Security Analysis of Signal's PQXDH Handshake

Rune Fiedler[1] Felix Günther[2]

[1] Cryptoplexity, Technische Universität Darmstadt, Germany
[2] IBM Research Europe – Zurich, Switzerland
rune.fiedler@cryptoplexity.de, mail@felixguenther.info

Abstract. Signal recently deployed a new handshake protocol named PQXDH to protect against "harvest now, decrypt later" attacks of a future quantum computer. To this end, PQXDH adds a post-quantum KEM to the Diffie–Hellman combinations of the prior X3DH handshake.

In this work, we give a reductionist security analysis of Signal's PQXDH handshake in a game-based security model that captures the targeted "maximum-exposure" security, allowing fine-grained compromise of user's long-term, semi-static, and ephemeral key material. We augment prior such models to capture not only the added KEM component but also the signing of public keys, which prior analyses did not capture but which adds an additional flavor of post-quantum security in PQXDH. We then establish a fully parameterized, concrete security bound for the session key security of PQXDH, in particular shedding light on a KEM binding property we require for PQXDH's security, and how to avoid it.

Our discussion of KEM binding complements the tool-based analysis of PQXDH by Bhargavan, Jacomme, Kiefer, and Schmidt, which pointed out a potential re-encapsulation attack if the KEM shared secret does not bind the public key. We show that both Kyber (used in PQXDH) and its current NIST draft standard ML-KEM (foreseen to replace Kyber once standardized) satisfy a novel binding notion we introduce and rely on for our PQXDH analysis, which may be of independent interest.

1 Introduction

Billions of people today use messaging apps such as Facebook Messenger, Google Messages, Skype, Signal, or WhatsApp, in which the Signal end-to-end encryption protocol [Sig] secures the communication. The Signal protocol consists of two main components: The initial handshake protocol, which allows two parties to derive a shared session key while authenticating each other, and the Double Ratchet protocol, which allows renewing the session key in an ongoing session to achieve forward secrecy and post-compromise security. Until 2023, Signal used X3DH [MP16] as initial handshake protocol. In 2023, Signal released PQXDH [KS23, KS24], which modifies X3DH to add protection against quantum adversaries; in particular against "harvest now, decrypt later" attacks where an adversary records communication today to break its confidentiality when a cryptographically-relevant quantum computer becomes available in the future.

In X3DH, each user has Diffie–Hellman (DH) keys of different lifetimes (long-term, semi-static, and ephemeral). Combining several DH keys of two users Alice and Bob in the initial key agreement ensures mutual authentication and, as long as one of the combinations of keys remains uncompromised, that the derived session key is secure. To protect against "harvest now, decrypt later" quantum adversaries, PQXDH [KS24] adds a quantum-safe key encapsulation mechanism (KEM) to X3DH while keeping the existing, well-understood [CCD+17] handshake structure unmodified. Furthermore, X3DH signs the involved semi-static DH key and PQXDH additionally the KEM key under the the user's long-term key.

In a nutshell, the protocol message flow is as follows (where the KEM appears only in PQXDH): Bob sends his long-term, semi-static, and ephemeral DH shares, as well as an ephemeral KEM public key for a full handshake; a reduced handshake omits the ephemeral keys and has a semi-static KEM public key instead. Alice generates an ephemeral DH key pair, encapsulates against Bob's KEM key, and sends her ephemeral DH key and KEM ciphertext to Bob. Both parties derive the session key from DH shared secrets from three or four DH key combinations (long-term/semi-static, ephemeral/long-term, ephemeral/semi-static, and, if present, ephemeral/ephemeral) and the KEM shared secret.

Previous Security Analyses of X3DH and PQXDH

The development of PQXDH was accompanied by formal, tool-based verification conducted by Bhargavan, Jacomme, Kiefer, and Schmidt (BJKS) [BJKS23, BJK23]. Their analysis provided improvements and security assurance for PQXDH (leading to revisions of the protocol description [KS23, KS24]), using both the symbolic-analysis tool ProVerif [BC] and the computational-analysis tool CryptoVerif [Bla]. Security of Signal's original X3DH handshake (as well as its ratcheting protocol) was established by Cohn-Gordon, Cremers, Dowling, Garratt, and Stebila (CCDGS) [CCD$^+$17] through a reductionist security analysis in what we refer to as a "maximum-exposure"[1], game-based security model. Vatandas, Gennaro, Ithurburn, and Krawczyk [VGIK20] analyzed the deniability of X3DH, and Fiedler and Janson [FJ24] the deniability of PQXDH.

The core difference between these prior analyses of PQXDH and X3DH is that while the tool-based analyses of BJKS provide machine-checked assurance, the ProVerif results remain on the symbolic, protocol-logic level and the CryptoVerif model does not account for all "maximum-exposure" attack vectors that Signal aims to protect against. Concretely, for proof complexity reasons, the CryptoVerif model of BJKS [BJKS23, BJK23] only captures the compromise of long-term user keys (but not of semi-static and ephemeral keys) and separately considers attacks against the classic DH and the post-quantum KEM security. Such compromise, however, is a main reason for the many key combination Signal's X3DH and PQXDH handshake, and combined classic/post-quantum hybrid guarantees a main reason behind the PQXDH design specifically.

Our Contributions

This work completes the analysis picture for PQXDH by providing a reductionist security analysis of PQXDH (Revision 3 [KS24], throughout this paper) in a "maximum-exposure", game-based security model following that for X3DH by CCDGS [CCD$^+$17].

Augmented game-based security model. To this end, we first augment the coded game-based security model of Brendel, Fiedler, Günther, Janson, and Stebila (BFGJS) [BFG$^+$22, BFG$^+$21] to capture the added KEM component in PQXDH. In particular, we model that the KEM key and the semi-static DH keys in PQXDH are signed (instead of assuming them to be authentically distributed like in the prior game-based models [CCD$^+$17, BFG$^+$22, BFG$^+$21]). This requires careful adaptation of related corruption modeling, which adds both model and proof complexity but leads to a security model which more closely[2] captures the real-world deployment.

[1] The adversary may compromise nearly all secrets as long as trivial wins are excluded.

[2] Security modeling still always requires trade-offs in abstraction. For example, like all prior analyses [CCD$^+$17, BFG$^+$22, BFG$^+$21, BJKS23, BJK23], we assume dedicated signing keys whereas Signal's implementation re-uses the long-term DH user keys for signing.

Concrete security bound for PQXDH. We then analyze the PQXDH handshake, establishing its security based on the Gap Diffie–Hellman (GapDH) assumption holding in the DH group, the KEM providing one-way security under chosen-ciphertext attacks (OW-CCA), a certain form of binding property (which we expand on below), and low key-collision probability and correctness errors of the KEM, the signature scheme being existentially unforgeable, and the key derivation function behaving like a random oracle. In particular, we confirm that combining DH and KEM keys yields a combined, *hybrid* security bound, compared to the separate analysis in BJKS [BJKS23]. Our resulting security theorem for PQXDH is fully parameterized, giving concrete security bounds for each component.

Design discussion and KEM binding. We conclude with a discussion of the PQXDH design, with insights that complement those by BKJS [BJKS23]. In particular, we provide a point of view on the key derivation approach taken by Signal's X3DH and PQXDH, why it leads to requiring a certain *binding* property from the KEM in our analysis, and how this requirement could be easily avoided.

For background, BKJS in their analysis [BJKS23, BJK23] discovered the following potential "KEM re-encapsulation attack" on PQXDH: An adversary can learn a KEM shared secret by compromising the involved KEM secret key, and then re-encapsulating this shared secret under another, uncompromised KEM public key. In consequence, decapsulations under two distinct public keys yield the same shared secret, and two sessions with different KEM public keys compute the same session key, violating the protocol's security goals.

The BKJS analysis excludes this attack by modeling that the KEM public key is included in the associated data when AEAD-encrypting messages, a suggestion that the PQXDH description however only follows optionally: it argues that Kyber, used in the Signal implementation, already prevents the re-encapsulation attack [KS24, Sections 3.3 and 4.12]. Our analysis covers Signal's approach, providing fine-grained insight into the properties required of the KEM to ensure security of the initial key agreement, *without* relying on the AEAD. Concretely, we require that the derived shared secret *binds* the involved public key and ciphertext, in a setting where the adversary knows the involved keys *and the randomness* to generate them. We prove that both Kyber [SAB$^+$] (currently deployed in PQXDH) and the current NIST draft of ML-KEM [NIS23] (provisioned to replace Kyber in PQXDH once standardized) satisfy this binding property.

We view these KEM binding results as being of independent interest: They complement a range of binding notions for KEMs defined in concurrent work by Cremers, Dax, and Medinger (CDM) [CDM23], yet their notion that would fit the PQXDH setting is *not* satisfied by ML-KEM [Sch24]. We discuss the framework of CDM in more detail in Section 3 and relate the binding notion we introduce and require in the PQXDH analysis to theirs. Binding properties also arise in other key-combiner settings, e.g., in multi-recipient KEMs [AHK$^+$23] and X-Wing [BCD$^+$24], where the shared secret shall bind the ciphertext.

Related work

Several fully post-quantum replacements for X3DH have been considered in the literature: Hashimoto, Katsumata, Kwiatkowski, and Prest [HKKP22] propose SC-DAKE, a generic construction based on KEMs and a ring signature. BFGJS [BFG$^+$22] propose a similar construction called SPQR based on KEMs and a designated verifier signature. Dobson and Galbraith [DG22] lift X3DH to supersingular isogenies with their construction called SI-X3DH, which is however broken by the SIDH attack [CD23, MMP$^+$23, Rob23]. Collins, Huguenin-Dumittan, Nguyen, Rolin, and Vaudenay [CHDN$^+$24] propose K-Waay, based on adapting split KEMs [BFG$^+$20]. In this work, we focus on Signal's deployed PQXDH protocol.

```
$\underline{\mathcal{G}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})}$:
$(pk, sk) \leftarrow\!\!\$\ \mathsf{SIG.KGen}()$
$Q \leftarrow \emptyset$
$(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign}}(pk)$
return $[\![\mathsf{Vf}(pk, m^*, \sigma^* \wedge m^* \notin Q]\!]$

$\underline{\textsc{Sign}(m)}$:
$Q \leftarrow Q \cup \{m\}$
return $\mathsf{SIG.Sig}(sk, m)$
```

Figure 1: EUF-CMA security for a signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$.

2 Preliminaries

2.1 Notation

We refer to the first element of a list $\vec{l}$ with $\vec{l}[1]$. For $n \in \mathbb{N}$, $[n]$ denotes $\{1, \ldots, n\}$. We write a deterministic or randomized algorithm A with input x and output y as $y \leftarrow A(x)$ resp. $y \leftarrow\!\!\$\ A(x)$ (or also $A(x) \rightarrow y$ resp. $A(x)\ \$\!\!\rightarrow y$), for the latter with explicit randomness r we write $y \leftarrow B(x; r)$. We sample an element e uniformly at random from a set S with $e \leftarrow\!\!\$\ S$. We write $\Pr[\mathcal{G}(\mathcal{A})]$ for the probability that a game $\mathcal{G}$ involving an adversary $\mathcal{A}$ outputs true.

2.2 Signatures

Definition 2.1 (Signature scheme)**.** *A signature scheme* $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *with associated message space* $\mathcal{M}$ *is a triple of* PPT *algorithms:*

- $\mathsf{KGen}()\ \$\!\!\rightarrow (pk, sk)$*: The probabilistic key generation outputs a public-key/secret-key pair* (pk, sk)*.*
- $\mathsf{Sig}(sk, m)\ \$\!\!\rightarrow \sigma$*: The probabilistic signing takes as input a secret key sk and a message m and outputs a signature* σ*.*
- $\mathsf{Vf}(pk, m, \sigma) \rightarrow d$*: The deterministic verification takes as input a public key pk, a message m, and a signature* σ *and outputs a decision* $d \in \{0, 1\}$*. If* $d = 1$ *we say the signature is valid.*

A signature scheme is correct, if for all $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}()$ *and for all* $m \in \mathcal{M}$ *it holds that* $\Pr[\mathsf{Vf}(pk, m, \mathsf{Sig}(sk, m)) = 1]$.

Definition 2.2 (EUF-CMA Security of Signatures)**.** *Let* $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *be a signature scheme. We say that* SIG *is* $(t, \epsilon, q_{\mathrm{Sig}})$*–*$\mathsf{EUF\text{-}CMA}$*-secure, if for any adversary* $\mathcal{A}$ *against Game* $\mathcal{G}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})$ *defined in Figure 1 with running time at most t and making at most* q_{Sig} *queries to the* $\textsc{Sign}$ *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) := \Pr\Big[\mathcal{G}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})\Big] \leq \epsilon.$$

2.3 Hash functions

Definition 2.3 (Preimage resistance of hash functions)**.** *Let* $\mathsf{H}\colon \{0,1\}^* \rightarrow \{0,1\}^n$ *be a (hash) function. We say that* H *is* $(t, \epsilon_d, \mathcal{M})$*–preimage resistant for randomly sampled preimages (*$\mathsf{PR\text{-}d}$*-secure) and* (t, ϵ_r)*–preimage resistant for randomly sampled images (*$\mathsf{PR\text{-}r}$*-secure), respectively, if for any adversary* $\mathcal{A}$ *with running time at most t we have that*

$$\mathsf{Adv}^{\mathsf{PR\text{-}d}}_{\mathsf{H},\mathcal{M}}(\mathcal{A}) := \Pr\big[d = \mathsf{H}(m') \mid m \leftarrow\!\!\$\ \mathcal{M}, d \leftarrow \mathsf{H}(m), m' \leftarrow\!\!\$\ \mathcal{A}(d)\big] \leq \epsilon_d, \quad \textit{resp.}$$
$$\mathsf{Adv}^{\mathsf{PR\text{-}r}}_{\mathsf{H}}(\mathcal{A}) := \Pr\big[d = \mathsf{H}(m') \mid d \leftarrow\!\!\$\ \{0,1\}^n, m' \leftarrow\!\!\$\ \mathcal{A}(d)\big] \leq \epsilon_r.$$

```
$\mathcal{G}^{\mathsf{GDH}}_{(\mathbb{G},g,q)}(\mathcal{A})$:
$a, b \leftarrow\$ \mathbb{Z}_q$
$g^z \leftarrow\$ \mathcal{A}^{\mathrm{DDH}}((\mathbb{G}, g, q), a, b)$
return $[\![ g^z = g^{ab} ]\!]$
```

```
$\mathrm{DDH}(g^x, g^y, g^z)$:
return $[\![ \mathsf{CDH}(g^x, g^y) = g^z ]\!]$
```

```
$\mathcal{G}^{\mathsf{OW\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A})$:
$(pk, sk) \leftarrow\$ \mathsf{KGen}()$
$(ct^*, ss^*) \leftarrow\$ \mathsf{Enc}(pk)$
$ss' \leftarrow\$ \mathcal{A}^{\mathrm{DECAPS}}(pk, ct^*)$
return $[\![ ss' = ss^* ]\!]$
```

```
$\mathrm{DECAPS}(ct)$:
if $ct = ct^*$
    return $\perp$
else
    return $\mathsf{Dec}(sk, ct)$
```

Figure 2: Security game for the GapDH problem defined in Definition 2.6 (left) and for OW-CCA security of KEM = (KGen, Enc, Dec) defined in Definition 3.3 (right).

Definition 2.4 (Collision resistance of hash functions)**.** *Let* $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^n$ *be a (hash) function. We say that* H *is* (t, ϵ)*–collision-resistant (*CR*-secure), if for any adversary* $\mathcal{A}$ *with running time at most* t *we have that*

$$\mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H}}(\mathcal{A}) := \Pr\big[\mathsf{H}(m) = \mathsf{H}(m') \wedge m \neq m' \mid (m, m') \leftarrow\$ \mathcal{A}()\big] \leq \epsilon_{\mathsf{H}}.$$

Our analysis further uses that finding two input values that collide in the truncated (256-bit) output is hard. Prior work discusses the related (weak) near collision resistance [MvV97, PS14, JKN21], partial collision resistance [WY05, LT11, YCW14], and truncated collision resistance [JK18]. We model this as collision resistance of a hash function H truncated to the first 256 bits, which we denote by H_{256}.

SHA3. Bertoni, Daemen, Peters, and Van Assche [BDPV08] show that SHA3 is indifferentiable from a random oracle when the underlying permutation is modeled as a random permutation. Thus, finding a collision or preimage in SHA3 or $\mathsf{SHA3}_{256}$ is as hard as finding it in a random oracle yielding digests of the same length.

2.4 Diffie–Hellman Key Exchange

Definition 2.5 (Diffie–Hellman Key Exchange)**.** *A* Diffie–Hellman Key Exchange (DH) scheme *is a tuple of algorithms* (KGen, DH)*, defined as follows:*

- $\mathsf{KGen}() \xrightarrow{\$} (pk, sk)$*: This probabilistic algorithm returns a key pair* (pk, sk)
- $\mathsf{DH}(pk_A, sk_B) \to \mathsf{DH}_{AB}$*: On input a public key* pk_A *and a secret key* sk_B*, this deterministic algorithm returns the shared DH secret* DH_{AB}*.*

We say that a DH key exchange DH *is* correct *if, for every* $(pk_A, sk_A), (pk_B, sk_B) \leftarrow\$ \mathsf{KGen}()$*, it holds that*

$$\Pr[\mathsf{DH}(pk_A, sk_B) = \mathsf{DH}(pk_B, sk_A)] = 1.$$

For notational convenience we allow the arguments to be in arbitrary order, i.e., $\mathsf{DH}(pk_A, sk_B) = \mathsf{DH}(sk_A, pk_B)$. We write $\mathsf{CDH}(pk_A, pk_B)$ to refer to the DH shared secret DH_{AB} when we do not care which secret key is used.

We rely on the GapDH problem [OP01, CCD+17], i.e., that the computational DH problem is hard given a decisional DH oracle.

Definition 2.6 (GapDH problem)**.** *Let* $(\mathbb{G}, g, q)$ *be a group of order* q *with generator* g*. We say that the* $(t, \epsilon_{\mathsf{GDH}}, q_{\mathrm{DDH}})$-GapDH *problem holds in* $(\mathbb{G}, g, q)$ *if for any adversary* $\mathcal{A}$ *with running time at most* t *making at most* q_{DDH} *queries to its DDH oracle, we have that*

$$\mathsf{Adv}^{\mathsf{GDH}}_{(\mathbb{G},g,q)}(\mathcal{A}) := \Pr\Big[\mathcal{G}^{\mathsf{GDH}}_{(\mathbb{G},g,q)}(\mathcal{A})\Big] \leq \epsilon_{\mathsf{GDH}},$$

where $\mathcal{G}^{\mathsf{GDH}}_{(\mathbb{G},g,q)}(\mathcal{A})$ *is defined in Figure 2.*

3 Key Encapsulation Mechanisms and Binding Properties

We first recap the basic syntax, correctness, and security of key encapsulation mechanisms (KEMs), before discussing the novel KEM binding properties.

