to check whether $\mathcal{A}$ performs such a query using the DH oracle. We then use the table $\mathbf{E}$ to ensure consistency between the random oracle H and the Dec oracle. Therefore, $\mathcal{B}$ simulates $\mathcal{A}$'s view on the IND-CCA game perfectly and wins the SDH game iff the adversary queries H with the result of $\mathsf{exp}(Y, x)$, which is the k_2^* used to generate k^*. Note that the adversary $\mathcal{B}$ makes at most $2q_h$ queries to the DH oracle.

Game 3 G_3 introduces changes that prevent the adversary from querying the Dec oracle with a ciphertext c_1 that is different from the ciphertext c_1^* that was used to generate the challenge ciphertext c^*, but results in the same k_1 as the k_1^* that was used to generate k^*.

Claim 3

$$|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]| \leq Pr[abort_2] = \text{Adv}^{\text{C2PRI}}_{\text{KEM},\mathcal{C}}.$$

Proof To prove this claim, we show the existence of an adversary $\mathcal{C}$, depicted in Figure 7, that simulates $\mathcal{A}$'s view on the IND-CCA game and wins iff $\mathcal{A}$ would trigger $abort_2$.

All oracles are simulated perfectly. This is easily possible, since $\mathcal{C}$ also obtains the secret key from its challenger. $\mathcal{C}$ stops exactly when the $abort_2$ would be hit in G_3. When the stop instruction is reached, $\mathcal{C}$ obtained a k_1 that is equal to k_1^* but where c_1 and c_1^* are different. This means that $\mathcal{C}$ found a second ciphertext that decapsulates

Adversary $\mathcal{C}(sk_1, pk_1, k_1^*, c_1^*)$
$\sum[\cdot] \leftarrow \perp$
$sk_2 \leftarrow\!\!\$\ \varepsilon_u$
$pk_2 \leftarrow \mathsf{exp}(g, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
$sk_e \leftarrow\!\!\$\ \varepsilon_u$
$c_2^* \leftarrow \mathsf{exp}(g, sk_e)$
$k_2^* \leftarrow \mathsf{exp}(pk_2, sk_e)$
$c^* \leftarrow (c_1^*, c_2^*)$
$s \leftarrow \text{label} \| k_1^* \| k_2^* \| c_2^* \| pk_2$
if $\sum[s] = \perp$ **then**
  $\sum[s] \leftarrow\!\!\$\ \{0,1\}^n$
$k^* \leftarrow \sum[s]$
$b' \leftarrow \mathcal{A}^{\mathsf{Dec}(\cdot), H(\cdot)}(pk, c^*, k^*)$

Oracle $\mathsf{Dec}(c)$
if $c = c^*$ **then**
  return $\perp$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \text{KEM.}\mathsf{Dec}(c_1, sk_1)$
$k_2 \leftarrow \mathsf{exp}(c_2, sk_2)$
if $k_1 = \perp$ **then**
  return $\perp$
if $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then**
  stop with c_1
$k \leftarrow H_2(\text{label} \| k_1 \| k_2 \| c_2 \| pk_2)$
return k

$H(m)$
if $\text{label} \| k_1 \| k_2 \| c_2 \| pk_2 = m$ **then**
  if $k_2 = k_2^*$ **then**
    $abort_1$
if $\sum[m] = \perp$ **then**
  $\sum[m] \leftarrow\!\!\$\ \{0,1\}^n$
return $\sum[m]$

Figure 7: Adversary $\mathcal{C}$.

to the same shared key as the challenge key k_1^*. The ciphertext c_1 is therefore a correct answer for the C2PRI game.

Game 4 In G_4 we replace the shared challenge key k^* with a uniform random key. With this change we reached the IND-CCA^1 game instantiated with QSF. Therefore,

Claim 4

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[G_4^{\mathcal{A}} \Rightarrow 1].$$

Proof To prove that this change is undetectable by $\mathcal{A}$, we need to argue that $\mathcal{A}$'s view when interacting with the various oracles is independent of the value of k^*. The abort condition in H prevents the adversary from querying the random oracle with the value of k_2^* used to generate the shared key k^* from the challenge. Since the adversary $\mathcal{A}$ cannot query the random oracle H with the input used to generate k^*, the output of the random oracle H is independent of k^*. The abort condition in Dec also prevents the adversary from querying the Dec oracle in a way that would result in k^* being outputted. Either c_1 or c_2 must be different from c_1^* or c_2^*. If c_1 is different, then the resulting key k_1 must be different from k_1^*, otherwise the abort condition would be triggered. If c_2 is different, then the input to the random oracle must also be different, since it is included in its input. Therefore, the behavior of the Dec oracle is independent of the value of k^*. Since the behavior of all oracles is independent of k^*, the adversary cannot detect this game hop.