Definition 3.1 (Key Encapsulation Mechanisms)**.** *A* key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of the following three algorithms:*

- $\mathsf{KGen}() \,{\scriptstyle\$}{\rightarrow}\, (pk, sk)$*: The probabilistic* key generation *with randomness space* $\mathcal{R}_{\mathsf{KEM.KGen}}$ *outputs a public-key/secret-key pair with* $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$*.*
- $\mathsf{Enc}(pk) \,{\scriptstyle\$}{\rightarrow}\, (ct, ss)$*: The probabilistic* encapsulation *algorithm with randomness space* $\mathcal{R}_{\mathsf{KEM.Enc}}$ *takes as input a public key* $pk \in \mathcal{PK}$ *and outputs a ciphertext* $ct \in \mathcal{C}$ *and the therein encapsulated shared secret* $ss \in \mathcal{SS}$*.*
- $\mathsf{Dec}(sk, ct) \rightarrow ss'$*: The deterministic* decapsulation *algorithm takes as input a ciphertext* $ct \in \mathcal{C}$ *and secret key* sk *and outputs* $ss' \in \mathcal{SS} \cup \{\bot\}$*, where* $\bot$ *indicates an error.*

We say that a KEM $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* δ-correct *if, for every key pair* $(pk, sk) \leftarrow_{\$} \mathsf{KGen}()$*, and every encapsulation* $(ct, ss) \leftarrow_{\$} \mathsf{Enc}(pk)$*, we have*

$$\Pr\big[ss' \neq ss \mid ss' \leftarrow \mathsf{Dec}(sk, ct)\big] \leq \delta.$$

In our analysis, we sometimes need to rule out that honestly generated KEM public keys collide. We capture that probability in the following. For Kyber [SAB+] and ML-KEM [NIS23], such public-key collisions boil down to random 256-bit seeds colliding under SHA3-512.

Definition 3.2 (KEM public-key collision probability)**.** *We define the* public-key collision probability *of a KEM via a function* $\gamma_{\mathsf{coll}} \colon \mathbb{N} \rightarrow [0, 1]$*, letting* $\gamma_{\mathsf{coll}}(n)$ *denote the probability that two among* n *honestly generated public keys collide:*

$$\gamma_{\mathsf{coll}}(n) := \Pr\Big[pk_i = pk_j \wedge i \neq j \mid (pk_i, sk_i) \leftarrow_{\$} \mathsf{KGen}() \textit{ for } i \in [1, n]\Big].$$

Our analysis requires KEM *one-way security under chosen-ciphertext attacks* (OW-CCA), which is implied by standard IND-CCA security.

Definition 3.3 (OW-CCA Security of KEMs)**.** *Let* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a KEM. We say that* KEM *is* $(t, \epsilon, q_{\mathrm{Dec}})$*–*OW-CCA*-secure, if for any adversary* $\mathcal{A}$ *against Game* $\mathcal{G}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A})$ *defined in Figure 2 with running time at most* t *and making at most* q_{Dec} *queries to the* Decaps *oracle, we have that*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{OW\text{-}CCA}}(\mathcal{A}) := \Pr\Big[\mathcal{G}_{\mathsf{KEM}}^{\mathsf{OW\text{-}CCA}}(\mathcal{A})\Big] \leq \epsilon.$$

3.1 Binding Properties

Due to the way PQXDH includes (only) the KEM shared secret in its key derivation (but not the KEM public key or ciphertext), our security analysis relies on a novel binding property of the KEM scheme. In a nutshell, we ask that it is hard to find two distinct ciphertexts or (honestly generated) public keys so that the corresponding decapsulations output the same shared secrets, even when given full control over the ciphertexts and knowing corresponding secret keys and the *randomness* used to generate these keys. The latter is a result of capturing the "maximum-exposure" security of PQXDH, allowing an adversary to reveal a session's random coins (cf. Section 4). The PQXDH setting, in which an adversary interacts with many KEM keys, also motivates our notion being *multi-user*.

```
$\mathcal{G}^{LEAK^+\text{-BIND-}SS\text{-}\{CT,PK\}}_{\mathsf{KEM},n}(\mathcal{A})$:
for $k \in [n]$
    $r_k \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{KEM.KGen}}$
    $(pk_k, sk_k) \leftarrow \mathsf{KEM.KGen}(; r_k)$
$(i, ct_i, j, ct_j) \leftarrow\!\!\$\ \mathcal{A}((pk_k, sk_k, r_k)_{k\in[n]})$
    // $\mathcal{A}$ gets all key pairs *and* KGen randomness
$ss_i \leftarrow \mathsf{KEM.Dec}(sk_i, ct_i)$
$ss_j \leftarrow \mathsf{KEM.Dec}(sk_j, ct_j)$
if $ss_i = \bot \vee ss_j = \bot$: return false
return $[\![ ss_i = ss_j \wedge (ct_i \neq ct_j \vee pk_i \neq pk_j) ]\!]$
```

```
$\mathcal{G}^{X\text{-BIND-}SS\text{-}\{CT,PK\}}_{\mathsf{KEM}}(\mathcal{A})$:
if $X = LEAK$:
    $(pk_0, sk_0) \leftarrow\!\!\$\ \mathsf{KEM.KGen}()$
    $(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{KEM.KGen}()$
    $(ct_0, ct_1) \leftarrow\!\!\$\ \mathcal{A}(pk_0, sk_0, pk_1, sk_1)$
if $X = MAL$:
    $(pk_0, sk_0, pk_1, sk_1, ct_0, ct_1) \leftarrow\!\!\$\ \mathcal{A}()$
$ss_0 \leftarrow \mathsf{KEM.Dec}(sk_0, ct_0)$
$ss_1 \leftarrow \mathsf{KEM.Dec}(sk_1, ct_1)$
if $ss_0 = \bot \vee ss_1 = \bot$: return false
return $[\![ ss_0 = ss_1 \wedge (ct_0 \neq ct_1 \vee pk_0 \neq pk_1) ]\!]$
```

Figure 3: Security games for our (multi-user) $LEAK^+$-BIND-SS-$\{CT, PK\}$ notion (left) and the $LEAK$-BIND-SS-$\{CT, PK\}$ and MAL-BIND-SS-$\{CT, PK\}$ notion from [CDM23] (right). In all games, the adversary's goal is to produce colliding shared secrets ss under distinct ciphertexts ct or public keys pk.

Aligning with language of Cremers, Dax, and Medinger (CDM) [CDM23] from their concurrent work systematizing *binding* notions for KEMs, our notion asks that the KEM shared secret binds both public key and ciphertext used to produce it, under leakage of key generation randomness (and hence secret keys), which is a novel variant we denote as $LEAK^+$-BIND-SS-$\{CT, PK\}$. In general, their notion X-BIND-P-Q captures that the components in set $P \in \{\{ss\}, \{ct\}, \{ss, ct\}\}$ bind the components in set $Q \in \{\{pk\}, \{ss\}, \{ct\}\}$, where the KEM keys are chosen by the adversary ($X = MAL$), honestly generated and leaked to the adversary ($X = LEAK$), or honestly generated without leakage of secrets ($X = HON$). We reproduce their $LEAK$ and MAL notions for ss simultaneously binding pk and ct in our syntax[3] on the right-hand side of Figure 3.

In the following, we formalize our new $LEAK^+$-BIND-SS-$\{CT, PK\}$ binding property and relate it to the notions in the CDM framework [CDM23]. In particular, we show in Section 3.3 below that both Kyber and the current NIST draft standard of ML-KEM satisfy the $LEAK^+$-BIND-SS-$\{CT, PK\}$ property, while the notion from [CDM23] that implies ours, MAL-BIND-SS-$\{CT, PK\}$, is not satisfied by ML-KEM [Sch24].[4] While Signal's current implementation of PQXDH uses Kyber [SAB+], it is already prepared[5] to transition to the NIST standard ML-KEM [NIS23].

Definition 3.4 ($LEAK^+$-BIND-SS-$\{CT, PK\}$)**.** *Let* KEM = (KGen, Enc, Dec) *be a KEM and* $n \geq 2$. *We say that* KEM *is* (t, ϵ, n)-$LEAK^+$-BIND-SS-$\{CT, PK\}$-secure*, if for any adversary* $\mathcal{A}$ *against Game* $\mathcal{G}^{LEAK^+\text{-BIND-}SS\text{-}\{CT,PK\}}_{\mathsf{KEM},n}(\mathcal{A})$ *defined in Figure 3 with running time at most* t*, we have that*

$$\mathsf{Adv}^{LEAK^+\text{-BIND-}SS\text{-}\{CT,PK\}}_{\mathsf{KEM},n}(\mathcal{A}) := \Pr\Big[\mathcal{G}^{LEAK^+\text{-BIND-}SS\text{-}\{CT,PK\}}_{\mathsf{KEM},n}(\mathcal{A})\Big] \leq \epsilon.$$

Figure 4 visualizes the relations between $LEAK^+$-BIND-SS-$\{CT, PK\}$ and the corresponding notions from [CDM23]: The MAL variant is strictly stronger than the $LEAK^+$ variant, which in turn is strictly stronger than the $LEAK$ variant. Notably, ML-KEM separates the MAL and our $LEAK^+$ variants; we show the other implications and separations below, incl. relating our multi-user notion to the 2-user case.

[3] Note that in the syntax of [CDM23] the KEMs shared secret ss is simply called a "key", denoted k. Accordingly, in their notation, $P \in \{\{\mathsf{k}\}, \{ct\}, \{\mathsf{k}, ct\}\}$ and $Q \in \{\{pk\}, \{\mathsf{k}\}, \{ct\}\}$. We will stick to our syntax and hence, e.g., write MAL-BIND-SS-CT corresponding to their notion MAL-BIND-K-CT.

[4] CDM [CDM23, Appendices B and E] conjecture that Kyber is MAL-BIND-SS-$\{CT, PK\}$–secure as it hashes both pk and ct into the key derivation. Their generic argument [CDM23, Thm. E.1] requires including both via an injective function, a hash function as used in Kyber however is clearly not injective. We leave it as an open question whether Kyber achieves MAL-BIND-SS-$\{CT, PK\}$ security; for us, the $LEAK^+$ variant suffices.

[5] https://github.com/signalapp/libsignal/commit/0670f0d

$$\begin{tikzcd}
\mathit{MAL}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT, PK\} \arrow[r, shift left, "\text{Thm. 3.5}"] & \mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT, PK\} \text{ for n users} \arrow[l, shift left, "/" marking, "\text{Thm. 3.9 (ML-KEM)}"] \arrow[d, shift left, "\text{Thm. 3.7}"] \\
\mathit{LEAK}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT, PK\} \arrow[r, shift right, "/" marking, "\text{Thm. 3.8}"'] & \mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT, PK\} \text{ for 2 users} \arrow[l, shift right, "\text{Thm. 3.6}"'] \arrow[u, dashed, shift left, "\text{Thm. 3.7}"]
\end{tikzcd}$$

Figure 4: Relations between our LEAK^{+}-BIND-SS-$\{CT, PK\}$ binding notion and the corresponding MAL, LEAK notions from [CDM23] for KEMs. Solid arrows indicate implications, dashed ones loose implications, and crossed-out ones separations. Annotations indicate the respective theorems.

3.2 Relations of Binding Properties

First, we show the implication from MAL to LEAK^{+} (Theorem 3.5) and from LEAK^{+} to LEAK (Theorem 3.6). Second, we show that our notion is equivalent for n and 2 users, up to a factor of n^2 (Theorem 3.7). Third, we give a separation between LEAK and LEAK^{+} (Theorem 3.8). Lastly, we show that ML-KEM and Kyber are LEAK^{+} (Theorem 3.9 and Theorem 3.10), while ML-KEM is known to not be MAL [Sch24], giving us a separation for LEAK^{+} and MAL.

Note that for the following theorems we assume KEMs with a public key space $|\mathcal{PK}| \geq 2$.

Theorem 3.5 (MAL-BIND-SS-$\{CT, PK\} \implies \mathit{LEAK}^{+}$-BIND-$\mathit{SS}$-$\{CT, PK\}$)**.** *A MAL-BIND-SS-$\{CT, PK\}$–secure KEM* KEM *is also LEAK^{+}-BIND-SS-$\{CT, PK\}$–secure. Concretely, for any LEAK^{+}-BIND-SS-$\{CT, PK\}$ adversary $\mathcal{A}$ against* KEM *there exist an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT,PK\}}_{\mathsf{KEM},n}(\mathcal{A}) \leq \mathsf{Adv}^{\mathit{MAL}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT,PK\}}_{\mathsf{KEM}}(\mathcal{B}).$$

Proof. To win in the LEAK^{+}-BIND-SS-$\{CT, PK\}$ game, $\mathcal{A}$ must produce ciphertexts ct_i, ct_j such that $ss_i = ss_j$ and one (or both) of $pk_i \neq pk_j$ and $ct_i \neq ct_j$ hold.

We let $\mathcal{B}$ truthfully simulate $\mathcal{G}^{\mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT,PK\}}_{\mathsf{KEM},n}(\mathcal{A})$ for $\mathcal{A}$, i.e., honestly computing keys pk_k for $k \in [n]$, obtaining i, ct_i, j, ct_j from $\mathcal{A}$. Then, $\mathcal{B}$ outputs $(pk_i, sk_i, pk_j, sk_j, ct_i, ct_j)$. If $\mathcal{A}$ wins, we have that $ss_i = ss_j$ while $pk_i \neq pk_j$ or $ct_i \neq ct_j$, so $\mathcal{B}$ also wins in the MAL-BIND-SS-$\{CT, PK\}$ game. □

Theorem 3.6 (LEAK^{+}-BIND-SS-$\{CT, PK\} \implies \mathit{LEAK}$-BIND-$\mathit{SS}$-$\{CT, PK\}$)**.** *A LEAK^{+}-BIND-SS-$\{CT, PK\}$–secure KEM* KEM *is also LEAK-BIND-SS-$\{CT, PK\}$–secure. Concretely, for any LEAK-BIND-SS-$\{CT, PK\}$ adversary $\mathcal{A}$ against* KEM *there exist an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathit{LEAK}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT,PK\}}_{\mathsf{KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{CT,PK\}}_{\mathsf{KEM},2}(\mathcal{B}).$$

Proof. We let $\mathcal{B}$ start $\mathcal{A}$ on its own inputs except the key generation randomness, thereby truthfully simulating the LEAK-BIND-SS-$\{CT, PK\}$ game for $\mathcal{A}$. When $\mathcal{A}$ terminates, $\mathcal{B}$ returns the output of $\mathcal{A}$. If $\mathcal{A}$ wins, then so does $\mathcal{B}$. □

Theorem 3.7 (LEAK^{+}-BIND-SS-$\{CT, PK\}$: n users $\iff$ 2 users)**.** *Let $n \geq 2$. A (t, ϵ, n)-LEAK^{+}-BIND-SS-$\{CT, PK\}$-secure KEM* KEM *is also $(t, \epsilon, 2)$-LEAK^{+}-BIND-SS-$\{CT, PK\}$-secure and a $(t, \epsilon, 2)$-LEAK^{+}-BIND-SS-$\{CT, PK\}$-secure KEM* KEM *is also $(t', n^2 \cdot \epsilon, n)$-$\mathit{LEAK}^{+}$-BIND-$\mathit{SS}$-$\{CT, PK\}$-secure* *(for $t' \approx t$). Concretely, for any $LEAK^{+}$-BIND-SS-$\{CT, PK\}$ adversary $\mathcal{A}$ against* KEM *there exist an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathsf{KEM},2}^{LEAK^{+}\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{KEM},n}^{LEAK^{+}\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{A}) \quad \textit{and}$$
$$\mathsf{Adv}_{\mathsf{KEM},n}^{LEAK^{+}\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{A}) \leq n^2 \cdot \mathsf{Adv}_{\mathsf{KEM},2}^{LEAK^{+}\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{B}).$$

Proof. The "$\Longrightarrow$" direction straightforwardly holds by letting $\mathcal{A}$ ignore the remaining $n-2$ keys.

The "$\Longleftarrow$" direction holds via a guessing argument: Let $\mathcal{B}$ guess two distinct key indices $i, j \in [n]$, embed the two keys obtained in its game in these positions, and sample the remaining $n-2$ keys itself. If $\mathcal{A}$ uses i (and possibly j) for its attack, $\mathcal{B}$ is successful if $\mathcal{A}$ is. The chance of $\mathcal{B}$ guessing correctly is $\frac{1}{n^2}$, establishing the claim. □

Theorem 3.8 ($LEAK$-BIND-SS-$\{CT, PK\} \not\Longrightarrow LEAK^{+}$-BIND-$SS$-$\{CT, PK\}$)**.** *Assuming a $LEAK$-BIND-SS-$\{CT, PK\}$–secure KEM* KEM′ *and a preimage-resistant (*PR-d*) hash function* H *(when sampling from $\{0,1\}^{256}$), there exists a KEM* KEM *which is $LEAK$-BIND-SS-$\{CT, PK\}$–secure but not $LEAK^{+}$-BIND-SS-$\{CT, PK\}$–secure. Concretely, for any $LEAK$-BIND-SS-$\{CT, PK\}$ adversary $\mathcal{A}$ against* KEM′ *there exist adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ such that*

$$\mathsf{Adv}_{\mathsf{KEM},2}^{LEAK^{+}\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{B}_1) = 1 \ \textit{and}$$
$$\mathsf{Adv}_{\mathsf{KEM}}^{LEAK\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{H}}^{\mathsf{PR\text{-}d}}(\mathcal{B}_2) + \mathsf{Adv}_{\mathsf{KEM}'}^{LEAK\text{-}\mathsf{BIND}\text{-}SS\text{-}\{CT,PK\}}(\mathcal{B}_3).$$

Proof. Let $\mathsf{KEM}' = (\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}')$ with shared secret space $\mathcal{SS}'$ and randomness space $\mathcal{R}_{\mathsf{KGen}'}$ for key generation be $LEAK$-BIND-SS-$\{CT, PK\}$–secure, and consider the following modification KEM with shared secret space $\mathcal{SS}' \cup \{ss^*\}$ where ss^* denotes a distinguished shared secret with $ss^* \notin \mathcal{SS}'$ and randomness space $\{0,1\}^{256} \times \mathcal{R}_{\mathsf{KGen}'}$ for key generation.

```
KGen(; $r$):
1 $(x, r_{KG}) \leftarrow r$
2 $y \leftarrow \mathsf{H}(x)$
3 $(pk', sk') \leftarrow \mathsf{KGen}'(; r_{KG})$
4 return $(pk', (sk', y))$
```

```
Enc($pk$):
5 return $(\mathsf{Enc}'(pk), \perp)$
```

```
Dec($sk, pk, ct$):
6 $(sk', y) \leftarrow sk$
7 $(ct', z) \leftarrow ct$
8 if $y = \mathsf{H}(z)$
9   return $ss^*$
10 return $\mathsf{Dec}'(sk', pk, ct')$
```

KEM is not $LEAK^{+}$-BIND-SS-$\{CT, PK\}$-secure: An adversary $\mathcal{B}_1$ gets $(pk_1, sk_1, r_1, pk_2, sk_2, r_2)$ as input, parses the randomness as $(x_1, r_{KG,1}) \leftarrow r_1$ and $(x_2, r_{KG,2}) \leftarrow r_2$, and outputs the two ciphertexts $(0, x_1), (1, x_2)$ (omitting the indices for the two users). Both decapsulations trigger the special case. Hence, the shared secrets collide, while the ciphertexts differ (at least) in their first component and $\mathcal{B}_1$ always wins.

KEM is $LEAK$-BIND-SS-$\{CT, PK\}$-secure: If $\mathcal{A}$ triggers the special condition in the decapsulation, then it has found a preimage under H (for randomly sampled 256-bit preimages), breaking preimage resistance. Otherwise, the adversary cannot take advantage of the extra components in the secret key and ciphertext. Hence, breaking $LEAK$-BIND-SS-$\{CT, PK\}$ of KEM implies breaking $LEAK$-BIND-SS-$\{CT, PK\}$ of KEM′, which we have excluded by assumption. □

```
KGen():
1  $z \leftarrow_\$ \{0,1\}^{256}$
2  $(pk, sk') \leftarrow_\$ \mathsf{PKE.KGen}()$
3  $sk \leftarrow (sk', pk, \mathsf{H}(pk), z)$
4  return $(pk, sk)$

Enc(pk):
5  $m \leftarrow_\$ \{0,1\}^{256}$
6  $\boxed{m \leftarrow \mathsf{H}(m)}$
7  $(ss', r) \leftarrow \mathsf{G}(m \| \mathsf{H}(pk))$
8  $ct \leftarrow \mathsf{PKE.Enc}(pk, m; r)$
9  $\boxed{ss \leftarrow \mathsf{KDF}(ss' \| \mathsf{H}(ct))}$
   $\dbox{ss \leftarrow ss'}$
10 return $(ct, ss)$

Dec(sk, ct):
11 $(sk', pk, h, z) \leftarrow sk$
12 $m' \leftarrow \mathsf{PKE.Dec}(sk', ct)$
13 $(ss', r') \leftarrow \mathsf{G}(m' \| h)$
14 $ct \leftarrow \mathsf{PKE.Enc}(pk, m'; r')$
15 if $ct = ct'$:
16     $\boxed{ss \leftarrow \mathsf{KDF}(ss' \| \mathsf{H}(ct))}$
       $\dbox{ss \leftarrow ss'}$
17 else
18     $\boxed{ss \leftarrow \mathsf{KDF}(z \| \mathsf{H}(ct))}$
       $\dbox{ss \leftarrow \mathsf{J}(z \| ct)}$
19 return $ss$
```

Figure 5: Algorithmic description of ML-KEM and Kyber using functions H, G, and J resp. KDF and a public key encryption scheme PKE. $\boxed{\text{Solid boxes}}$ are exclusive to Kyber, $\dbox{\text{dashed boxes}}$ are exclusive to ML-KEM.

3.3 Kyber and ML-KEM

Figure 5 gives an algorithmic description of Kyber [SAB+] and its corresponding current draft NIST standard ML-KEM [NIS23]. On a high level, the scheme uses a public key encryption scheme PKE in an FO transform [FO99], as well as three functions H, G, and J (ML-KEM) resp. KDF (Kyber), instantiated as SHA3-256, SHA3-512, and SHAKE256 with 256 bits output, respectively. Our interest being in the binding properties, we focus here on the internals of decapsulation. The decapsulation algorithm first decrypts the PKE ciphertext ct to m'. Then, it hashes m' and h (the hashed public key, stored in the secret key) under G onto ss', r'. It uses r' as randomness for re-encrypting m'. If the re-encrypted ciphertext matches ct, Kyber outputs $\mathsf{KDF}(ss' \| \mathsf{H}(ct))$ as shared secret while ML-KEM outputs ss' directly. Otherwise, perform an implicit rejection, i.e., compute the shared secret as $\mathsf{KDF}(z \| \mathsf{H}(ct))$ for Kyber resp. $\mathsf{J}(z \| ct)$ for ML-KEM, where z is a secret random 256-bit string, which is part of the secret key.

In the following, we show that ML-KEM and Kyber satisfy the LEAK^+-BIND-SS-$\{\mathit{CT}, \mathit{PK}\}$ binding property, assuming collision resistance of all involved hash functions, (only for ML-KEM) random-oracle properties of H and J, and (only for Kyber) preimage resistance of G_{256} for randomly sampled images.

Theorem 3.9 (ML-KEM is LEAK^+-BIND-SS-$\{\mathit{CT}, \mathit{PK}\}$–secure)**.** *Assuming* H *is* $(t, \epsilon_{\mathsf{H}})$*–collision-resistant,* G_{256} *is* $(t, \epsilon_{\mathsf{G}})$*–collision-resistant,* J *is* $(t, \epsilon_{\mathsf{J}})$*–collision-resistant, and modeling* H, J *as independent random oracles,* ML-KEM *is* (t', ϵ, n)*–*LEAK^+-BIND-SS-$\{\mathit{CT}, \mathit{PK}\}$*–secure for* $t' \approx t$ *and*

$$\epsilon = \frac{n^2}{2^{256}} + 2\epsilon_{\mathsf{J}} + \frac{q_{\mathrm{RO}}^2}{2^{256}} + \epsilon_{\mathsf{H}} + 2\epsilon_{\mathsf{G}},$$

where q_{RO} *is the number of random oracle queries made by the adversary.*

Proof. We analyze the probability of $\mathcal{A}$ winning over a series of game hops.

Game 0. We start with the original binding game $\mathcal{G}_{\text{ML-KEM},n}^{\mathit{LEAK}^+\text{-BIND-}\mathit{SS}\text{-}\{\mathit{CT},\mathit{PK}\}}(\mathcal{A})$:

$$\mathsf{Adv}_{\text{ML-KEM},n}^{\mathit{LEAK}^+\text{-BIND-}\mathit{SS}\text{-}\{\mathit{CT},\mathit{PK}\}}(\mathcal{A}) = \mathsf{Adv}_{\text{ML-KEM}}^{\mathcal{G}_0}(\mathcal{A}).$$

We distinguish several cases for the adversary to win, depending on how the adversary fulfills the winning condition and if the decapsulation accepts (line 15 in Figure 5 evaluates to true) or implicitly rejects (i.e., decapsulation branches into line 17). We denote line 15 being taken with A_x and line 17 with R_x, where $x \in \{i, j\}$ are the key indices in the binding game. We index any intermediate values from decapsulation of ct_i and ct_j with i and j, respectively. We assume a successful adversary, i.e., $ss_i = ss_j$. Note that R_x and A_x are mutually exclusive. The cases are:

A. Both decapsulations reject and the ciphertexts are for two distinct public keys, i.e., $R_i \wedge R_j \wedge pk_i \neq pk_j$.

B. Both decapsulations reject and the ciphertexts are distinct, i.e., $R_i \wedge R_j \wedge ct_i \neq ct_j$.

C. One decapsulation rejects, i.e., wlog. $R_i \wedge A_j$.

D. Both decapsulations accept and the ciphertexts are for two distinct public keys, i.e., $A_i \wedge A_j \wedge pk_i \neq pk_j$.

E. Both decapsulations accept and the ciphertexts are distinct, i.e., $A_i \wedge A_j \wedge pk_i = pk_j \wedge ct_i \neq ct_j$.

We treat these cases as events in $\mathcal{G}_0$ and indicate the occurrence of event X by $\mathcal{G}_0[X]$. By the union bound we get:

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \sum_{X \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D},\mathrm{E}\}} \mathsf{Adv}^{\mathcal{G}_0[X]}_{\mathsf{ML\text{-}KEM}}(\mathcal{A})$$

Case A (Both decapsulations reject, distinct public keys: $R_i \wedge R_j \wedge pk_i \neq pk_j$).

Here, we bound the probability of the adversary producing ciphertexts that both get implicitly rejected during decapsulation and decapsulate to the same shared secret under distinct public keys.

Game A.0. This is the game conditioned on $R_i \wedge R_j \wedge pk_i \neq pk_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[R_i \wedge R_j \wedge pk_i \neq pk_j]}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}).$$

Game A.1 (Colliding z values). We let $\mathcal{G}_{\mathrm{A.0}}$ return false if there is a collision among the z values in the n secret keys. Since the z values are 256-bit strings sampled uniformly at random, by the birthday bound we get:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.1}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) + \frac{n^2}{2^{256}}.$$

Game A.2 (Collision in J). Now, we have established $z_i \| ct_i \neq z_j \| ct_j$ (since $z_i \neq z_j$), yet $\mathsf{J}(z_i \| ct_i) = \mathsf{J}(z_j \| ct_j)$. We build a reduction $\mathcal{B}_1$ that outputs $z_i \| ct_i, z_j \| ct_j$, breaking $(t, \epsilon_{\mathsf{J}})$–collision resistance of J:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.1}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \epsilon_{\mathsf{J}}.$$

Case B (Both decapsulations reject, distinct ciphertexts: $R_i \wedge R_j \wedge ct_i \neq ct_j$).

Here, we bound the probability of the adversary producing two distinct ciphertexts that both get implicitly rejected during decapsulation and decapsulate to the same shared secret.

Game B.0. This is the game conditioned on $R_i \wedge R_j \wedge ct_i \neq ct_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{B.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[R_i \wedge R_j \wedge ct_i \neq ct_j]}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}).$$

Since $ct_i \neq ct_j$, we also have $z_i \| ct_i \neq z_j \| ct_j$, yet $\mathsf{J}(z_i \| ct_i) = \mathsf{J}(z_j \| ct_j)$. We build a reduction $\mathcal{B}_2$ that outputs $z_i \| ct_i, z_j \| ct_j$, breaking $(t, \epsilon_{\mathsf{J}})$–collision resistance of J:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{B.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \epsilon_{\mathsf{J}}.$$

Case C (One decapsulation rejects: $R_i \wedge A_j$).

Here, we bound the probability of the adversary producing exactly one ciphertext that gets implicitly rejected during decapsulation while both ciphertexts decapsulate to the same shared secret.

Game C.0. This is the game conditioned on $R_i \wedge A_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{C.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[R_i \wedge A_j]}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}).$$

For $\mathcal{A}$ to win, it needs to create a collision $\mathsf{J}(z_i \| ct_i) = ss_i = ss_j = \mathsf{G}_{256}(m'_j \| h_j)$ between J and G. Assuming both J and G beave like (independent) random oracles, the probability of finding such a collision with q_{RO} many random oracle queries is upper bounded by:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{C.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \frac{q_{\mathrm{RO}}^2}{2^{256}}.$$

Case D (Both decapsulations accept, distinct public keys: $A_i \wedge A_j \wedge pk_i \neq pk_j$).

Here, we bound the probability of the adversary producing ciphertexts that both get accepted during decapsulation and decapsulate to the same shared secret under distinct public keys.

Game D.0. This is the game conditioned on $A_i \wedge A_j \wedge pk_i \neq pk_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{D.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[A_i \wedge A_j \wedge pk_i \neq pk_j]}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}).$$

Since we know that $pk_i \neq pk_j$ and $\mathsf{G}_{256}(m_i \| \mathsf{H}(pk_i)) = ss_i = ss_j = \mathsf{G}_{256}(m_j \| \mathsf{H}(pk_j))$, it must be that $\mathcal{A}$ provides us with a collision either in H or in G_{256}. Once more, we can distinguish two cases:

1. The first case is $m_i \| \mathsf{H}(pk_i) = m_j \| \mathsf{H}(pk_j)$. Then we can build a reduction $\mathcal{B}_3$ that outputs pk_i, pk_j as collision in H, breaking $(t, \epsilon_{\mathsf{H}})$-collision resistance of H.
2. Otherwise, $m_i \| \mathsf{H}(pk_i) \neq m_j \| \mathsf{H}(pk_j)$, and we can build reduction $\mathcal{B}_4$ that outputs $m_i \| \mathsf{H}(pk_i), m_j \| \mathsf{H}(pk_j)$ as collision in G_{256}. Hence, $\mathcal{B}_2$ breaks the $(t, \epsilon_{\mathsf{G}})$-collision resistance of G_{256}.

Jointly,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{D.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{G}}.$$

Case E (Both decapsulations accept, distinct ciphertexts: $A_i \wedge A_j \wedge pk_i = pk_j \wedge ct_i \neq ct_j$).

Here, we bound the probability of the adversary producing ciphertexts that both get accepted during decapsulation and decapsulate to the same shared secret under the same public key.

Game E.0. This is the game conditioned on $A_i \wedge A_j \wedge pk_i = pk_j \wedge ct_i \neq ct_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[A_i \wedge A_j \wedge pk_i = pk_j \wedge ct_i \neq ct_j]}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}).$$

We know that $pk_i = pk_j$ (and hence $h_i = h_j$) for $\mathsf{G}_{256}(m_i \| h_i) = ss_i = ss_j = \mathsf{G}_{256}(m_j \| h_j)$. Note that we must have $m_i \neq m_j$: Otherwise, the re-encryption step in both decapsulations would result in identical ciphertexts $ct_i = ct_j$, which contradicts the condition of event E.

Now, we can build a reduction $\mathcal{B}_5$ that outputs $m_i \| H(pk_i), m_j \| H(pk_j)$ as collision in G_{256}. Since G collides in its first output, which is 256 bits, $\mathcal{B}_3$ breaks the $(t, \epsilon_{\mathsf{G}})$-collision resistance of G_{256}.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.0}}}_{\mathsf{ML\text{-}KEM}}(\mathcal{A}) \leq \epsilon_{\mathsf{G}}.$$

Collecting the bounds yields the claim. □

For the Kyber proof, we do not require any collisions *across* hash functions, allowing us to avoid relying on the random oracle. Instead, we additionally require G_{256} to be $(t, \epsilon'_{\mathsf{G}})$–preimage-resistant for randomly sampled images.

Theorem 3.10 (Kyber is LEAK^{+}-BIND-SS-$\{\mathit{CT}, \mathit{PK}\}$–secure)**.** *Assuming* H *is* $(t, \epsilon_{\mathsf{H}})$*–collision-resistant,* G_{256} *is* $(t, \epsilon_{\mathsf{G}})$*–collision-resistant and* $(t, \epsilon'_{\mathsf{G}})$*–preimage-resistant (*$\mathsf{PR\text{-}r}$*, i.e., when sampling from the range), and* KDF *is* $(t, \epsilon_{\mathsf{KDF}})$*–collision-resistant,* Kyber *is* (t', ϵ, n)*–*LEAK^{+}-BIND-SS-$\{\mathit{CT}, \mathit{PK}\}$*–secure for* $t' \approx t$ *and*

$$\begin{aligned}\epsilon &= \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{KDF}} + \frac{n^2}{2^{256}} + \epsilon_{\mathsf{KDF}} + \epsilon_{\mathsf{KDF}} + n\epsilon'_{\mathsf{G}} + \epsilon_{\mathsf{KDF}} + \epsilon_{\mathsf{G}} + \epsilon_{\mathsf{H}} \\ &= 2\epsilon_{\mathsf{H}} + \epsilon_{\mathsf{G}} + n\epsilon'_{\mathsf{G}} + 4\epsilon_{\mathsf{KDF}} + \frac{n^2}{2^{256}}.\end{aligned}$$

Proof. We analyze the probability of $\mathcal{A}$ winning over a series of game hops.

Game 0. We start with the original binding game $\mathcal{G}_{\mathsf{Kyber},n}^{\mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{\mathit{CT},\mathit{PK}\}}(\mathcal{A})$:

$$\mathsf{Adv}_{\mathsf{Kyber},n}^{\mathit{LEAK}^{+}\text{-}\mathsf{BIND}\text{-}\mathit{SS}\text{-}\{\mathit{CT},\mathit{PK}\}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{Kyber}}^{\mathcal{G}_0}(\mathcal{A}).$$

We distinguish several cases for the adversary to win, depending on how the adversary fulfills the winning condition and if the decapsulation accepts (line 15 in Figure 5 evaluates to true) or implicitly rejects (i.e., decapsulation branches into line 17). We denote line 15 being taken with A_x and line 17 with R_x, where $x \in \{i, j\}$ are the key indices in the binding game. We index any intermediate values from decapsulation of ct_i and ct_j with i and j, respectively. We assume a successful adversary, i.e., $ss_i = ss_j$. Note that R_x and A_x are mutually exclusive. The cases are:

A. Both ciphertexts are distinct, i.e., $ct_i \neq ct_j$.

B. Both decapsulations reject and the ciphertexts are for two distinct public keys, i.e., $R_i \wedge R_j \wedge pk_i \neq pk_j$.

C. One decapsulation rejects and the ciphertexts are for two distinct public keys, i.e., wlog. $R_i \wedge A_j \wedge pk_i \neq pk_j$.

D. Both decapsulations accept and the ciphertexts are for two distinct public keys, i.e., $A_i \wedge A_j \wedge pk_i \neq pk_j$.

We treat these cases as events in $\mathcal{G}_0$ and indicate the occurrence of event X by $\mathcal{G}_0[X]$. By the union bound we get:

$$\mathsf{Adv}_{\mathsf{Kyber}}^{\mathcal{G}_0}(\mathcal{A}) \leq \sum_{X \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}} \mathsf{Adv}_{\mathsf{Kyber}}^{\mathcal{G}_0[X]}(\mathcal{A})$$

Case A (distinct ciphertexts: $ct_i \neq ct_j$).