Now that k^* is chosen uniformly at random, which means that G_4 matches the definition of the IND-CCA$^1_{\text{QSF},\mathcal{A}}$ game perfectly. Therefore,

$$\Pr[G_4^{\mathcal{A}} \Rightarrow 1] = \Pr[\text{IND-CCA}^1_{\text{QSF},\mathcal{A}} \Rightarrow 1].$$

This concludes the proof of Theorem 1.

□

6.2 Reduction to security of KEM in the standard model

Theorem 2. *Let $\mathcal{N} = (\mathcal{G}, g, p, \varepsilon_h, \varepsilon_u, \mathsf{exp})$ be a nominal group and KEM be a key encapsulation mechanism, let H be a secure PRF with an output size of n when keyed on k_1, and let $\mathcal{A}$ be an adversary against the IND-CCA security of QSF in Definition 6. Then, there exist $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{E}$, such that,*

$$Adv^{QSF}_{IND\text{-}CCA,\mathcal{A}} \leq Adv^{KEM}_{IND\text{-}CCA,\mathcal{B}} + Adv^{KEM}_{IND\text{-}CCA,\mathcal{C}} + Adv^{H}_{PRF,\mathcal{D}} + Adv^{H}_{PRF,\mathcal{E}} + 2\delta\,,$$

where δ is the correctness bound for KEM. The run-times of $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{E}$ are roughly the same as of $\mathcal{A}$. $\mathcal{B}$ and $\mathcal{C}$ perform at most q_d queries to their own decapsulation oracles.

Proof. This proof follows closely the proof for [GHP18, Theorem 1].

Game $G_0/G_1/G_2/G_3/G_4$

$sk_1, pk_1 \leftarrow\$ \text{KEM.KeyGen}()$
$sk_2 \leftarrow\$ \varepsilon_h$
$pk_2 \leftarrow \mathsf{exp}(g, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
$sk \leftarrow (sk_1, sk_2)$
$k_1^*, c_1^* \leftarrow\$ \text{KEM.Enc}(pk_1)$
$sk_e \leftarrow\$ \varepsilon_u$
$c_2^* \leftarrow \mathsf{exp}(g, sk_e)$
$k_2^* \leftarrow \mathsf{exp}(pk_2, sk_e)$
$k_1^* \leftarrow\$ \mathcal{K}$ ▷ $G_1 - G_3$
$k^* \leftarrow H(\text{label}\|k_1^*\|k_2^*\|c_2^*\|pk_2)$
$k^* \leftarrow\$ \{0,1\}^n$ ▷ $G_2 - G_4$
$c^* \leftarrow (c_1^*, c_2^*)$
$b' \leftarrow\$ \mathcal{A}^{\text{Dec}(\cdot)}(pk, c^*, k^*)$
return b'

Oracle $\text{Dec}(c)$

if $c = c^*$ **then**
  return $\perp$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \text{KEM.Dec}(c_1, sk_1)$
$k_2 \leftarrow \mathsf{exp}(c_2, sk_2)$
if $k_1 = \perp$ **then**
  return $\perp$
if $c_1 = c_1^*$ **then** ▷ $G_1 - G_3$
  $k_1 \leftarrow k_1^*$
$k \leftarrow H(\text{label}\|k_1\|k_2\|c_2\|pk_2)$
if $c_1 = c_1^*$ **then** ▷ G_2
  $k \leftarrow\$ \{0,1\}^n$
return k

Figure 8: Games G_0 to G_4.

Game 0 This is the standard IND-CCA security game for QSF and so,

$$\Pr[\text{IND-CCA}^0_{\text{QSF},\mathcal{A}} \Rightarrow 1] = \Pr[G_0 \Rightarrow 1].$$

Game 1 The challenge shared key produced by KEM is replaced with a random key.

Claim 1 This change should not be noticeable by the adversary $\mathcal{A}$. If it is, then we can construct an efficient adversary $\mathcal{B}$ against the IND-CCA security of KEM with roughly the same running time as $\mathcal{A}$, such that,

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \text{Adv}^{\text{IND-CCA},\mathcal{B}}_{\text{KEM}} + \delta.$$

Proof We construct $\mathcal{B}$ as in Figure 9. The adversary $\mathcal{B}$ simulates the environment of $\mathcal{A}$ by calculating the nominal group components of QSF itself and embedding the KEM challenge into the QSF challenge. The decapsulation oracle is simulated by $\mathcal{B}$ as follows: it can calculate the nominal group part itself and query its own KEM decapsulation oracle on all ciphertexts except c_1^*. Because of this, it will simply use its own challenge shared key k_1^* if c_1^* is in the decapsulation query by $\mathcal{A}$.