Here, we bound the probability of the adversary producing two distinct ciphertexts that decapsulate to the same shared secret.

Game A.0. This is the game conditioned on $ct_i \neq ct_j$ being satisfied.

$$\mathsf{Adv}_{\mathsf{Kyber}}^{\mathcal{G}_{\mathrm{A}.0}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{Kyber}}^{\mathcal{G}_0[ct_i \neq ct_j]}(\mathcal{A}).$$

We know that $\mathsf{KDF}(x_i \| \mathsf{H}(ct_i)) = ss_i = ss_j = \mathsf{KDF}(x_j \| \mathsf{H}(ct_j))$ for x_i being either ss'_i (in case of an accepting decapsulation) or z_i (in case of an implicitly rejecting decapsulation), and x_j likewise. Since we know that $ct_i \neq ct_j$, we can now either build a reduction $\mathcal{B}_1$ that outputs ct_i, ct_j, breaking $(t, \epsilon_{\mathsf{H}})$–collision-resistance of H, or a reduction $\mathcal{B}_2$ that outputs $x_i \| \mathsf{H}(ct_i), x_j \| \mathsf{H}(ct_j)$, breaking $(t, \epsilon_{\mathsf{KDF}})$–collision-resistance of KDF:

$$\mathsf{Adv}^{\mathcal{G}_{\text{A.0}}}_{\mathsf{Kyber}}(\mathcal{A}) \leq \epsilon_{\mathsf{H}} + \epsilon_{\mathsf{KDF}}.$$

Case B (Both decapsulations reject, distinct public keys: $R_i \wedge R_j \wedge pk_i \neq pk_j$).

Here, we bound the probability of the adversary producing ciphertexts that both get implicitly rejected during decapsulation and decapsulate to the same shared secret under distinct public keys.

Game B.0. This is the game conditioned on $R_i \wedge R_j \wedge pk_i \neq pk_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\text{B.0}}}_{\mathsf{Kyber}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[R_i \wedge R_j \wedge pk_i \neq pk_j]}_{\mathsf{Kyber}}(\mathcal{A}).$$

Game B.1 (Colliding z values). We let $\mathcal{G}_{\text{B.0}}$ return false if there is a collision among the z values in the n secret keys. Since the z values are 256-bit strings sampled uniformly at random, by the birthday bound we get:

$$\mathsf{Adv}^{\mathcal{G}_{\text{B.0}}}_{\mathsf{Kyber}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathcal{G}_{\text{B.1}}}_{\mathsf{Kyber}}(\mathcal{A}) + \frac{n^2}{2^{256}}.$$

Game B.2 (Collision in KDF). Now, we have established $z_i \| \mathsf{H}(ct_i) \neq z_j \| \mathsf{H}(ct_j)$ (since $z_i \neq z_j$), yet $\mathsf{KDF}(z_i \| \mathsf{H}(ct_i)) = \mathsf{KDF}(z_j \| \mathsf{H}(ct_j))$. We build a reduction $\mathcal{B}_3$ that outputs $z_i \| \mathsf{H}(ct_i), z_j \| \mathsf{H}(ct_j)$, breaking $(t, \epsilon_{\mathsf{KDF}})$–collision resistance of KDF:

$$\mathsf{Adv}^{\mathcal{G}_{\text{B.1}}}_{\mathsf{Kyber}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}}.$$

Case C (One decapsulation rejects, distinct public keys: $R_i \wedge A_j \wedge pk_i \neq pk_j$).

Here, we bound the probability of the adversary producing exactly one ciphertext that gets implicitly rejected during decapsulation, while both ciphertexts decapsulate to the same shared secret under distinct public keys.

Game C.0. This is the game conditioned on $R_i \wedge A_j \wedge pk_i \neq pk_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\text{C.0}}}_{\mathsf{Kyber}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[R_i \wedge A_j \wedge pk_i \neq pk_j]}_{\mathsf{Kyber}}(\mathcal{A}).$$

If $ss'_i \neq z_j$ (for $ss' = \mathsf{G}_{256}(m'_i \| h_i)$), we can build a reduction $\mathcal{B}_4$ that returns $ss'_i \| \mathsf{H}(ct_i), z_j \| \mathsf{H}(ct_j)$ to break $(t, \epsilon_{\mathsf{KDF}})$–collision-resistance of KDF. Otherwise, we can build a reduction $\mathcal{B}_5$ that guesses $j \in [n]$ and embeds an image of G_{256} in z_j, breaking the $(t, \epsilon'_{\mathsf{G}})$–preimage resistance of G_{256} for randomly sampled images ($\mathsf{PR\text{-}r}$) if it guesses j correctly. Hence,

$$\mathsf{Adv}^{\mathcal{G}_{\text{C.0}}}_{\mathsf{Kyber}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + n \cdot \epsilon'_{\mathsf{G}}.$$

Case D (Both decapsulations accept, distinct public keys: $A_i \wedge A_j \wedge pk_i \neq pk_j$).

Here, we bound the probability of the adversary producing ciphertexts that both get accepted during decapsulation and decapsulate to the same shared secret under distinct public keys.

Game D.0. This is the game conditioned on $A_i \wedge A_j \wedge pk_i \neq pk_j$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\text{D.0}}}_{\mathsf{Kyber}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0[A_i \wedge A_j \wedge pk_i \neq pk_j]}_{\mathsf{Kyber}}(\mathcal{A}).$$

Now, $\mathsf{KDF}(\mathsf{G}_{256}(m_i \| h_i) \| \mathsf{H}(ct_i)) = ss_i = ss_j = \mathsf{KDF}(\mathsf{G}_{256}(m_j \| h_j) \| \mathsf{H}(ct_j))$ for $pk_i \neq pk_j$. It must be that $\mathcal{A}$ provides us with a collision either in KDF, G_{256}, or H. Once more, we can distinguish the cases:

1. In case $\mathsf{G}_{256}(m_i \| h_i) \| \mathsf{H}(ct_i) \neq \mathsf{G}_{256}(m_j \| h_j) \| \mathsf{H}(ct_j)$, then we can build a reduction $\mathcal{B}_6$ that outputs $\mathsf{G}_{256}(m_i \| h_i) \| \mathsf{H}(ct_i), \mathsf{G}_{256}(m_j \| h_j) \| \mathsf{H}(ct_j)$, breaking $(t, \epsilon_{\mathsf{KDF}})$–collision-resistance of KDF.

2. Otherwise, if $m_i \| h_i \neq m_j \| h_j$, then we can build a reduction $\mathcal{B}_7$ that outputs $m_i \| h_i, m_j \| h_j$, breaking $(t, \epsilon_{\mathsf{G}})$–collision-resistance of G_{256}.

3. Otherwise, we must have $m_i \| h_i = m_j \| h_j$ and $\mathsf{H}(pk_i) = h_i = h_j = \mathsf{H}(pk_j)$. By $\mathcal{G}_{\mathrm{C}.0}$, $pk_i \neq pk_j$ and we can build a reduction $\mathcal{B}_8$ that outputs pk_i, pk_j, breaking $(t, \epsilon_{\mathsf{H}})$–collision-resistance of H.

Jointly,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{C}.0}}_{\mathsf{Kyber}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \epsilon_{\mathsf{G}} + \epsilon_{\mathsf{H}}.$$

Collecting the bounds yields the claim. □

4 Security Model

We analyze security of the PQXDH protocol in a computational, game-based security model for authenticated key exchange protocols, following the tradition of Bellare and Rogaway (BR) [BR94], but refined towards the maximum-exposure security that Signal's handshakes (both X3DH and PQXDH) aim at. Our model is based on prior Signal and Signal-like security models by CCDGS [CCD+17] and particularly BFGJS [BFG+22, BFG+21], the latter introducing syntax and modeling for the particular class of asynchronous key exchange protocols that the Signal handshakes belong to.

In BR-style models, a computational adversary interacts with multiple users across multiple sessions of the key exchange protocol, fully controlling the network and being able to reroute, modify, inject, and drop messages at will (through a Send oracle). For maximum-exposure, the model allows the adversary to compromise a user's long-term key (via a CorruptLTKey oracle) and semi-static keys (CorruptSSKey), as well as reveal a session's key (RevealSessKey) and ephemeral randomness (RevealRand). The targeted security property finally is key indistinguishability, asking that keys established in so-called "fresh" sessions are indistinguishable from random keys. "Freshness" here encodes that a session is not trivially compromised and that the protocol under analysis aims to protect against the involved key compromises (defined through a set of "clean" predicates). We will detail those technical bits below.

For the analysis of PQXDH, we extend the prior models in several ways:

- Prior models [CCD+17, BFG+22, BFG+21] assumed semi-static keys being authentically distributed. However, only long-term keys can be verified out-of-band in Signal, and semi-static keys are signed with those keys instead. We are the first to capture these *signed semi-static keys* in a game-based model for a computational security analysis. In both the model and the security analysis (cf. Section 5), capturing these signatures leads to notable added complexity.

- In contrast to Signal's classical X3DH handshake, PQXDH also involves *signed ephemeral (KEM) keys*. We capture this, too, expanding on the set of attack vectors our model covers.

- PQXDH adding KEM keys means that *multiple semi-static keys* (DH and KEM) can now be involved in a handshake. We capture this by accordingly modified identification of semi-static keys.

Note that CCDGS [CCD+17] also analyzed the ratcheting protocol, working in a multi-stage key exchange model [FG14]; PQXDH does not affect the ratcheting part, hence we focus on the initial handshake. BFGJS [BFG+22, BFG+21] in turn also studied deniability, which we do not consider in this work.

In the following, we will first review the syntax required to express both the PQXDH protocol and the security model formally; we mostly follow BFGJS [BFG+22, BFG+21] here, in parts verbatim, and focus on highlighting (with light-gray background) the main differences introduced in this work to account for signed semi-static and ephemeral keys as well as multiple identifiers for semi-static keys.

4.1 Syntax and Notation

Key exchange syntax. We define a two-party key exchange protocol via the following probabilistic algorithms:

- $\mathsf{KGenLT}() \xrightarrow{\$} (ltpk, ltsk)$: The *long-term key generation* algorithm that outputs a party's public-key/secret-key pair.
- $\mathsf{KGenSS}(ltsk) \xrightarrow{\$} ((sspk_1, sssk_1, \sigma_1), \dots)$: The *semi-static key generation* algorithm that takes as input a long-term secret key $ltsk$ and outputs a vector of public-key/secret-key pairs and corresponding signatures.[6]
- $\mathsf{Run}(ltsk, \vec{sssk}, \vec{ltpk}, \pi, m) \xrightarrow{\$} (\pi', m')$: The *session execution algorithm* that takes as input a party's long-term secret key $ltsk$, that party's semi-static secret keys $\vec{sssk}$, all parties' long-term public keys $\vec{ltpk}$, a session state π, and an incoming message m, and outputs an updated session state π' and a (possibly empty) outgoing message m'. The session sending the first message is set up by calling Run with a distinguished message $m = (\mathsf{create}, (\vec{\mathsf{ssid}}, \mathsf{type}))$, where $\vec{\mathsf{ssid}}$ indicates the semi-static keys[7] to be used and type whether a full ($\mathsf{type} = \mathsf{full}$) or reduced ($\mathsf{type} = \mathsf{reduced}$) handshake should be performed.

Note that an explicit ephemeral key generation algorithm is not mandatory and can happen inside of Run.

Parties and sessions. In our model, *parties* $P \in [n_p]$, each holding a long-term public-key/secret-key pair generated by KGenLT may run multiple instances of the protocol (simultaneously or sequentially); we denote the ith such *session* of party P by π_P^i. Each session maintains the following information:

- $\mathsf{oid} \in [n_p]$: The identity of the session owner.
- $\mathsf{pid} \in [n_p] \cup \{\star\}$: The identity of the intended peer, which may initially be unknown (indicated by $\star$).
- $\mathsf{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$: The role of the party.
- $\mathsf{st}_{\mathsf{exec}} \in \{\bot, \mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$: The status of this session's execution.
- $\mathsf{sid} \in \{0,1\}^* \cup \{\bot\}$: A session identifier defining partnering.
- $\mathsf{cid} \in \{0,1\}^* \cup \{\bot\}$: A contributive identifier, defining a preliminary form of partnering (often as a substring or prefix of the session identifier) for the case the session is not yet bound to an authenticated peer [DFGS15].
- $\mathsf{K} \in \mathcal{K}_{\mathsf{KE}} \cup \{\bot\}$: The session key established in this session, initialized to $\bot$.

[6]The length of this vector is protocol-dependent; e.g., in X3DH, semi-static keys consist of one DH key pair, and PQXDH adds a second semi-static key (for a KEM scheme). For simplicity, we have a set of semi-static keys be generated in one operation, yet in the protocol each key can be used independently.

[7]In the model, semi-static keys are identified by some value $\mathsf{ssid} = "s, n"$ where s indicates the KGenSS call through which they were generated and n the key's position in that call's output. In practice, the latter signifies the type of semi-static key if there are several; for PQXDH, $n = 1$ for DH keys and $n = 2$ for KEM keys.

- $\mathsf{type} \in \{\mathsf{full}, \mathsf{reduced}\}$: Indicator whether an ephemeral pre-key was used ($\mathsf{type} = \mathsf{full}$) for key establishment, or not ($\mathsf{type} = \mathsf{reduced}$).
- $\mathsf{coins} \in \mathcal{R}_{\mathsf{KE}}$: The random coins from the randomness space $\mathcal{R}_{\mathsf{KE}}$ used in the execution of Run; set by the game and read-only thereafter.
- $\mathsf{sspks} \in (\{0,1\}^*)^* \cup \{\bot\}$: The semi-static public keys used in this session.

For bookkeeping in the security game we additionally introduce the following flags, which are not accessible by the protocol sessions:

- $\mathsf{revrand} \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the random coins $\pi.\mathsf{coins}$ have been revealed via a RevealRand query. The default value is false.
- $\mathsf{pcorr} \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the peer's long-term key was corrupted at the point in time when this session accepted. The default value is false.

Session partnering. We say two sessions π^i_U and π^j_V are *partnered* if they agree on the session identifier: $\pi^i_U.\mathsf{sid} = \pi^j_V.\mathsf{sid} \neq \bot$. *Contributive* identifiers (cid) indicate when sessions may eventually derive the same key but are not fully partnered (yet); we use these to model security of initiator's keys in incomplete handshakes.

4.2 Security Game

The security property of an authenticated key exchange protocol KE we consider in this work is *indistinguishability of session keys* (KI), formalized through the game $\mathcal{G}^{\mathsf{KI}}_{\mathsf{KE}}(\mathcal{A})$ in Figure 6 played by an adversary $\mathcal{A}$. At the start of the game, a random challenge bit $b_{\mathsf{test}} \leftarrow_{\$} \{0,1\}$ is fixed and long-term public-key/secret-key pairs are generated for all n_p honest parties and their public keys $\vec{ltpk}$ provided to the adversary. The adversary is then able to interact with honest parties via the following queries:

- Send(U, i, m): Sends message m to session π^i_U, which corresponds to executing $\mathsf{Run}(ltsk_U, \vec{sssk}_U, \vec{ltpk}, \pi^i_U, m)$, saving the updated session state π' as π^i_U, and returning the outgoing message m' to the adversary.
- CorruptLTKey(U): Returns party U's long-term secret key $ltsk_U$ to the adversary; recorded through the flag $\mathsf{corrltk}_U$.
- CorruptSSKey(U, ssid): Returns party U's semi-static secret key $sssk^{\mathsf{ssid}}_U$ to the adversary; recorded through the flag $\mathsf{corrssk}_{sspk}$, where $sspk$ is the public key corresponding to $sssk^{\mathsf{ssid}}_U$.[8]
- RevealRand(U, i): Returns the random coins of session π^i_U to the adversary, recorded in the session through the flag $\mathsf{revrand}$.
- RevealSessKey(U, i): If session π^i_U has accepted, return its session key $\pi^i_U.\mathsf{K}$ to the adversary.
- Test(U, i): If the Test query has been called before or session π^i_U has not accepted, then return $\bot$. Otherwise; if $b_{\mathsf{test}} = 0$, return $\pi^i_U.\mathsf{K}$, otherwise return a randomly sampled session key from the protocol's key space $\mathcal{K}_{\mathsf{KE}}$. Record the test session as $\pi^* \leftarrow \pi^i_U$.

[8]Since we model that semi-static keys are not authentically distributed (but signed), sessions only know the public key received ($sspk$), but not the game label ssid, which was used in prior models [CCD+17, BFG+22, BFG+21] to track compromise more easily.

At the end of the game, the adversary outputs a bit b'. The adversary is said to win if $b' = b_{\mathsf{test}}$ and the test session π^* is fresh. Formally, if the test session is fresh, the experiment outputs 1 if $b' = b_{\mathsf{test}}$ and 0 otherwise; if the test session is not fresh, then the experiment outputs a random bit. The adversary's advantage in the key indistinguishability game measured as the experiment outputting 1 minus the adversary's guessing chane, $\frac{1}{2}$.

Definition 4.1 (Key indistinguishability)**.** *Let* KE *be a key exchange protocol and* $\mathcal{A}$ *an adversary against the key indistinguishability (*KI*) game* $\mathcal{G}^{\mathsf{KI}}_{\mathsf{KE}}(\mathcal{A})$ *in Figure 6. We say that* KE *achieves* $(t, \epsilon, (q_{\mathrm{Snd}}, q_{\mathrm{CorrLT}}, q_{\mathrm{CorrSS}}, q_{\mathrm{RevR}}, q_{\mathrm{RevSK}}))$–key indistinguishability*, if for any adversary* $\mathcal{A}$ *against* $\mathcal{G}^{\mathsf{KI}}_{\mathsf{KE}}(\mathcal{A})$ *with running time at most* t *and making at most* q_{Snd}*,* q_{CorrLT}*,* q_{CorrSS}*,* q_{RevR}*, resp.* q_{RevSK} *queries to its* Send*,* CorruptLTKey*,* CorruptSSKey*,* RevealRand*, resp.* RevealSessKey *oracles, we have that*

$$\mathsf{Adv}^{\mathsf{KI}}_{\mathsf{KE}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{KI}}_{\mathsf{KE}}(\mathcal{A})\right] - \frac{1}{2} \le \epsilon.$$

Note that the model restricts the adversary to a single query to the Test *oracle.*

Soundness. The model also captures soundness (via the predicate sound), i.e., that session identifiers appropriately reflect correct protocol executions. Concretely, soundness demands that an adversary cannot create one of the following situations (or else it will win the game immediately):

(i) Two sessions accept with the same session identifier, but derive different session keys, indicate different handshake types (full vs. $\mathsf{reduced}$), or do not agree on their contributive identifiers (Fig. 6, line 18).