The strategy adopted by $\mathcal{B}$ correctly interpolates between games G_0 and G_1, except when c_1^* would decapsulate to something other than k_1^* in G_0. We can bound the probability of this inconsistency using the correctness of KEM, which justifies the δ term in the claim. More precisely, we have

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G^0_{\mathrm{KEM},\mathcal{B}} \Rightarrow 1]\right| \leq \delta,$$

and

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G^1_{\mathrm{KEM},\mathcal{B}} \Rightarrow 1],$$

which means that

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathrm{Adv}^{\mathrm{KEM}}_{\text{IND-CCA},\mathcal{B}} + \delta,$$

and $\mathcal{B}$ will clearly perform at most as many decapsulation queries to its KEM decapsulation oracle as $\mathcal{A}$ makes decapsulation queries.

Game 2 In this game, all QSF shared keys that are provided to the adversary computed using k_1^* are replaced with random values.

Claim 2 This change should not be noticeable by the adversary $\mathcal{A}$. If it is, then we can construct an efficient adversary $\mathcal{D}$ against the PRF security of H with roughly the same running time as $\mathcal{A}$, such that

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathrm{Adv}^{H}_{\mathrm{PRF},\mathcal{D}}.$$

Proof This is a simple reduction shown in Figure 10. This adversary can generate all cryptographic parameters except k_1^*, for which it uses its PRF oracle. As we can see, when the challenge PRF key is real, then $\mathcal{D}$ will be using a real output of the pseudorandom function and hence running G_1; whereas if the challenge key is random, $\mathcal{D}$ will be using a uniformly sampled output and running G_2. Hence,

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G^0_{H,\mathcal{D}} \Rightarrow 1],$$

and

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G^1_{H,\mathcal{D}} \Rightarrow 1],$$

which means that

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathrm{Adv}^{H}_{\mathrm{PRF},\mathcal{D}},$$

and $\mathcal{D}$ will perform one evaluation query for generating the challenge key and perform at most one evaluation query per decapsulation query by $\mathcal{A}$.

Game 3 We undo the modification introduced in the previous game, but only for keys output by the decapsulation oracle.

Claim 3 We justify this hop with a reduction to the PRF property of H very similar to the previous one.

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathrm{Adv}^{H}_{\mathrm{PRF},\mathcal{E}}.$$

Proof We construct $\mathcal{E}$ as in Figure 10, and the analysis is similar to the previous hop.

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{PRF}^0_{H,\mathcal{E}} \Rightarrow 1],$$

and

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{PRF}^1_{H,\mathcal{E}} \Rightarrow 1],$$

which means that

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathrm{Adv}^{H}_{\mathrm{PRF},\mathcal{E}},$$

and $\mathcal{E}$ will perform at most one evaluation query per decapsulation query by $\mathcal{A}$ as it will never call the evaluation oracle in its main algorithm.

Game 4 Finally, we revert the changes introduced in Game 1 and use a real key for k_1^* once more. This makes the decapsulation oracle identical to Game 0, and the only remaining change in the main experiment is that k^* is random.

Claim 4 This hop is justified in a way that is very similar to the jump to Game 1. We introduce adversary $\mathcal{C}$ against the IND-CCA security of KEM, with roughly the same running time as $\mathcal{A}$, in Figure 9. We claim that,

$$\left|\Pr[G_3^{\mathcal{A}} \Rightarrow 1] - \Pr[G_4^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathrm{Adv}^{\mathrm{KEM}}_{\text{IND-CCA},\mathcal{C}} + \delta\,.$$

Proof Again, the reduction to the IND-CCA security of KEM is perfect, except if c_2^* would not decapsulate to k_1^* in Game 4. This event can be bound by the correctness of KEM, and the claim follows.

Claim 5 $\Pr[G_4^{\mathcal{A}} \Rightarrow 1] = \Pr[\text{IND-CCA}^1_{\mathrm{QSF},\mathcal{A}} \Rightarrow 1]$

Proof This follows from the definition of $\text{IND-CCA}^1_{\mathrm{QSF},\mathcal{A}}$.

The theorem follows from collecting all the terms in the claims.