(ii) Two initiator sessions accept with the same session identifier (Fig. 6, line 19).

(iii) Three sessions accept with the same session identifier in full handshake type (Fig. 6, line 20).

Freshness. Unrestricted access to the game oracles allows for trivial wins, e.g., by testing a session key and also revealing it (or its partner). The freshness predicate fresh rules out such trivial wins, and further encodes that the adversary has not obtained sufficiently many secrets, via CorruptLTKey and/or CorruptSSKey and/or RevealRand queries, to derive the session key of the test session itself and/or substituted signed keys in its execution; this is encoded through a set of so-called clean predicates (Figure 6, line 16 and lines 46 ff.).

Clean predicates. Following the terminology of CCDGS [CCD+17], the set of clean predicates capture the maximum-exposure properties of the protocol being analyzed. Figure 6, lines 46 ff. formally capture these for PQXDH. The core properties closely follow those for the classic $\mathsf{X3DH}$ handshake [CCD+17, BFG+22]: in handshakes where the long-term/semi-static, ephemeral/long-term, or ephemeral/semi-static (or, for $\mathsf{type} = \mathsf{full}$ handshakes, ephemeral/ephemeral) secrets combination between initiator/responder is not revealed, we expect security of the derived session key. Jumping ahead, for those combinations where PQXDH employs both DH and KEM keys, we will obtain hybrid/combiner security terms in our analysis (see Section 5), capturing that an adversary has to break *both* primitives to successfully attack the protocol via these compromise angles.

In addition to prior work [CCD+17, BFG+22, BFG+21], our clean predicates further model the signatures on the responder's (semi-static and ephemeral public) keys and allow the adversary to replace semi-static and ephemeral keys sent over the network. Instead of assuming authentic distribution of semi-static public keys, we merely preclude the adversary from compromising the (long-term) signing key prior

```
$\mathcal{G}^{\mathsf{KI}}_{\mathsf{KE}}(\mathcal{A})$:
$b_{\mathsf{test}} \leftarrow\!\$ \{0,1\}$ // sample challenge bit
$\pi^* \leftarrow \bot$ // variable for test session
for $U \in [n_p]$ // generate long-term keys
    $(pk_U, sk_U) \leftarrow\!\$ \mathsf{KGenLT}()$
    for $s \in [n_{ss}]$ // generate semi-static keys
        $\big((sspk_U^{s,1}, sssk_U^{s,1}, \sigma_U^{s,1}), \dots\big) \leftarrow\!\$ \mathsf{KGenSS}(ltsk_U)$
    $\vec{sssk}_U \leftarrow \{sssk_U^{s,1}, \dots\}^{s\in[n_{ss}]}$
$\vec{ltpk} \leftarrow \{ltpk_U\}_{U\in[n_p]}$
$\vec{sspk} \leftarrow \{(sspk_U^{s,1}, \sigma_U^{s,1}), \dots\}^{s\in[n_{ss}]}_{U\in[n_p]}$
$b' \leftarrow\!\$ \mathcal{A}(\vec{ltpk}, \vec{sspk})$ // run adversary
if sound() = false: return true // adversary wins if it breaks soundness
if fresh($\pi^*$) = false : $b' \leftarrow 0$ // attack invalid if test session is not fresh
return $[\![b' = b_{\mathsf{test}}]\!]$ // determine win or loss
```

```
fresh($\pi^*$):
if $\pi^*$.revealed = true : return false // test session is revealed
if $\exists\, \pi_V^j \neq \pi^*$ : ($\pi_V^j$.sid = $\pi^*$.sid $\land$ $\pi_V^j$.revealed = true) : return false
    // test session's partner is revealed
return $\mathsf{clean}_{\pi^*.\mathsf{type}}(\pi^*)$
    // test session is clean wrt. to its handshake type (full resp. reduced)
```

```
sound():
return $\forall$ distinct $\pi, \pi', \pi''$(
    ($\pi$.sid = $\pi'$.sid $\neq \bot \implies \pi$.K = $\pi'$.K $\land$ $\pi$.type = $\pi'$.type $\land$ $\pi$.cid = $\pi'$.cid)
    // same session identifiers imply same key, type, contributive identifiers
    and ($\pi$.sid = $\pi'$.sid $\neq \bot \land \pi$.role = initiator $\implies \pi'$.role = responder)
    // session identifiers of two initiator sessions never collide
    and ($\pi$.sid = $\pi'$.sid = $\pi''$.sid $\neq \bot \implies \pi$.type = reduced))
    // session identifiers of three sessions only collide in reduced mode
```

```
SEND($U, i, m$):
if $\pi_U^i = \bot$ // initiate session: for responders, we have $m$ = (create, ($\vec{\mathsf{ssid}}$, type))
    $\pi_U^i$.oid $\leftarrow U$ // set owner identity
    if $m$ = (create, ...): $\pi_U^i$.role $\leftarrow$ responder // set responder role
    else $\pi_U^i$.role $\leftarrow$ initiator // set initiator role ($m$ is first message)
    $\pi_U^i$.coins $\leftarrow\!\$ \mathcal{R}_{\mathsf{KE}}$ // sample session randomness
    $\pi_U^i.\mathsf{st}_{\mathsf{exec}} \leftarrow$ running
$(\pi_U^i, m') \leftarrow \mathsf{Run}(sk_U, \vec{sssk}_U, \vec{ltpk}, \pi_U^i, m)$
    // run session, random coins in $\pi_U^i$
if $\pi_U^i.\mathsf{st}_{\mathsf{exec}}$ = accepted: // flag if peer was corrupted upon acceptance
    $\pi_U^i$.pcorr $\leftarrow \mathsf{corrltk}_{\pi_U^i.\mathsf{pid}}$
return $(m', \pi_U^i.\mathsf{st}_{\mathsf{exec}})$ // return message and session state
```

```
TEST($U, i$):
if $\pi_U^i = \bot$ or $\pi_U^i.\mathsf{st}_{\mathsf{exec}} \neq$ accepted or $\pi^* \neq \bot$: return $\bot$
    // session does not exist, has not accepted yet, or test already asked
$\pi^* \leftarrow \pi_U^i$ // record test session
$\mathsf{K}_0 \leftarrow \pi_U^i$.K
$\mathsf{K}_1 \leftarrow\!\$ \mathcal{K}_{\mathsf{KE}}$
return $\mathsf{K}_{b_{\mathsf{test}}}$ // return real-or-random challenge key
```

```
CORRUPTLTKEY($U$):
$\mathsf{corrltk}_U \leftarrow$ true // mark long-term key corrupted
return $sk_U$ // return long-term secret key
```

```
CORRUPTSSKEY($U$, ssid):
$\mathsf{corrssk}_{sspk_U^{\mathsf{ssid}}} \leftarrow$ true // mark semi-static key corrupted
return $sssk_U^{\mathsf{ssid}}$ // return semi-static secret key
```

```
REVEALRAND($U, i$):
if $\pi_U^i = \bot$: return $\bot$ // session does not exist
$\pi_U^i$.revrand $\leftarrow$ true // mark randomness revealed
return $\pi_U^i$.coins // return session's random coins
```

```
REVEALSESSKEY($U, i$):
if $\pi_U^i = \bot$ or $\pi_U^i.\mathsf{st}_{\mathsf{exec}} \neq$ accepted: return $\bot$
    // session does not exist or has not yet derived session key
$\pi_U^i$.revealed $\leftarrow$ true // mark session key revealed
return $\pi_U^i$.K // return session key
```

```
$\mathsf{clean}_{\mathsf{full}}(\pi^*)$:
return $\mathsf{clean}_{\mathsf{reduced}}(\pi^*)$ or $\mathsf{clean}_{\mathsf{EE}}(\pi^*)$

$\mathsf{clean}_{\mathsf{reduced}}(\pi^*)$:
return $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$ or $\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$ or $\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$

$\mathsf{clean}_{\mathsf{EE}}(\pi^*)$:
return $\neg\pi^*$.revrand and ($\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$ or $\mathsf{clean}_{\mathsf{sigE}}(\pi^*)$)
    // test session randomness is unrevealed and peer's ephemeral contribution is clean or cleanly signed

$\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$:
return
        ($\pi^*$.role = initiator and $\exists \pi \neq \pi^*$ :
    ($\pi$.role = responder and $\pi^*$.cid = $\pi$.cid and $\neg\pi$.revrand))
    // contributively-partnered responder session's randomness is unrevealed
or ($\pi^*$.role = responder and $\exists \pi \neq \pi^*$ :
    ($\pi$.role = initiator and $\pi^*$.sid = $\pi$.sid and $\neg\pi$.revrand))
    // partnered initiator session's randomness is unrevealed

$\mathsf{clean}_{\mathsf{sigE}}(\pi^*)$:
return ($\pi^*$.role = initiator and $\neg\pi^*$.pcorr and $\forall \pi$ :
    (($\pi$.role = responder and $\pi^*$.cid = $\pi$.cid) $\implies \neg\pi$.revrand))
    // long-term (signing) secret of the responder peer was uncompromised upon
    acceptance, and if a partner exists, then that partner's randomness is unrevealed
```

```
$\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$:
return
    ($\pi^*$.role = initiator and $\neg\mathsf{corrltk}_{\pi^*.\mathsf{oid}}$ and
        $\forall sspk \in \pi^*$.sspks : ($\neg\mathsf{corrssk}_{sspk}$) and $\neg\pi^*$.pcorr)
    // tested initiator's long-term and responder's semi-static secrets are uncompromised
      and responder long-term was uncompromised upon acceptance
or ($\pi^*$.role = responder and $\neg\mathsf{corrltk}_{\pi^*.\mathsf{pid}}$ and
        $\forall sspk \in \pi^*$.sspks : ($\neg\mathsf{corrssk}_{sspk}$))
    // initiator long-term and tested responder's semi-static secrets are uncompromised

$\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$:
return
    ($\pi^*$.role = initiator and $\neg\pi^*$.revrand and $\neg\mathsf{corrltk}_{\pi^*.\mathsf{pid}}$)
    // tested initiator's randomness is unrevealed and responder long-term secret is uncompromised
or ($\pi^*$.role = responder and $\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$ and $\neg\mathsf{corrltk}_{\pi^*.\mathsf{oid}}$)
    // intiator's ephemeral contribution is clean and tested responder's long-term secret is uncompromised

$\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$:
return
    ($\pi^*$.role = initiator and $\neg\pi^*$.revrand and
        $\forall sspk \in \pi^*$.sspks : ($\neg\mathsf{corrssk}_{sspk}$) and $\neg\pi^*$.pcorr)
    // tested initiator's randomness is unrevealed and responder semi-static secrets
    are uncompromised and responder long-term was uncompromised upon acceptance
or ($\pi^*$.role = responder and $\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$ and
        $\forall sspk \in \pi^*$.sspks : ($\neg\mathsf{corrssk}_{sspk}$))
    // initiator's ephemeral contribution is clean and tested responder's semi-static secrets are uncompromised
```

Figure 6: Key indistinguishability (KI) game for key exchange protocol KE (top), in which adversary $\mathcal{A}$ has access to oracles SEND, TEST, CORRUPTLTKEY, CORRUPTSSKEY, REVEALRAND, and REVEALSESSKEY (middle), and wrt. to clean predicates for PQXDH (bottom). Highlighted code reflects the main changes compared to the model of BFGJS [BFG+22, BFG+21].

to the targeted session having accepted (cf. Figure 6, lines 54 and 60). Treating signatures explicitly, we also capture the additional guarantees that signing KEM keys give: for the ephemeral/ephemeral combination to contribute security, it is now (also) sufficient to have the peer's contribution be cleanly signed (cf. Figure 6, line 48). To that end we set a flag pcorr in a session if its peer was corrupted upon acceptance (Figure 6, line 29). For PQXDH, this part of the model ensures that a quantum-later adversary cannot, in addition to a "harvest now, decrypt later" attack, replace the ephemeral KEM public key with its own without being noticed.

In more formal detail, let π^* denote the test session. Depending on whether an ephemeral pre-key was used in the key derivation of π^* or not, we apply either the $\mathsf{clean}_{\mathsf{full}}$ or the $\mathsf{clean}_{\mathsf{reduced}}$ predicate to π^*.

Since $\mathsf{clean}_{\mathsf{reduced}}$ is part of the description of $\mathsf{clean}_{\mathsf{full}}$, we first discuss the case $\pi^*.\mathsf{type} = \mathsf{reduced}$. Intuitively, a session key derived in such a session remains unknown to the adversary, if one of the four keys (i.e., excluding the unused DH_4) that constitute the master secret is "clean", i.e., cannot be computed by the adversary. This is the case if one of the following three clean predicates holds for the test session π^*:

$\mathsf{clean}_{\mathsf{LTSS}}$**:** This predicate indicates whether the combination of the long-term key of the initiator and the semi-static keys of the responder are unknown to the adversary. If the test session is an initiator, the responder's signing key must further be uncompromised upon acceptance.

$\mathsf{clean}_{\mathsf{ELT}}$**:** This predicate indicates whether the combination of the ephemeral contribution of the initiator and the long-term key of the responder is unknown to the adversary.

$\mathsf{clean}_{\mathsf{ESS}}$**:** This predicate indicates whether the combination of the ephemeral contribution of the initiator and the semi-static keys of the responder are unknown to the adversary. If the test session is an initiator, the responder's signing key must further be uncompromised upon acceptance.

If the test session π^* is a responder session, the evaluation of $\mathsf{clean}_{\mathsf{ELT}}$ and $\mathsf{clean}_{\mathsf{ESS}}$ necessitates a further predicate called $\mathsf{clean}_{\mathsf{peerE}}$ (in all other cases, it is sufficient to consider the compromise of keys and randomness via the flags corrltk, corrssk, revrand, and pcorr).

$\mathsf{clean}_{\mathsf{peerE}}$**:** This (sub)predicate indicates that the randomness used within any (sid- or cid-)partnered session is unknown to the adversary.

For test sessions in full handshake mode, i.e., where $\pi^*.\mathsf{type} = \mathsf{full}$, it must either hold that $\mathsf{clean}_{\mathsf{reduced}}$ is true or that the additional input to the master secret computation is clean. The latter is captured by the following predicate:

$\mathsf{clean}_{\mathsf{EE}}$**:** This predicate indicates that the ephemeral contribution of the test session is unknown to the adversary and that the ephemeral contribution of the peer is unknown to the adversary (captured by $\mathsf{clean}_{\mathsf{peerE}}$) or was "cleanly" signed (captured by the new $\mathsf{clean}_{\mathsf{sigE}}$ predicate we introduce). It is the second part of this "or" statement by which our model captures that KEM ephemeral keys are protected by signatures against replacement.

Again, the predicate $\mathsf{clean}_{\mathsf{peerE}}$ helps to determine within $\mathsf{clean}_{\mathsf{EE}}$ whether the randomness of the test session's (contributive) partners is unrevealed. The following predicate precludes that the adversary substitutes the responder's ephemeral key and signs it with the signing key obtained from corrupting the long-term key.

$\mathsf{clean}_{\mathsf{sigE}}$**:** This (sub)predicate indicates that for initiator test sessions the peer was not corrupted at the time of acceptance and, if the test session has a contributively partnered session, then the ephemeral contribution of that session is not known to the adversary.

5 PQXDH Analysis

We formalize the PQXDH handshake in Figure 7, highlighting the changes compared to X3DH to achieve post-quantum security in dark gray. Ours is the first reductionist analysis to model the signatures on public keys, which we accent with light gray. Long-term key pairs consist of a DH key pair and a signing key pair.[9] Semi-static key pairs consist of a DH key pair and a KEM key pair, each signed under the long-term signing key. The protocol then works as follows.

First, Bob produces a pre-key bundle with semi-static and ephemeral keys as indicated by the $\vec{\mathsf{ssid}}$ vector and type: The pre-key bundle always includes the semi-static DH key and for reduced handshakes a semi-static KEM key. Only for full handshakes, the pre-key bundle includes ephemeral DH and KEM keys.[10] Signatures on the semi-static DH key and the KEM key (regardless if it is semi-static or ephemeral) are always included.

Second, Alice verifies both signatures and samples an ephemeral DH key pair of her own. She computes several DH secret combinations, namely long-term/semi-static, ephemeral/long-term, ephemeral/semi-static, and ephemeral/ephemeral (the last one only for full handshakes), and encapsulates against Bob's KEM public key. She derives the session key via a key derivation function KDF on input all three/four DH shared secrets and the KEM shared secret and sends her sampled DH ephemeral public key and the KEM ciphertext to Bob.

Third, Bob computes the same DH shared secrets and decapsulates the KEM ciphertext. He derives the session key in the same manner and accepts.

Security

We show that PQXDH achieves key indistinguishability wrt. to the maximum-exposure attack vectors formalized through the clean predicates in our model (cf. Section 4). As for the prior analysis of X3DH [CCD+17] whose proof structure we follow in parts, the bound is highly non-tight; we still give it in concrete terms for clarity.

Theorem 5.1 (Key indistinguishability of PQXDH)**.** *The* PQXDH *protocol given in Figure 7 with randomness space* $\mathcal{R}_{\mathsf{KE}} = \mathcal{R}_{\mathsf{DH.KGen}} \times \mathcal{R}_{\mathsf{KEM.KGen}} \times \mathcal{R}_{\mathsf{KEM.Enc}} \times \mathcal{R}_{\mathsf{SIG.Sig}}$ *achieves* $(t, \epsilon, (q_{\mathrm{Snd}}, q_{\mathrm{CorrLT}}, q_{\mathrm{CorrSS}}, q_{\mathrm{RevR}}, q_{\mathrm{RevSK}}))$*–key indistinguishability, assuming (for* $t' \approx t$*) the GapDH problem in the group* $(\mathbb{G}, g, q)$ *is* $(t', \epsilon_{\mathsf{GDH}}, 1.5(q_{\mathrm{RO}}+q_{\mathrm{Snd}})^2)$*-hard,* KEM *is a* $(t', \epsilon_{\mathsf{CCA}}, n_s)$*–*OW-CCA*-secure and* $(t', \epsilon_{LEAK^+}, n_p{\cdot}n_{ss}{+}n_s)$*–LEAK*$^+$*-*BIND*-SS-*$\{CT, PK\}$*-secure KEM with public-key collision probability* γ_{coll} *and correctness error* δ_{corr}*,* SIG *is a* $(t', \epsilon_{\mathsf{SIG}}, 2n_{ss}+n_s)$*-unforgeable signature scheme, and* KDF *behaves like a (programmable) random oracle. Concretely, for any efficient adversary* $\mathcal{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{KI}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \frac{(n_p + n_p \cdot n_{ss} + n_s)^2}{q} + \gamma_{\mathsf{coll}}(n_p \cdot n_{ss} + n_s) + n_s \cdot \delta_{\mathsf{corr}} + \epsilon_{LEAK^+}$$
$$+ \left(\begin{array}{ll} \quad (n_p \cdot (\epsilon_{\mathsf{SIG}} + n_p \cdot n_{ss} \cdot \epsilon_{\mathsf{GDH}})) & // \ \mathsf{clean}_{\mathsf{LTSS}} \\ + (n_s \cdot n_p \cdot \epsilon_{\mathsf{GDH}}) & // \ \mathsf{clean}_{\mathsf{ELT}} \\ + (n_p \cdot (\epsilon_{\mathsf{SIG}} + n_{ss} \cdot n_s \cdot \epsilon_{\mathsf{GDH}})) & // \ \mathsf{clean}_{\mathsf{ESS}} \wedge \mathsf{type} = \mathsf{full} \\ + (n_p \cdot (\epsilon_{\mathsf{SIG}} + n_{ss} \cdot n_s \cdot \min(\epsilon_{\mathsf{GDH}}, \epsilon_{\mathsf{CCA}}))) & // \ \mathsf{clean}_{\mathsf{ESS}} \wedge \mathsf{type} = \mathsf{reduced} \\ + (n_s^2 \cdot \min(\epsilon_{\mathsf{GDH}}, \epsilon_{\mathsf{CCA}})) & // \ \mathsf{clean}_{\mathsf{EE}} \wedge \mathsf{clean}_{\mathsf{peerE}} \\ + (n_p \cdot (\epsilon_{\mathsf{SIG}} + n_s^2 \cdot q_{\mathrm{RO}} \cdot \epsilon_{\mathsf{CCA}})) & // \ \mathsf{clean}_{\mathsf{EE}} \wedge \mathsf{clean}_{\mathsf{sigE}} \end{array} \right).$$

For easier accessibility, we first give a summary of the proof; the full proof follows below.

[9]The PQXDH description [KS24] suggests to use a single key pair for both DH and the signature scheme XEdDSA [Per16], modeling which we leave to future work.

[10]Ephemeral DH and KEM pre-keys are used as long as there are some left on the Signal server. In practice, ephemeral DH and KEM pre-keys may run out at different points in time. We do not model this to improve accessibility of the proof.

```
$\underline{\mathsf{KGenLT}():}$
$(ltpk^{\mathsf{DH}}, ltsk^{\mathsf{DH}}) \leftarrow\!\!\$ \ \mathsf{DH.KGen}()$
$(ltpk^{\mathsf{SIG}}, ltsk^{\mathsf{SIG}}) \leftarrow\!\!\$ \ \mathsf{SIG.KGen}()$
$\textbf{return}\ ((ltpk^{\mathsf{DH}}, ltpk^{\mathsf{SIG}}),\ (ltsk^{\mathsf{DH}}, ltsk^{\mathsf{SIG}}))$
```

```
$\underline{\mathsf{KGenSS}(ltsk):}$
$(sspk^{\mathsf{DH}}, sssk^{\mathsf{DH}}) \leftarrow\!\!\$ \ \mathsf{DH.KGen}()$ ; $(sspk^{\mathsf{KEM}}, sssk^{\mathsf{KEM}}) \leftarrow\!\!\$ \ \mathsf{KEM.KGen}()$
$(ltsk^{\mathsf{DH}}, ltsk^{\mathsf{SIG}}) \leftarrow ltsk$
$\sigma^{\mathsf{DH}} \leftarrow\!\!\$ \ \mathsf{SIG.Sig}(ltsk^{\mathsf{SIG}}, sspk_B^{\mathsf{DH}})$ ; $\sigma^{\mathsf{KEM}} \leftarrow\!\!\$ \ \mathsf{SIG.Sig}(ltsk^{\mathsf{SIG}}, sspk_B^{\mathsf{KEM}})$
$sssk^{\mathsf{DH}*} \leftarrow (sssk^{\mathsf{DH}}, sspk^{\mathsf{DH}}, \sigma^{\mathsf{DH}})$ ; $sssk^{\mathsf{KEM}*} \leftarrow (sssk^{\mathsf{KEM}}, sspk^{\mathsf{KEM}}, \sigma^{\mathsf{KEM}})$
$\textbf{return}\ ((sspk^{\mathsf{DH}}, sssk^{\mathsf{DH}*}, \sigma^{\mathsf{DH}}), (sspk^{\mathsf{KEM}}, sssk^{\mathsf{KEM}*}, \sigma^{\mathsf{KEM}}))$
```

Alice **Bob**

```
Bob:
$\underline{\mathsf{Run}(ltsk_B, \vec{ltpk}, \vec{sssk}_B, \pi_B, (\mathsf{create}, (\vec{\mathsf{ssid}}, \mathsf{type})))}$
$(r_1, r_2, r_3, \bot) \leftarrow \pi_B.\mathsf{coins}$
$(sssk_B^{\mathsf{DH}}, sspk_B^{\mathsf{DH}}, \sigma_B^{\mathsf{DH}}) \leftarrow sssk_B^{\vec{\mathsf{ssid}}[1]}$
$\textbf{if}\ \mathsf{type} = \mathsf{full}$ // full handshake
    $(epk_B^{\mathsf{DH}}, esk_B^{\mathsf{DH}}) \leftarrow \mathsf{DH.KGen}(; r_1)$
    $(epk_B^{\mathsf{KEM}}, esk_B^{\mathsf{KEM}}) \leftarrow \mathsf{KEM.KGen}(; r_2)$
    $(ltsk_B^{\mathsf{DH}}, ltsk_B^{\mathsf{SIG}}) \leftarrow ltsk_B$
    $\sigma^{\mathsf{KEM}} \leftarrow \mathsf{SIG.Sig}(ltsk_B^{\mathsf{SIG}}, epk_B^{\mathsf{KEM}}; r_3)$
    $epk_B \leftarrow (epk_B^{\mathsf{DH}}, epk_B^{\mathsf{KEM}})$
    $sspk \leftarrow (sspk_B^{\mathsf{DH}}, \bot)$
$\textbf{else}$ // reduced handshake
    $(sspk_B^{\mathsf{KEM}}, sssk_B^{\mathsf{KEM}}, \sigma_B^{\mathsf{KEM}}) \leftarrow sssk_B^{\vec{\mathsf{ssid}}[2]}$
    $epk_B \leftarrow \bot$
    $sspk \leftarrow (sspk_B^{\mathsf{DH}}, sspk_B^{\mathsf{KEM}})$
    $\sigma^{\mathsf{KEM}} \leftarrow \sigma_B^{\mathsf{KEM}}$
$\pi_B.\mathsf{pid} \leftarrow \star$
$\pi_B.\mathsf{sspks} \leftarrow sspk$
$\pi_B.\mathsf{type} \leftarrow \mathsf{type}$
$\pi_B.\mathsf{cid} \leftarrow (B, ltpk_B, sspk, epk_B)$
$\textbf{return}\ (\pi_B, m = (B, sspk, epk_B, \sigma_B^{\mathsf{DH}}, \sigma^{\mathsf{KEM}}))$
```

m

```
Alice:
$\underline{\mathsf{Run}(ltsk_A, \vec{sssk}_A, \vec{ltpk}, \pi_A, m; \mathsf{coins})}$
$(r_4, \bot, \bot, r_5) \leftarrow \pi_A.\mathsf{coins}$
$(ltsk_A^{\mathsf{DH}}, ltsk_A^{\mathsf{SIG}}) \leftarrow ltsk_A$
$(epk_A^{\mathsf{DH}}, esk_A^{\mathsf{DH}}) \leftarrow \mathsf{DH.KGen}(; r_4)$
$(B, sspk, epk_B, \sigma_B^{\mathsf{DH}}, \sigma^{\mathsf{KEM}}) \leftarrow m$
$(ltpk_B^{\mathsf{DH}}, ltpk_B^{\mathsf{SIG}}) \leftarrow ltpk_B$
$(sspk_B^{\mathsf{DH}}, sspk_B^{\mathsf{KEM}}) \leftarrow sspk$
$\textbf{if}\ \mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, sspk_B^{\mathsf{DH}}, \sigma_B^{\mathsf{DH}}) = \mathsf{false}$
    $\textbf{return}\ (\pi_A, \epsilon)$
$DH_1 \leftarrow \mathsf{DH}(ltsk_A^{\mathsf{DH}}, sspk_B^{\mathsf{DH}})$
$DH_2 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, ltpk_B^{\mathsf{DH}})$
$DH_3 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, sspk_B^{\mathsf{DH}})$
$\textbf{if}\ epk_B \neq \bot$ // full handshake
    $(epk_B^{\mathsf{DH}}, epk_B^{\mathsf{KEM}}) \leftarrow epk_B$
    $DH_4 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, epk_B^{\mathsf{DH}})$
    $\textbf{if}\ \mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, epk_B^{\mathsf{KEM}}, \sigma^{\mathsf{KEM}}) = \mathsf{false}$
        $\textbf{return}\ (\pi_A, \epsilon)$
    $(ct, ss) \leftarrow \mathsf{KEM.Enc}(epk_B^{\mathsf{KEM}}; r_5)$
    $\pi_A.\mathsf{type} \leftarrow \mathsf{full}$
$\textbf{else}$ // reduced handshake
    $DH_4 \leftarrow \epsilon$
    $\textbf{if}\ \mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, sspk_B^{\mathsf{KEM}}, \sigma^{\mathsf{KEM}}) = \mathsf{false}$
        $\textbf{return}\ (\pi_A, \epsilon)$
    $(ct, ss) \leftarrow \mathsf{KEM.Enc}(sspk_B^{\mathsf{KEM}}; r_5)$
    $\pi_A.\mathsf{type} \leftarrow \mathsf{reduced}$
$\pi_A.\mathsf{K} \leftarrow \mathsf{KDF}(DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss)$
$\pi_A.\mathsf{sid} \leftarrow \begin{pmatrix} A, B, ltpk_A, ltpk_B, \\ sspk, epk_B, epk_A^{\mathsf{DH}}, ct \end{pmatrix}$
$\pi_A.\mathsf{pid} \leftarrow B$
$\pi_A.\mathsf{sspks} \leftarrow sspk$
$\pi_A.\mathsf{cid} \leftarrow (B, ltpk_B, sspk, epk_B)$
$\pi_A.\mathsf{st_{exec}} \leftarrow \mathsf{accepted}$
$\textbf{return}\ (\pi_A, m' = (A, epk_A^{\mathsf{DH}}, ct))$
```

m'

```
Bob:
$\underline{\mathsf{Run}(ltsk_B, \vec{sssk}_B, \vec{ltpk}, \pi_B, m'; \mathsf{coins})}$
$(A, epk_A^{\mathsf{DH}}, ct) \leftarrow m'$
$(ltpk_A^{\mathsf{DH}}, ltpk_A^{\mathsf{SIG}}) \leftarrow ltpk_A$
$DH_1 \leftarrow \mathsf{DH}(ltpk_A^{\mathsf{DH}}, sssk_B^{\mathsf{DH}})$
$DH_2 \leftarrow \mathsf{DH}(epk_A^{\mathsf{DH}}, ltsk_B^{\mathsf{DH}})$
$DH_3 \leftarrow \mathsf{DH}(epk_A^{\mathsf{DH}}, sssk_B^{\mathsf{DH}})$
$\textbf{if}\ \pi_B.\mathsf{type} = \mathsf{full}$ // full handshake
    $DH_4 \leftarrow \mathsf{DH}(epk_A^{\mathsf{DH}}, esk_B^{\mathsf{DH}})$
    $ss \leftarrow \mathsf{KEM.Dec}(esk_B^{\mathsf{KEM}}, ct)$
$\textbf{else}$ // reduced handshake
    $DH_4 \leftarrow \epsilon$
    $ss \leftarrow \mathsf{KEM.Dec}(sssk_B^{\mathsf{KEM}}, ct)$
$\pi_B.\mathsf{K} \leftarrow \mathsf{KDF}(DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss)$
$\pi_B.\mathsf{sid} \leftarrow \begin{pmatrix} A, B, ltpk_A, ltpk_B, \\ sspk, epk_B, epk_A^{\mathsf{DH}}, ct \end{pmatrix}$
$\pi_B.\mathsf{pid} \leftarrow A$
$\pi_B.\mathsf{st_{exec}} \leftarrow \mathsf{accepted}$
$\textbf{return}\ (\pi_B, \epsilon)$
```

Figure 7: The PQXDH protocol. Darker gray code lines are additions compared to the original X3DH protocol to add post-quantum security. Lighter gray code lines are signature elements of the X3DH and PQXDH not covered in the prior analyses of [CCD+17].

Proof summary. The proof proceeds via a series of game hops. First, we exclude collisions in the DH and KEM public keys, yielding the terms $\frac{(n_p + n_p \cdot n_{ss} + n_s)^2}{q} + \gamma_{\mathsf{coll}}(n_p \cdot n_{ss} + n_s)$. Second, we exclude KEM correctness errors when decapsulating in sessions ($n_s \cdot \delta_{\mathsf{corr}}$). Now, soundness holds, as the session identifiers carrying non-colliding DH/KEM keys make them non-colliding themselves and sessions agreeing on session identifiers in particular derive the same session keys. Third, we ensure that KEM shared secrets bind the public keys and ciphertexts, introducing the ϵ_{LEAK^+} term; this in particular rules out re-encapsulation attacks where sessions that are not partnered possibly derive the same session key.[11]

At this stage we separate the proof into several cases based on the clean predicate (cf. the annotated sum in the bound of Theorem 5.1). Let us illustrate the fourth case ($\mathsf{clean}_{\mathsf{ESS}} \wedge \mathsf{type} = \mathsf{reduced}$) here since it covers the most interesting proof techniques, including handling the signatures and a hybrid DH/KEM argument; the other cases follow in a similar fashion. In this case, the $\mathsf{clean}_{\mathsf{ESS}}$ predicate is satisfied and the adversary is testing a session running a reduced ($\mathsf{type} = \mathsf{reduced}$) handshake. In the proof, we first guess the responder identity involved in the test session, introducing a n_p loss. Second, we ensure that the guessed responder's semi-static KEM and DH keys are not tampered with in the test session, reducing to the unforgeability of the signature scheme (ϵ_{SIG}). Third and fourth, we guess the two semi-static key identifiers of the responder as well as the initiator session, with a loss of $n_{ss}^2 \cdot n_s$. Fifth, we abort if the adversary queries the key derivation function KDF, modeled as random oracle, on the master secret of the test session. To bound this step, we embed *both* a GapDH challenge *and* a OW-CCA challenge into the test session (concretely, into the ephemeral DH and KEM encapsulation contribution of the initiator and the semi-static DH and KEM keys of the responder, which we both guessed before). If the adversary detects this change, the reduction can win *both* games, yielding the minimum of the advantages as the *hybrid* bound, i.e., $\min(\epsilon_{\mathsf{GDH}}, \epsilon_{\mathsf{CCA}})$. Finally, we can replace the session key of the test session with a random key, making it independent of the challenge bit b_{test}. Concluding that the session key of the test session is now independent of b_{test} since non-partnered sessions derive distinct keys (in particular due to the KEM binding hop in the beginning), this completes the proof case. $\square$

We get hybrid guarantees for the fourth and fifth proof cases ($\mathsf{clean}_{\mathsf{ESS}}$ in reduced mode and $\mathsf{clean}_{\mathsf{EE}}$ in full mode with a clean peer, i.e., $\mathsf{clean}_{\mathsf{peerE}}$), showing that for these attack vectors, an adversary would need to break *both* GapDH in the DH group and OW-CCA security of the KEM. The other attack vectors either involve only DH secrets (and hence do not provide post-quantum security), or, for the sixth case ($\mathsf{clean}_{\mathsf{EE}}$ in full mode with clean signatures), the adversary can manipulate the (unsigned) ephemeral DH key, hence only the KEM secret ensures security here. All in all, PQXDH *maintains* the classical guarantees of $\mathsf{X3DH}$ for the first three cases (as intended), extends the guarantees for the fourth and fifth case to provide *hybrid* guarantees, and *adds* a post-quantum guarantee for the last case due to the signed ephemeral KEM key (compared to no guarantee under this attack vector for $\mathsf{X3DH}$).

We now give the full proof for Theorem 5.1.

Proof. **Game 0.** We start with Game $\mathcal{G}_0$ being the original key indistinguishability game $\mathcal{G}^{\mathsf{KI}}_{\mathsf{PQXDH}}(\mathcal{A})$, i.e.,

$$\mathsf{Adv}^{\mathsf{KI}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game 1 (DH and KEM key collisions). We let $\mathcal{G}_0$ abort (overwriting the adversary's output with 0) if any two honestly generated DH keys or KEM keys coincide. There are n_p many long-term DH keys, $n_p \cdot n_{ss}$ many semi-static DH keys, and at most n_s many ephemeral DH keys, so by the birthday bound

[11] As we will detail in the proof (Game 3) and discuss in Section 6, if PQXDH included the session context (e.g., the values in the session identifier) in the key derivation, as is good practice, this problem would not occur: different KEM public keys or ciphertexts would lead to different session keys (by being part of the KDF input as session context).

the probability for any two DH keys colliding can be upper-bounded by $\frac{(n_p+n_p\cdot n_{ss}+n_s)^2}{q}$. As for KEM keys, there are $n_p \cdot n_{ss}$ semi-static ones and at most n_s ephemeral ones, which collide with probability at most $\gamma_{\mathsf{coll}}(n_p \cdot n_{ss} + n_s)$; cf. Definition 3.1. Put together, we have

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \frac{(n_p + n_p \cdot n_{ss} + n_s)^2}{q} + \gamma_{\mathsf{coll}}(n_p \cdot n_{ss} + n_s) + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game 2 (KEM correctness). We next let $\mathcal{G}_1$ abort if any honestly created KEM encapsulation fails to decapsulate correctly. This introduces a KEM correctness error term for each of the n_s many sessions:

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_s \cdot \delta_{\mathsf{corr}} + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Soundness. At this point soundness holds unconditionally; we consider the three sub-conditions of the sound predicate:

- *Agreement on shared key, type, contributive identifiers* (Figure 6, line 18): The DH keys and the KEM key and the KEM ciphertext in the session identifier determine all inputs to the key derivation function KDF. By $\mathcal{G}_2$, KEM ciphertexts decapsulate correctly and both parties derive the same session key. The session identifier further determines the type by epk_B being empty ($\mathsf{type} = \mathsf{reduced}$) or not ($\mathsf{type} = \mathsf{full}$). Finally, the contributive identifier is a subset of the session identifier, hence matching if the latter match.