Adversary $\mathcal{B}^{\mathsf{Dec}_O(\cdot)}/\mathcal{C}^{\mathsf{Dec}_O(\cdot)}(pk_1, k_1^*, c_1^*)$

$sk_2 \leftarrow\!\$\ \varepsilon_h$
$pk_2 \leftarrow \mathsf{exp}(g, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
$sk_e \leftarrow\!\$\ \varepsilon_u$
$k_1, c_1^* \leftarrow\!\$\ \mathrm{KEM.Enc}(pk_1)$
$c_2^* \leftarrow \mathsf{exp}(g, sk_e)$
$k_2^* \leftarrow \mathsf{exp}(pk_2, sk_e)$
$k^* \leftarrow H(\text{label} \| k_1^* \| k_2^* \| c_2^* \| pk_2)$
$k^* \leftarrow\!\$\ \{0,1\}^n$ ▷ $\mathcal{C}$
$c^* \leftarrow (c_1^*, c_2^*)$
$b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{Dec}(\cdot)}(pk, c^*, k^*)$
return b'

Oracle $\mathsf{Dec}(c)$

if $c = c^*$ **then**
  return $\perp$
$(c_1, c_2) \leftarrow c$
$k_2 \leftarrow \mathsf{exp}(c_2, sk_2)$
if $c_1 = c_1^*$ **then**
  $k_1 \leftarrow k_1^*$
else $k_1 \leftarrow \mathsf{Dec}_O(c_1)$
if $k_1 = \perp$ **then**
  return $\perp$
$k \leftarrow H(\text{label} \| k_1 \| k_2 \| c_2 \| pk_2)$
return k

Figure 9: Adversary $\mathcal{B}$ and $\mathcal{C}$ against the IND-CCA security of KEM. Note that both place at most q_d queries to Dec_O, one for each decapsulation query placed by $\mathcal{A}$.

□

Adversary $\mathcal{D}^{\mathsf{Eval}_H(\cdot)}/\mathcal{E}^{\mathsf{Eval}_H(\cdot)}()$
$sk_1, pk_1 \leftarrow\$ \text{KEM.KeyGen}()$
$sk_2 \leftarrow\$ \varepsilon_h$
$pk_2 \leftarrow \mathsf{exp}(g, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
$sk \leftarrow (sk_1, sk_2, pk_2)$
$k_1, c_1^* \leftarrow\$ \text{KEM.Enc}(pk_1)$
$sk_e \leftarrow\$ \varepsilon_u$
$c_2^* \leftarrow \mathsf{exp}(g, sk_e)$
$k_2^* \leftarrow \mathsf{exp}(pk_2, sk_e)$
$k^* \leftarrow \mathsf{Eval}_H(\text{label}\|k_2^*\|c_2^*\|pk_2)$
$k^* \leftarrow\$ \{0,1\}^n$ ▷ $\mathcal{E}$
$c^* \leftarrow (c_1^*, c_2^*)$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{Dec}(\cdot)}(pk, c^*, k^*)$
return b'

Oracle $\mathsf{Dec}(c)$
if $c = c^*$ **then**
  return $\perp$
$(c_1, c_2) \leftarrow c$
$k_2 \leftarrow \mathsf{exp}(c_2, sk_2)$
if $c_1 = c_1^*$ **then**
  $k \leftarrow \mathsf{Eval}_H(\text{label}\|k_2\|c_2\|pk_2)$
  return k
$k_1 \leftarrow \text{KEM.Dec}(sk_1, c_1)$
if $k_1 = \perp$ **then**
  return $\perp$
$k \leftarrow H(\text{label}\|k_1\|k_2\|c_2\|pk_2)$
return k

Figure 10: Adversary $\mathcal{D}$ and $\mathcal{E}$ against the PRF security of H.

7 X-Wing

Now that we have proven the security of the QSF framework, we want to introduce one concrete instantiation of QSF using X25519, ML-KEM-768 and SHA3-256. We believe that this instantiation provides a secure and efficient KEM suitable for most applications. As a name for this instantiation, we choose X-Wing. For our QSF proof to apply to X-Wing we need to show that X25519 can be modeled as a nominal group and that ML-KEM-768 is C2PRI secure. This is done in this section, followed by the definition of X-Wing.

7.1 X25519 is a nominal group

X-Wing uses X25519 as specified in [LHT16]. As shown in [ABH+21] X25519 can be modeled as a nominal group $\mathcal{N} = (\mathcal{G}, g, p, \varepsilon_h, \varepsilon_u, \mathsf{exp})$. We summarize the results in this section.

We first define $\mathcal{G}$ to be all 256 bit long bitstrings. We then define an encoding function encode_pk as well as a decoding function decode_pk, mapping point on Curve25519 to a bitstring and vice versa. For this, we can use the encoding and decoding functions as defined in [LHT16]. p is the order of the largest prime order subgroup and g is a Curve25519 point of order p, such that the number of points on Curve25519 is $8p$. $\varepsilon_h = \{8n | n \in [2^{251}, 2^{252} - 1]\}$ stands for all valid secret keys, respectively for the key generation function of X25519. $\mathsf{exp}(X, y) = \mathsf{encode\_pk}(y \cdot \mathsf{decode\_pk}(X))$, which corresponds to the X25519.DH function. Let $\varepsilon_u = \{8n | n \in [(p+1)/2, p-1]\}$. This provides us with a statistical difference between the uniform distribution over ε_u and ε_h of $\Delta_{\mathcal{N}} < 2^{-126}$ [ABH+21].

Therefore, the definition of this nominal group matches the definition of X25519 perfectly.

7.2 Second Preimage resistance of ML-KEM-768

For the security of X-Wing it is important that ML-KEM-768 is C2PRI secure. Hence, we will show the ciphertext second preimage resistance of ML-KEM-768 in this section.

Theorem 3 (ML-KEM-768 is C2PRI secure)**.** *Let k, d_u, d_v be integers greater than or equal to 1, let PKE.Enc and PKE.Dec be deterministic algorithms and let G and J be*

independent random oracles with output sizes of 512 and 256 bits, and let $\mathcal{A}$ be an adversary against the C2PRI security of ML-KEM-768 making at most q_g and q_j many hash queries to the random oracles G and J. Then, the advantage of adversary $\mathcal{A}$ against the C2PRI security of ML-KEM-768 is defined as,

$$Adv^{C2PRI}_{\text{ML-KEM-768},\mathcal{A}} \leq \frac{q_g + q_j + 2}{2^{256}}.$$

Proof. Let us recall the definition of the decapsulation algorithm of ML-KEM-768, depicted in Figure 11.

ML-KEM-768.Dec$(sk \in \{0,1\}^{768k+96}, c \in \{0,1\}^{256(d_u k + d_v)})$
$sk_{PKE} \leftarrow sk[0 : 384k]$
$pk_{PKE} \leftarrow sk[384k : 768k + 32]$
$h \leftarrow sk[768k + 32 : 768k + 64]$
$z \leftarrow sk[768k + 64 : 768k + 96]$
$m' \leftarrow \text{PKE.Dec}(sk_{PKE}, c)$
$(K, r') \leftarrow G(m' \| h)$
$\overline{K} \leftarrow J(z \| c, 32)$
$c' \leftarrow \text{PKE.Enc}(pk_{PKE}, m', r')$
if $c \neq c'$ **then**
  $K \leftarrow \overline{K}$
return K

Figure 11: ML-KEM-768 decapsulation [NIS23]

We notice that the shared key k^* of the challenge, for which the adversary tries to find a second preimage, is generated by the random oracle G, since this key was honestly generated using the ML-KEM-768.Enc procedure. So we have three cases to analyze. The first is that the adversary finds a second ciphertext c that is different from the challenge ciphertext c^* but results in the same input to G. In the second case, we need to analyze the probability that G will output the same shared secret with a different input. For the last case, we need to analyze the probability that J will output the same shared secret as the one from the challenge.

Case 1: The input to G collides. Suppose PKE.Dec outputs the same m' for two different ciphertexts c_1 and c_2. Then the intermediate values K and r' are the same for both decapsulations. Since PKE.Enc is deterministic, c' is the same for both decapsulations. This also means that the equality check $c_i = c'$ cannot succeed for both ciphertexts. This leads to a contradiction, since at least one of the ciphertexts has to be rejected and therefore the key generated by J is returned by ML-KEM-768.Dec.

Case 2: The output of G collides. The output of G is interpreted as the two values K and r'. An adversary can already find a second preimage in the decapsulation function if it can find a second input to G that results in the same output K. Since G is modeled as a random oracle, and therefore behaves like a random function, the best the adversary can do is query the random oracle and hope to find such an input by chance. Therefore, the probability of an adversary finding such an input is bounded by $\frac{q_g+1}{2^{256}}$, where the $\frac{1}{2^{256}}$ comes from the possibility that the adversary does not query the random oracle on this value, but a collision-causing query is performed during the decryption done by the game.

Case 3: The output of J collides. Since J is a random oracle independent of G, the best an adversary can do is to query the random oracle J to find such an input by chance.

Therefore, as in the previous case, the probability of finding such an input can also be bounded by $\frac{q_j+1}{2^{256}}$.