- *No initiator session identifiers collide* (Figure 6, line 19): Every initiator samples a fresh ephemeral DH key epk^{DH}_A included in the session identifier. These DH keys do not collide by Game $\mathcal{G}_1$. Hence, initiator session identifiers do not collide.

- *No three session identifiers collide in full mode* (Figure 6, line 20): Three colliding session identifiers implies colliding identifiers for two initiator sessions or for two responder sessions. Collisions of initiator session are ruled out above already. Collisions of responder sessions imply colliding ephemeral DH keys epk^{DH}_B, which are ruled out by Game $\mathcal{G}_1$.

Game 3 (KEM shared secret collisions). We will need later that two sessions that use distinct (semi-static or ephemeral) KEM public keys or distinct ciphertexts do not end up using the same shared secret ss (and hence possibly the same session key, despite not being partnered).[12] In this game, we abort if any two sessions derive the same shared secret ss while using different KEM public keys (either key may be an ephemeral key in a full session or a semi-static key in a reduced session) or different KEM ciphertexts.

We bound the probability of this abort happening by the advantage of a reduction $\mathcal{B}_1$ in breaking KCR security of the KEM scheme for $n = n_p \cdot n_{ss} + n_s$. Let $\mathcal{B}_1$ use the first $n_p \cdot n_{ss}$ challenge keys as semi-static KEM keys in KGenSS instead of generating them itself, and the remaining challenge keys in place of the (at most n_s) ephemeral KEM keys generated by responder sessions (letting the $\mathsf{KEM.KGen}$ randomness obtained in the KCR game replace r_2 in the session's random coins). If any two sessions π_i, π_j obtain the same KEM shared secret $ss_i = ss_j$ (as the output of encapsulation for initiator sessions, or as the

[12] If PQXDH included the session context (e.g., the values in the session identifier) in the key derivation, we would not need this game hop, since different KEM public keys or ciphertexts would lead to different session keys. See also the discussion in Section 6.

output of decapsulation for responder sessions) involving two distinct public keys pk_i, pk_j or two distinct ciphertexts ct_i, ct_j, $\mathcal{B}_1$ outputs (i, ct_i, j, ct_j) and wins. Hence,

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \epsilon_{LEAK^+} + \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Separating the clean cases. At this point, we will divide the proof into six sub-cases following the structure of the $\mathsf{clean}_{\mathsf{full}}$ resp. $\mathsf{clean}_{\mathsf{reduced}}$ predicates evaluated on the test session π^*. We can bound the advantage of the adversary in Game $\mathcal{G}_3$ by the sum of its advantages in Games $\mathcal{G}_3[\mathsf{c}]$ which are $\mathcal{G}_3$ conditioned on a sub-predicate c:

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \sum_{\substack{\mathsf{c} \in \{ \mathsf{clean}_{\mathsf{LTSS}}(\pi^*),\ \mathsf{clean}_{\mathsf{ELT}}(\pi^*), \\ \mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{full},\ \mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{reduced}, \\ \mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{peerE}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{full}, \\ \mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{sigE}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{full} \}}} \mathsf{Adv}^{\mathcal{G}_3[\mathsf{c}]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

In the remainder of the proof, we will bound each of these cases, numbered A–F, separately.

Case A ($\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$).

In this proof case, the $\mathsf{clean}_{\mathsf{LTSS}}$ predicate ensures for the test session π^* that

1. the initiator's long-term key is uncompromised,
2. the responder's semi-static key is uncompromised, and
3. if π^* is an initiator, the responder's long-term (signing) key was uncompromised upon acceptance.

Via the last point, we can guarantee that initiator and responder indeed agree on the semi-static key (identifier), given signatures are unforgeable. Then, similar to the classical Signal proof [CCD+17], the uncompromised long-term/semi-static Diffie–Hellman combination then ensures key indistinguishability for the test session.

Game A.0. This case begins with Game $\mathcal{G}_3$ conditioned on $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.0}}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_3[\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game A.1 (Guess responder identity V^*). We first guess the identity V^* of the responder involved in the test session, overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.0}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_p \cdot \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.1}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game A.2 (Signature unforgeability). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the test session π^* is an initiator session and accepts using a semi-static DH public key $sspk^{\mathsf{DH}}_{V^*}$ that was not generated through a KGenSS run for V^*. This ensures that the test session accepts with $\pi^*.\mathsf{sspks} = (sspk^{\mathsf{DH}}_{V^*}, \cdot)$ corresponding to a DH key pair of which the adversary does not know the secret key, since $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$ guarantees that the semi-static secret belong to $sspk^{\mathsf{DH}}_{V^*}$ is not compromised. The probability of such an abort can be bounded by the advantage of the following reduction $\mathcal{B}_2$ against the $(t, \epsilon_{\mathsf{SIG}}, 2n_{ss} + n_s)$-unforgeability of SIG.

The reduction $\mathcal{B}_2$ samples all key components itself except for the signature key of V^*: In place of the long-term public signature key $ltpk^{\mathsf{SIG}}_{V^*}$ of V^* it uses the public key pk obtained in its unforgeability game. In its simulation of Game $\mathcal{G}_{\mathrm{A.1}}$, $\mathcal{B}_2$ uses its signing oracle to obtain signatures under $ltsk^{\mathsf{SIG}}_{V^*}$: two per semi-static key (for DH and KEM public keys) and up to n_s signatures on ephemeral KEM keys. Since $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*) = \mathsf{true}$, we know that $\pi^*.\mathsf{pcorr} = \mathsf{false}$, i.e., the long-term key of V^* was not corrupted when the test session accepted and so $\mathcal{B}_2$ does not have to answer a CorruptLTKey(V^*) query prior to the abort event. Hence, $\mathcal{B}_2$ provides a perfect simulation of Game $\mathcal{G}_{\mathrm{A.1}}$ up to when the abort would happen, and if π^* receives a signature σ_{V^*} on a semi-static DH public key that V^* did not generate, then $\mathcal{B}_2$ can output this as its forgery and wins. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.1}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \epsilon_{\mathsf{SIG}} + \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.2}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game A.3 (Guess initiator identity U^*). We guess the identity U^* of the initiator to the test session, overwriting the adversary's bit guess with 0 if this guess was incorrect. This step again loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.2}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_p \cdot \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.3}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game A.4 (Guess semi-static key identifier ssid^* of V^*). We now guess the identifier ssid of the responder V^*'s (uncorrupted) semi-static DH key $sspk^{\mathsf{DH}}_{V^*}$. Note that depending on the role of π^* this is either the test session's own key (if $\pi^*.\mathsf{role} = \mathsf{responder}$), or of the intended peer (if $\pi^*.\mathsf{role} = \mathsf{initiator}$). We denote the guessed identifier by ssid^*, and abort, setting the adversary's output bit to 0, if this guess is incorrect, losing at most a factor of the number of semi-static keys per user n_{ss}:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.3}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_{ss} \cdot \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.4}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game A.5 (GapDH). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the adversary queries the random oracle on a value formatted like a master secret and beginning with $\mathsf{CDH}(ltpk^{\mathsf{DH}}_{U^*}, sspk^{\mathsf{DH}}_{V^*})$, i.e., the shared DH key between $ltpk^{\mathsf{DH}}_{U^*}$ and $sspk^{\mathsf{DH}}_{V^*}$, where $sspk^{\mathsf{DH}}_{V^*}$ is the semi-static DH key with identifier ssid^*. The probability of such an abort can be bounded by the advantage of the following reduction $\mathcal{B}_3$ against the $(t, \epsilon_{\mathsf{GDH}}, q_{\mathrm{DDH}})$-hardness of the GapDH problem in $(\mathbb{G}, g, q)$.

Reduction $\mathcal{B}_3$ samples all key components itself except for replacing the long-term DH key $ltpk^{\mathsf{DH}}_{U^*}$ of U^* and the semi-static DH key $sspk^{\mathsf{DH}}_{V^*}$ of V^* with its own challenge values g^a and g^b. Since $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*) = \mathsf{true}$, $\mathcal{B}_3$ never has to answer the queries CorruptLTKey(U^*) or CorruptSSKey(V^*, ssid^*) in its simulation of Game $\mathcal{G}_{\mathrm{A.4}}$. Though, $\mathcal{B}_3$ may have to answer for RevealSessKey and Send queries that involve the substituted keys. Hence, the reduction patches the random oracle and the Send oracle to generate the session keys in a consistent manner. In consequence, the reduction answers queries to the RevealSessKey oracle consistently as well.

Figure 8 gives an algorithmic description of the changes. In particular, the reduction gives special treatment to Send queries that involve any of the challenge secrets, i.e., for Send queries where the initiator is the initiator of the test session U^* partnered with the responder of the test session V^* using the semi-static key ssid^* of the test session, the responder is U^*, or the responder is V^* with semi-static key ssid^* and the initiator is not U^*. In these cases the reduction cannot compute all DH shared secrets

```
$RO'(x)$:
if $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss = x$ // if we can parse the query string $x$ as three or four DH shared secrets ($DH_4$ may be empty) and a KEM shared secret...
    if $\mathrm{DDH}(ltpk^{\mathsf{DH}}_{U^*}, sspk^{\mathsf{DH}}_{V^*}, DH_1)$ // referring to $V^*$'s key from $\mathsf{ssid}^*$
        $\mathcal{B}_3$ halts and returns $DH_1$ as its GapDH solution // change for Game $\mathcal{G}_{A.5}$
    return $\mathsf{patch}(DH_1, DH_2, DH_3, DH_4, ss)$ // extra routine to ensure consistency
return $RO(x)$

$\mathsf{patch}(PoS_1, PoS_2, PoS_3, DH_4, ss)$:
    // the first three arguments may pairs of DH public keys or DH shared secrets
foreach $(e_1, e_2, e_3, e_4, e_5, y)$ in $L$ // iterate over list entries
    if $\mathsf{DDH\text{-}eq}(PoS_1, e_1) \wedge \mathsf{DDH\text{-}eq}(PoS_2, e_2) \wedge \mathsf{DDH\text{-}eq}(PoS_3, e_3) \wedge DH_4 = e_4 \wedge ss = e_5$
        return $y$ // query matches a prior one, respond with recorded value $y$
$y \leftarrow\!\!\$\ \{0,1\}^{256}$ // new query; sample value at random
$L \leftarrow L \cup \{(PoS_1, PoS_2, PoS_3, DH_4, ss, y)\}$ // record response $y$ on this query
return $y$

$\mathsf{DDH\text{-}eq}(PoS_1, PoS_2)$:
if $(pk_1, pk_2) = PoS_1 \wedge$ if $(pk_3, pk_4) = PoS_2$ // if both $PoS_1$ and $PoS_2$ parse as two public keys...
    return $\{pk_1, pk_2\} = \{pk_3, pk_4\}$
if $(pk_1, pk_2) = PoS_1$ // if $PoS_1$ parses as two public keys...
    return $\mathrm{DDH}(pk_1, pk_2, PoS_2)$
if $(pk_1, pk_2) = PoS_2$ // if $PoS_2$ parses as two public keys...
    return $\mathrm{DDH}(pk_1, pk_2, PoS_1)$
return $PoS_1 = PoS_2$ // else, both are DH shared secrets

$\textsc{Send}(W, i, m)$ substitutes KDF call in Run where GapDH challenge keys are involved:
...
$U \leftarrow \pi^i_W.\mathsf{initiator}$   $V \leftarrow \pi^i_W.\mathsf{responder}$   $\mathsf{ssid} \leftarrow \pi^i_W.\mathsf{ssid}$
$(\ldots, epk^{\mathsf{DH}}_U, \ldots) \leftarrow \pi^i_W.\mathsf{sid}$
if $U = U^* \wedge V = V^* \wedge \mathsf{ssid} = \mathsf{ssid}^*$
    compute $DH_2, DH_4, ss$ with the corresponding secret keys as specified in Run
    $\pi^i_W.\mathsf{K} \leftarrow \mathsf{patch}((ltpk^{\mathsf{DH}}_{U^*}, sspk^{\mathsf{DH}}_{V^*}), DH_2, (epk^{\mathsf{DH}}_U, sspk^{\mathsf{DH}}_{V^*}), DH_4, ss)$
else if $V = U^*$
    compute $DH_1, DH_3, DH_4, ss$ with the corresponding secret keys as specified in Run
    $\pi^i_W.\mathsf{K} \leftarrow \mathsf{patch}(DH_1, (epk^{\mathsf{DH}}_U, ltpk^{\mathsf{DH}}_{U^*}), DH_3, DH_4, ss)$
else if $U \neq U^* \wedge V = V^* \wedge \mathsf{ssid} = \mathsf{ssid}^*$
    compute $DH_1, DH_2, DH_4, ss$ with the corresponding secret keys as specified in Run
    $\pi^i_W.\mathsf{K} \leftarrow \mathsf{patch}(DH_1, DH_2, (epk^{\mathsf{DH}}_U, sspk^{\mathsf{DH}}_{V^*}), DH_4, ss)$
else // $\mathcal{B}_3$ can compute all secrets
    compute $DH_1, DH_2, DH_3, DH_4, ss$ with the corresponding secret keys as specified in Run
    $\pi^i_W.\mathsf{K} \leftarrow \mathsf{patch}(DH_1, DH_2, DH_3, DH_4, ss)$
...
```

Figure 8: Algorithmic description of how $\mathcal{B}_3$ simulates the random oracle and the SEND oracle in game $\mathcal{G}_{A.5}$. To ensure consistency of queries we use an additional (global) list L, which is initialized as empty. This list L contains six-tuples consisting of three pairs of DH public keys or DH shared secrets (PoS), a DH shared secret, a KEM shared secret, and the associated RO output. The patch routine checks if incoming queries match list entries with the DDH-eq routine: DDH-eq returns true if both PoS argument are the same two public keys, if the two PoS arguments are a valid DH tuple according to the DDH oracle, or if the two PoS arguments are identical DH shared secrets. We patch the SEND oracle to use the patch routine in lieu of computing the KDF, whenever there are challenge secret keys involved that $\mathcal{B}_3$ does not know.

entering the key derivation function and chooses a session key via the patch routine. Specifically, the patch routine takes as arguments three pairs of DH public keys or DH shared secrets PoS (since DH_1, DH_2, DH_3 are the DH shared secrets that the reduction can potentially not compute), a DH shared secret, and a KEM shared secret, and returns a RO output that is consistent with previous queries. To this end, patch saves all queries together with their RO response in a list L and checks new queries against L. Note here that two PoS are equivalent if the pair of public keys in one of them forms a valid Diffie–Hellman triple with the shared secret in the other one according to the DDH oracle or if the PoS components are equivalent, as described by the $\mathsf{DDH\text{-}eq}$ routine. If a query does not match any previous query, the patch routine randomly chooses a new session key and records the query with the response in L.

Since the SEND oracle computes consistent session keys for each session, $\mathcal{B}_3$ also consistently answers to the REVEALSESSKEY oracle queries. To ensure consistency with queries to the random oracle, the reduction also uses the patch routine for RO queries. Finally, in case the adversary queries the random oracle on a value beginning with $\mathsf{CDH}(ltpk^{\mathsf{DH}}_{U^*}, sspk^{\mathsf{DH}}_{V^*})$, the reduction $\mathcal{B}_3$ halts and returns this value to its own challenger.

Note that for each RO query the reduction makes up to $1 + 2|L| \leq 3|L|$ queries to the DDH oracle (line 2 and up to two times per entry in line 7). For each query to the SEND oracle,[13] the reduction calls the patch routine once, which makes up to $3|L|$ queries to the DDH oracle. Each call to patch adds at most one entry to L, so $|L|$ grows from 1 to $q_{\mathrm{RO}} + q_{\mathrm{Snd}}$. Using the Gaussian sum we get

$$q_{\mathrm{DDH}} \leq \sum_{i=1}^{q_{\mathrm{RO}}+q_{\mathrm{Snd}}} 3 \cdot (i-1) = 3 \cdot \frac{(q_{\mathrm{RO}} + q_{\mathrm{Snd}} - 1) \cdot (q_{\mathrm{RO}} + q_{\mathrm{Snd}})}{2} \leq 1.5(q_{\mathrm{RO}} + q_{\mathrm{Snd}})^2.$$

Unless $\mathcal{A}$ queries the random oracle on $\mathsf{CDH}(ltpk^{\mathsf{DH}}_{U^*}, sspk^{\mathsf{DH}}_{V^*})$ in the DH_1 position, $\mathcal{B}_3$ provides a perfect simulation of game $\mathcal{G}_{\mathrm{A.4}}$. If $\mathcal{A}$ makes such a random oracle query, then $\mathcal{B}_3$ will detect this and win its GapDH game. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.4}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \epsilon_{\mathsf{GDH}} + \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.5}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game A.6 (Replacing the session key). We now replace the session key of the test session π^* with a uniformly sampled key. Since Game $\mathcal{G}_{\mathrm{A.5}}$ has ruled out that the adversary queries the RO on the master secret $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$ of the test session π^*, the adversary has no chance of detecting this change. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.5}}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_{\mathrm{A.6}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

To conclude this proof case, observe that in Game $\mathcal{G}_{\mathrm{A.6}}$ the session key of the test session π^* is now a uniformly random key, independent of b_{test}. Furthermore, $\mathcal{A}$ cannot reveal the session key of the test session π^* via a REVEALSESSKEY query on π^* or any partnered session which might hold the same key. Finally, any non-partnered session will derive a different session key: any difference in the identities A, B yields different DH public keys by Game $\mathcal{G}_1$, any difference in the DH shares implies a difference in the $\mathsf{DH}[1]$–$\mathsf{DH}[4]$ inputs, while differing KEM public keys or ciphertexts yield a different ss input by Game $\mathcal{G}_3$. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{A.6}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq 0.$$

[13]If PQXDH would include the session context (e.g., the values in the session identifier) in the key derivation, as is good practice, the reduction would need less DDH oracle queries: Using the context, the reduction can detect when a challenge DH key is combined with a maliciously generated (ephemeral) key. In the modified SEND oracle in Figure 8, only the first PoS in line 24 remains a pair of public keys, while the other three occurrences are solved via context. In consequence, at maximum PoS per entry in L contains two public keys and we get a maximum of $q_{\mathrm{RO}} + q_{\mathrm{Snd}}$ DDH queries.