This concludes the proof for Theorem 3. □

Note that all KEMs that are the result of the Fujisaki-Okamoto transformation and use explicit rejection with independent random oracles are also C2PRI secure, since all arguments hold for them as well.

7.3 Definition of X-Wing

Definition 8 (X-Wing KEM)**.** Given the ML-KEM-768 KEM, X25519 Diffie-Hellman key exchange, and the SHA3-256 hash function, the X-Wing KEM is defined as a tuple of algorithms $\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$, which are in turn defined in Figure 12.

Algorithm $\mathsf{KeyGen}()$
$sk_1, pk_1 \leftarrow\$ \text{ ML-KEM-768.KeyGen}()$
$sk_2 \leftarrow\$ \mathsf{random}(32)$
$pk_2 \leftarrow \text{X25519.DH}(sk_2, g_{\text{X25519}})$
$sk \leftarrow (sk_1, sk_2, pk_2)$
$pk \leftarrow (pk_1, pk_2)$
return (sk, pk)

Algorithm $\mathsf{Enc}(pk)$
$(pk_1, pk_2) \leftarrow pk$
$sk_e \leftarrow\$ \mathsf{random}(32)$
$c_2 \leftarrow \text{X25519.DH}(sk_e, g_{\text{X25519}})$
$k_1, c_1 \leftarrow\$ \text{ ML-KEM-768.Enc}(pk)$
$k_2 \leftarrow \text{X25519.DH}(sk_e, pk_2)$
$s \leftarrow \texttt{"\./\/^\"} \| k_1 \| k_2 \| c_2 \| pk_2$
$k \leftarrow \text{SHA3-256}(s)$
$c \leftarrow (c_1, c_2)$
return (k, c)

Algorithm $\mathsf{Dec}(c, sk)$
$(sk_1, sk_2, pk_2) \leftarrow sk$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \text{ML-KEM-768.Dec}(c_1, sk_1)$
$k_2 \leftarrow \text{X25519.DH}(sk_2, c_2)$
$s \leftarrow \texttt{"\./\/^\"} \| k_1 \| k_2 \| c_2 \| pk_2$
$k \leftarrow \text{SHA3-256}(s)$
return k

Figure 12: X-Wing KEM. $\mathsf{random}(N)$ generates N random bytes. X25519.DH is the byte-oriented function X25519 defined in section 5 of RFC 7748. g_{X25519} is the X25519 base point where $u = 9$ on curve25519 defined in section 6.1 of RFC 7748. ML-KEM-768.KeyGen, ML-KEM-768.Enc, and ML-KEM-768.Dec are defined in FIPS 203. SHA3-256 is defined in FIPS 202 [Dwo15]. The X-Wing label is inlined as the first 6 ASCII-encoded bytes of the SHA3-256 input.

X-Wing has been brought to the IETF to be standardized [CSW24] as an RFC. The final definition may differ.

8 Benchmarks

We developed a reference and optimized X-Wing implementation in the C programming language[3] and integrated it in the benchmarking environment of Libjade[4]. The code

[3] https://github.com/x-wing-kem-team/xwing
[4] https://github.com/formosa-crypto/libjade

is available from https://github.com/x-wing-kem-team/xwing-benchmarks. The code package also includes a reference and optimized variant of X-Wing that includes the ML-KEM-768 ciphertext in the input to SHA3-256. In the following we refer to this variant as X-Wing-Hash-CT. Hence, X-Wing-Hash-CT uses 1222 bytes as input for the final key derivation, while X-Wing uses only 134 bytes by omitting the ML-KEM-768 ciphertext. Benchmarking both variants allows us to quantify the savings in computational cost we obtain by proving that it is secure to not include the ML-KEM-768 ciphertext in the key-derivation hash in X-Wing. We also note that this is not the only performance optimization in X-Wing. Besides not including the large ML-KEM-768 ciphertext in the final key derivation, we also proved that it is sufficient to use X25519 as the Diffie-Hellman key exchange directly instead of using a KEM based on X25519. This also saves an additional key derivation step that would have been performed by the KEM based on X25519. We decided not to benchmark this, as it is highly dependent on the X25519 based KEM we are comparing to. However, we believe that the results for omitting the ML-KEM-768 ciphertext in the final key derivation is already a strong argument for X-Wing.

We chose to use the PQ-Crystals reference and optimized implementation of ML-KEM[5] and we also utilize their reference and optimized implementation of SHA3-256, which itself is based on TweetFIPS202[6] and the `crypto_hash/keccakc512/simple/` SUPERCOP implementation[7]. For all Diffie-Hellman operations, we used lib25519[8].

8.1 Benchmarking procedure

Procedure. The benchmarking focused on counting the CPU clock-cycles used by the key generation, encapsulation and decapsulation algorithms for both X-Wing and X-Wing-Hash-CT. Furthermore, we also demonstrated the CPU clock-cycles required for SHA3-256 to process a 134 byte, and a 1222 byte input. These benchmarks were run on an x86-64 Debian 12 machine with an 11th Gen Intel Core i7-11700K with its frequency set to 3.5GHz and with 16Gb of RAM. Turbo-boost and hyperthreading were disabled. We used GCC version 13.2.0 with the flags taken directly from the AVX2 PQ-Crystals ML-KEM-768 implementation, namely `-mavx2 -mbmi2 -mpopcnt -march=native -mtune=native -O3 -fomit-frame-pointer -z noexecstack`. Each implementation's various algorithms (i.e. key generation, encaps and decaps) is run 100 times and the benchmark that we used is the median of the results.

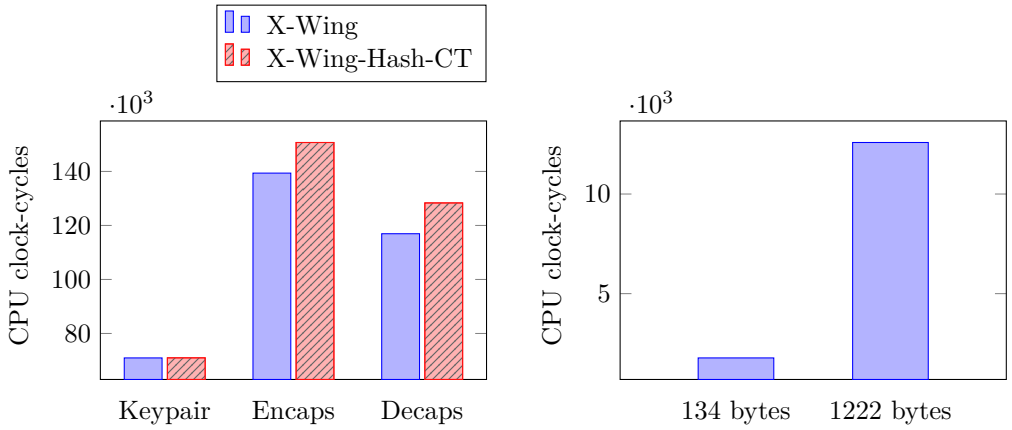
Safety and correctness. The X-Wing and X-Wing-Hash-CT implementations were verified to be memory-safe via Valgrind and have 99% test coverage. These include functionality tests (e.g., if the decapsulated secret is equal to the secret produced by encapsulation) and, if applicable, verify that the implementation's output matches the test vectors in defined in [CSW24].

8.2 Benchmarking results

Figure 14 contains tables that display the results of the benchmarks for X-Wing and X-Wing-Hash-CT, Table (a), and the SHA3-256, Table (b), optimized implementations. The results are plotted as a bar graph in Figure 13, whereby Figure 13a contains the plot of X-Wing and X-Wing-Hash-CT benchmarks, and Figure 13b contains the plot of the SHA3-256 benchmarks.

[5] https://github.com/pq-crystals/kyber/tree/standard
[6] https://keccak.team/2015/tweetfips202.html
[7] http://bench.cr.yp.to/supercop.html
[8] https://lib25519.cr.yp.to/

By omitting the ciphertext from the input to SHA3-256, we can see a 8% and 9% performance gain for encapsulation and decapsulation, respectively. This is in line with the SHA3-256 benchmarks, whereby Table c shows the clock-cycles of the encapsulation and decapsulation operations if we remove the respective clock-cycles that are due to calculating the hash. We see that the clock-cycles for both implementations coincide, with the small discrepancies being due to an extra `memcpy` being used in the X-Wing-Hash-CT.



(a) Benchmarks for X-Wing and X-Wing-Hash-CT AVX2 implementations.

(b) Benchmarks for SHA3-256 implementation with different input sizes.

Figure 13: Benchmark plots.

Implementation	**Keypair**	**Encaps**	**Decaps**
X-Wing	70942	139374	116930
X-Wing-Hash-CT	70976	150708	128336
Ratio	*1.00*	*1.08*	*1.10*

(a) Comparison of CPU clock-cycles when running X-Wing and X-Wing-Hash-CT.

Input length	**Hash**
134 bytes	1774
1222 bytes	12586
Ratio	*7.10*

(b) Comparison of CPU clock-cycles when running SHA3-256 when using 134 and 1222 byte input.

Implementation	**Encaps − Hash**	**Decaps − Hash**
X-Wing	137600	115156
X-Wing-Hash-CT	138122	115750
Ratio	*1.00*	*1.01*

(c) Comparison of CPU clock-cycles when running X-Wing and X-Wing-Hash-CT, minus the respective cost of the SHA3-256 operation.

Figure 14: Benchmark tables.

9 Conclusion

In this paper, we introduced the new hybrid KEM, X-Wing, which combines the security of X25519 and ML-KEM-768 to provide a robust KEM that is secure even if the security

of one of the schemes is broken. We improve on generic KEM combiners by not including the ML-KEM-768 ciphertext in the final key derivation, and by using X25519 as is instead of a KEM based on X25519. For all of these changes, we provide proofs to ensure that no vulnerabilities are introduced by these changes. We also show that these changes have a significant impact on the performance of the scheme by providing benchmarks of the proposed schemes. With all of this, we believe that X-Wing is a good choice for a hybrid KEM in most use cases.

Acknowledgments

We would like to thank all reviewers, our shepherd, and everybody who contributed to discussions about X-Wing on IETF's CFRG mailing list. The feedback and discussions helped us a lot to improve this paper.

This research was supported by Deutsche Forschungsgemeinschaft (DFG, German research Foundation) as part of the Excellence Strategy of the German Federal and State Governments – EXC 2092 CASA - 390781972; and by the European Commission through the ERC Starting Grant 805031 (EPOQUE); and by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDP/50014/2020.

References

[ABH+21] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. Analysing the HPKE standard. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 87–116, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-77870-5_4.

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158, San Francisco, CA, USA, April 8–12, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-45353-9_12.

[AHK+23] Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. Post-Quantum Multi-Recipient Public Key Encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1108–1122, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3576915.3623185.

[BBLW22] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid Public Key Encryption. RFC 9180, February 2022. URL: https://www.rfc-editor.org/info/RFC9180, doi:10.17487/RFC9180.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228, New York, NY, USA, April 24–26, 2006. Springer, Heidelberg, Germany. doi:10.1007/11745853_14.

[CDM23] Cas Cremers, Alexander Dax, and Niklas Medinger. Keeping Up with the KEMs: Stronger Security Notions for KEMs. Cryptology ePrint Archive, Paper 2023/1933, 2023. https://eprint.iacr.org/2023/1933. URL: https://eprint.iacr.org/2023/1933.

[CSW24] Deirdre Connolly, Peter Schwabe, and Bas Westerbaan. X-Wing: general-purpose hybrid post-quantum KEM. draft-connolly-cfrg-xwing-kem, 2024. URL: https://datatracker.ietf.org/doc/draft-connolly-cfrg-xwing-kem/.

[Dwo15] Morris J. Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology (NIST), July 2015. URL: http://dx.doi.org/10.6028/NIST.FIPS.202, doi:10.6028/nist.fips.202.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-76578-5_7.

[LHT16] Adam Langley, Mike Hamburg, and Sean Turner. Elliptic Curves for Security. RFC 7748, January 2016. URL: https://www.rfc-editor.org/info/rfc7748, doi:10.17487/RFC7748.

[NIS23] NIST. Module-Lattice-Based Key-Encapsulation Mechanism Standard. FIPS 203 (Initial Public Draft), August 2023. doi:10.6028/NIST.FIPS.203.ipd.

[O'B23] Devon O'Brien. Protecting Chrome Traffic with Hybrid Kyber KEM. Chromium Blog, 2023. https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html.

[OWK23] Mike Ounsworth, Aron Wussler, and Stavros Kousidis. Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs). Internet-Draft draft-ounsworth-cfrg-kem-combiners-04, Internet Engineering Task Force, July 2023. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ounsworth-cfrg-kem-combiners/04/.

[WR22] Bas Westerbaan and Cefan Rubin. Defending against future threats: Cloudflare goes post-quantum. The Cloudflare Blog, 2022. https://blog.cloudflare.com/post-quantum-for-all/.

[WS23] Bas Westerbaan and Douglas Stebila. X25519Kyber768Draft00 hybrid post-quantum key agreement. draft-westerbaan-cfrg-hpke-xyber768d00, 2023. URL: https://datatracker.ietf.org/doc/draft-tls-westerbaan-xyber768d00/03/.

[WW23] Bas Westerbaan and Christopher A. Wood. X25519Kyber768Draft00 hybrid post-quantum KEM for HPKE. draft-westerbaan-cfrg-hpke-xyber768d00, 2023. URL: https://datatracker.ietf.org/doc/draft-westerbaan-cfrg-hpke-xyber768d00/.