Case B **($\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$).**

In this proof case, the $\mathsf{clean}_{\mathsf{ELT}}$ predicate ensures for the test session π^* that

1. the initiator's randomness is not revealed, and
2. the responder's long-term key is uncompromised.

Similar to the classical Signal proof [CCD+17], the uncompromised ephemeral/long-term Diffie–Hellman combination ensures key indistinguishability for the test session.

Game B.0. This case begins with Game $\mathcal{G}_3$ conditioned on $\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{B.0}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_3[\mathsf{clean}_{\mathsf{ELT}}(\pi^*)]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game B.1 (Guess initiator session). We guess the initiator session π^*_{i} contributing to the test session π^* (i.e., either the test session itself if it is an initiator session, or the initiator session partnered to the test session), overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of sessions n_s:

$$\mathsf{Adv}^{\mathcal{G}_{B.0}}_{\mathsf{PQXDH}}(\mathcal{A}) \le n_s \cdot \mathsf{Adv}^{\mathcal{G}_{B.1}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game B.2 (Guess responder identity V^*). We guess the identity V^* of the responder involved in the test session, overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_{B.1}}_{\mathsf{PQXDH}}(\mathcal{A}) \le n_p \cdot \mathsf{Adv}^{\mathcal{G}_{B.2}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game B.3 (GapDH). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the adversary queries the random oracle on a value formatted like a master secret and beginning with $\mathsf{CDH}(epk^{\mathsf{DH}}, ltpk^{\mathsf{DH}}_{V^*})$, i.e., the shared DH key between the ephemeral key of the test session's initiator epk^{DH} and $ltpk^{\mathsf{DH}}_{V^*}$. The probability of such an abort can be bounded by the advantage of the following reduction $\mathcal{B}_4$ against the $(t, \epsilon_{\mathsf{GDH}}, q_{\mathrm{DDH}})$-hardness of the GapDH problem in $(\mathbb{G}, g, q)$.

The reduction follows the idea of Game $\mathcal{G}_{A.5}$, only that now we embed the GapDH challenge keys as the ephemeral key in the initiator session π^*_{i} and as the long-term key of V^*. The reduction does not need to answer corresponding RevealRand queries on π^*_{i} or CorruptLTKey queries on V^* due to $\mathsf{clean}_{\mathsf{ELT}}$. It can use a similar strategy to the one in Game $\mathcal{G}_{A.5}$ to patch the random oracle, yet it has an easier time ensuring consistency of the Send oracle: It can always compute DH_1 and DH_3 by using the responder's semi-static secret key, and DH_4 (if it is a full handshake) with the honest party's ephemeral secret key. Outside of the test session $\mathcal{B}_4$ can use the initiator's ephemeral secret key to compute DH_2 if the ephemeral key was honestly generated. Hence, $\mathcal{B}_4$ can compute all inputs to the KDF for all Send queries itself, except for the Send queries to V^* with a malicious ephemeral key on the initiator side. In consequence, $\mathcal{B}_4$ queries the DDH oracle at most once per Send query and once when checking RO queries for the DH_2 position. All in all, $\mathcal{B}_4$ needs a maximum of $q_{\mathrm{Snd}} + q_{\mathrm{RO}}$ queries to the DDH oracle, which is well below the number in Game $\mathcal{G}_{A.5}$.

Unless $\mathcal{A}$ queries the random oracle on $\mathsf{CDH}(epk^{\mathsf{DH}}, ltpk^{\mathsf{DH}}_{V^*})$ in the DH_2 position, $\mathcal{B}_4$ provides a perfect simulation of $\mathcal{G}_{B.2}$. If $\mathcal{A}$ does make such a random oracle query, then $\mathcal{B}_4$ will detect this and win its GapDH game. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{B.2}}_{\mathsf{PQXDH}}(\mathcal{A}) \le \epsilon_{\mathsf{GDH}} + \mathsf{Adv}^{\mathcal{G}_{B.3}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game B.4 (Replacing the session key). We now replace the session key of the test session π^* with a uniformly sampled key. Since Game $\mathcal{G}_{B.3}$ has ruled out that the adversary queries the RO on the master secret $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$ of the test session π^*, the adversary has no chance of detecting this change. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{B.3}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_{B.4}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

To conclude this proof case, observe that in Game $\mathcal{G}_{B.4}$ the session key of the test session π^* is now a uniformly random key, independent of b_{test}. As before, $\mathcal{A}$ can neither reveal the session key of the test session π^* nor of any partnered session, nor does any other session derive the same session key by Game $\mathcal{G}_3$. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

$$\mathsf{Adv}^{\mathcal{G}_{B.4}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq 0.$$

Case C ($\mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type} = \mathsf{full}$).

In this proof case, the test session performs a full handshake and the $\mathsf{clean}_{\mathsf{ESS}}$ predicate ensures for the test session π^* that

1. the initiator's randomness is not revealed, and
2. the responder's semi-static key is uncompromised, and
3. if π^* is an initiator, the responder's long-term (signing) key was uncompromised upon acceptance.

Via the last point, similarly to Case A, we can guarantee that initiator and responder indeed agree on the semi-static key (identifier), given signatures are unforgeable. Then, similar to the classical Signal proof [CCD+17], the uncompromised ephemeral/semi-static Diffie–Hellman combination ensures key indistinguishability for the test session.

Game C.0. This case begins with Game $\mathcal{G}_3$ conditioned on $\mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type} = \mathsf{full}$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{C.0}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_3[\mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{full}]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game C.1 (Guess responder identity V^*). We first guess the identity V^* of the responder to the test session, overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_{C.0}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_p \cdot \mathsf{Adv}^{\mathcal{G}_{C.1}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game C.2 (Signature unforgeability). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the test session π^* is an initiator session and accepts using a semi-static DH public key $sspk^{\mathsf{DH}}_{V^*}$ that was not generated through a KGenSS run for V^*. This ensures that the test session accepts with $\pi^*.\mathsf{sspks} = (sspk^{\mathsf{DH}}_{V^*}, \cdot)$ corresponding to a DH key pair of which the adversary does not know the secret key, since $\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$ guarantees that the semi-static secret belong to $sspk^{\mathsf{DH}}_{V^*}$ is not compromised. Similar to Game $\mathcal{G}_{A.2}$, the probability of such an abort can be bounded by the advantage of a reduction against the $(t, \epsilon_{\mathsf{SIG}}, 2n_{ss} + n_s)$-unforgeability of SIG. The argument is identical to the Game $\mathcal{G}_{A.2}$. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{C.1}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \epsilon_{\mathsf{SIG}} + \mathsf{Adv}^{\mathcal{G}_{C.2}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game C.3 (Guess semi-static key identifier ssid^* of V^*). We now guess the identifier ssid of the responder V^*'s (uncorrupted) semi-static key $sspk_{V^*}^{\mathsf{DH}}$. Note that depending on the role of π^* this is either the test session's own key (if $\pi^*.\mathsf{role} = \mathsf{responder}$), or of the intended peer (if $\pi^*.\mathsf{role} = \mathsf{initiator}$). We denote the guessed identifier by ssid^*, and abort, setting the adversary's output bit to 0, if this guess is incorrect, losing at most a factor n_{ss} of the number of semi-static keys per user:

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.2}}(\mathcal{A}) \leq n_{ss} \cdot \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.3}}(\mathcal{A}).$$

Game C.4 (Guess initiator session). We guess the initiator session π_{i}^* involved in the test session π^* (i.e., either the test session itself if it is an initiator session, or the initiator session partnered to the test session), overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of sessions n_s:

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.3}}(\mathcal{A}) \leq n_s \cdot \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.4}}(\mathcal{A}).$$

Game C.5 (GapDH). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the adversary queries the random oracle on a value formatted like a master secret, beginning with $\mathsf{CDH}(epk^{\mathsf{DH}}, sspk_{V^*}^{\mathsf{DH}})$, i.e., the shared DH key between the ephemeral key of the test session's initiator epk^{DH} and $sspk_{V^*}^{\mathsf{DH}}$, where $sspk_{V^*}^{\mathsf{DH}}$ is the semi-static DH key with identifier ssid^*. The probability of such an abort can be bounded by the advantage of the following reduction $\mathcal{B}_5$ against the $(t, \epsilon_{\mathsf{GDH}}, q_{\mathrm{DDH}})$-hardness of the GapDH problem in $(\mathbb{G}, g, q)$.

The proof follows the idea of Games $\mathcal{G}_{A.5}$ and $\mathcal{G}_{B.3}$. Though, now we embed the GapDH challenge keys as the ephemeral key in the initiator session π_{i}^* and as semi-static key of V^* with semi-static id ssid^*. The reduction does not need to answer corresponding RevealRand queries on π_{i}^* or CorruptSSKey(V^*, ssid^*) queries due to $\mathsf{clean}_{\mathsf{ESS}}$. It can use a similar strategy to patch the random oracle: It can always compute DH_1 and DH_2 by using the initiator's and responder's long-term secret key, respectively, and DH_4 with the honest party's ephemeral secret key. Outside of the test session $\mathcal{B}_5$ can use the initiator's ephemeral secret key to compute DH_3 if the ephemeral key was honestly generated. Hence, $\mathcal{B}_5$ can compute all inputs to the KDF for all Send queries itself, except for the Send queries to the test session or to V^* with ssid^* and a malicious ephemeral key on the initiator side. In consequence, $\mathcal{B}_5$ queries the DDH oracle at most once per Send query and once when checking RO queries for the DH_3 position. All in all, $\mathcal{B}_5$ needs a maximum of $q_{\mathrm{Snd}} + q_{\mathrm{RO}}$ queries to the DDH oracle, which is well below the number in Game $\mathcal{G}_{A.5}$.

Unless $\mathcal{A}$ queries the random oracle on $\mathsf{CDH}(epk^{\mathsf{DH}}, ltpk_{V^*}^{\mathsf{DH}})$ in the DH_2 position, $\mathcal{B}_5$ provides a perfect simulation of $\mathcal{G}_{C.4}$. If $\mathcal{A}$ does make such a random oracle query, then $\mathcal{B}_5$ will detect this and win its GapDH game. Thus,

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.4}}(\mathcal{A}) \leq \epsilon_{\mathsf{GDH}} + \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.5}}(\mathcal{A}).$$

Game C.6 (Replacing the session key). We now replace the session key of the test session π^* with a uniformly sampled key. Since Game $\mathcal{G}_{C.5}$ has ruled out that the adversary queries the RO on the master secret $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$ of the test session π^*, the adversary has no chance of detecting this change. Thus,

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.5}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.6}}(\mathcal{A}).$$

To conclude this proof case, observe that in Game $\mathcal{G}_{C.6}$ the session key of the test session π^* is now a uniformly random key, independent of b_{test}. As before, $\mathcal{A}$ can neither reveal the session key of the test session π^* nor of any partnered session, nor does any other session derive the same session key by Game $\mathcal{G}_3$. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{C.6}}(\mathcal{A}) \leq 0.$$

Case D ($\mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type} = \mathsf{reduced}$).

In this proof case, the test session performs a reduced handshake and the $\mathsf{clean}_{\mathsf{ESS}}$ predicate ensures for the test session π^* that

1. the initiator's randomness is not revealed, and
2. the responder's semi-static key is uncompromised, and
3. if π^* is an initiator, the responder's long-term (signing) key was uncompromised upon acceptance.

Via the last point, we can guarantee that initiator and responder indeed agree on the semi-static key (identifier), given signatures are unforgeable. The difference to Case C is that, in a reduced handshake, the semi-static KEM public key of the responder is used (and there is no ephemeral/ephemeral Diffie–Hellman combination). This means we get the following *hybrid* guarantee: both the ephemeral/semi-static Diffie–Hellman combination being uncompromised *and* the semi-static KEM key being uncompromised are, on their own, sufficient to ensure key indistinguishability for the test session.

Game D.0. This case begins with Game $\mathcal{G}_3$ conditioned on $\mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type} = \mathsf{reduced}$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{D.0}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_3[\mathsf{clean}_{\mathsf{ESS}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{reduced}]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game D.1 (Guess responder identity V^*). We first guess the identity V^* of the responder to the test session, overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_{D.0}}_{\mathsf{PQXDH}}(\mathcal{A}) \le n_p \cdot \mathsf{Adv}^{\mathcal{G}_{D.1}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game D.2 (Signature unforgeability). In this game case, we are interested in the authenticity of *both* semi-static DH and KEM keys, and hence now abort the game (again, returning 0 as the adversary's bit guess) in the event that the test session π^* is an initiator session and accepts using a semi-static DH public key $sspk^{\mathsf{DH}}_{V^*}$ *or* a semi-static KEM public key $sspk^{\mathsf{KEM}}_{V^*}$ that was not generated through a KGenSS run for V^*. Since for reduced handshakes both public keys are recorded in $\pi^*.\mathsf{sspks} = (sspk^{\mathsf{DH}}_{V^*}, sspk^{\mathsf{KEM}}_{V^*})$, this ensures that the test session accepts with semi-static DH and KEM public keys of which the adversary does not know the secret, since $\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$ guarantees that the semi-static secrets corresponding to $sspk^{\mathsf{DH}}_{V^*}$ or to $sspk^{\mathsf{KEM}}_{V^*}$ are not compromised.

Again similar to Game $\mathcal{G}_{A.2}$, we can reduce this hop to the $(t, \epsilon_{\mathsf{SIG}}, 2n_{ss} + n_s)$-unforgeability of SIG:

$$\mathsf{Adv}^{\mathcal{G}_{D.1}}_{\mathsf{PQXDH}}(\mathcal{A}) \le \epsilon_{\mathsf{SIG}} + \mathsf{Adv}^{\mathcal{G}_{D.2}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game D.3 (Guess semi-static key identifiers ssid^*_1, ssid^*_2 of V^*). We now guess the ssid identifiers of the responder V^*'s (uncorrupted) semi-static DH and KEM keys, $sspk^{\mathsf{DH}}_{V^*}$ resp. $sspk^{\mathsf{KEM}}_{V^*}$. Note that depending on the role of π^* this is either the test session's own key (if $\pi^*.\mathsf{role} = \mathsf{responder}$), or of the intended peer (if $\pi^*.\mathsf{role} = \mathsf{initiator}$). We denote the guessed identifiers by ssid^*_1 (for the DH key) and ssid^*_2 (for the KEM key), and abort, setting the adversary's output bit to 0, if this guess is incorrect, losing at most a factor n_{ss} of the number of semi-static keys of each type per user, for each of the two guesses:

$$\mathsf{Adv}^{\mathcal{G}_{D.2}}_{\mathsf{PQXDH}}(\mathcal{A}) \le n_{ss}^2 \cdot \mathsf{Adv}^{\mathcal{G}_{D.3}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game D.4 (Guess initiator session). We guess the initiator session π^*_{i} involved in the test session π^* (i.e., either the test session itself if it is an initiator session, or the initiator session partnered to the test session), overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of sessions n_s:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{D.3}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_s \cdot \mathsf{Adv}^{\mathcal{G}_{\mathrm{D.4}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game D.5 (GapDH + OW-CCA). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the adversary queries the random oracle on a value formatted like a master secret, beginning with $\mathsf{CDH}(epk^{\mathsf{DH}}, sspk^{\mathsf{DH}}_{V^*})$, i.e., the shared DH key between the ephemeral key of the test session's initiator epk^{DH} and $sspk^{\mathsf{DH}}_{V^*}$, and ending with the KEM shared secret ss^* resulting from π^*_{i} encapsulating against $sspk^{\mathsf{KEM}}_{V^*}$, where $sspk^{\mathsf{DH}}_{V^*}$ and $sspk^{\mathsf{KEM}}_{V^*}$ are the semi-static DH and KEM keys in $sspk^{\mathsf{ssid}^*}_{V^*}$. The probability of such an abort can be bounded by the *minimum* of the advantages of the following reduction $\mathcal{B}_6$ playing *simultaneously* against the $(t, \epsilon_{\mathsf{GDH}}, q_{\mathrm{DDH}})$-hardness of the GapDH problem in $(\mathbb{G}, g, q)$ and $(t, \epsilon_{\mathsf{CCA}}, n_s)$–$\mathsf{OW}\text{-}\mathsf{CCA}$ security of KEM.

The reduction embeds the GapDH challenge as the ephemeral DH share in the initiator session π^*_{i} and as semi-static DH key of V^* with semi-static id ssid^*_1 (as in Game $\mathcal{G}_{\mathrm{C.5}}$), and the KEM public key pk_{KEM} from the OW-CCA game as the semi-static key $sspk^{\mathsf{KEM}}_{V^*}$ of V^* with semi-static id ssid^*_2. Furthermore, it uses the KEM challenge ciphertext ct^* in π^*_{i}. Similar to Game $\mathcal{G}_{\mathrm{C.5}}$, the reduction does not need to answer corresponding RevealRand on π^*_{i} or CorruptSSKey$(V^*, \mathsf{ssid}^*_1/\mathsf{ssid}^*_2)$ queries due to $\mathsf{clean}_{\mathsf{ESS}}$. Furthermore, as for Game $\mathcal{G}_{\mathrm{C.5}}$, patching the Send oracle is relatively easy: Except for queries to the test session or to V^* using $sspk^{\mathsf{DH}}_{V^*}$ and a malicious ephemeral DH key on the initiator side, the reduction can compute all DH shared secrets with the DH secret keys and it can decapsulate the KEM shared secret with the KEM secret key or, for sessions using $sspk^{\mathsf{KEM}}_{V^*}$, the Decaps oracle of the OW-CCA game. Otherwise (and in particular in the test session), the reduction uses a freshly sampled key as session key. If the DDH oracle accepts a query, then the reduction returns the DH_3 and ss^* from the corresponding RO query to its GapDH and OW-CCA games after finishing the simulation for $\mathcal{A}$.

The reduction ensures consistency with the random oracle by checking RO queries against the above mentioned patches. In consequence, the remaining queries to the DDH oracle are once per Send query and when checking RO queries for the DH_3 position. All in all, $\mathcal{B}_6$ needs a maximum of $q_{\mathrm{Snd}} + q_{\mathrm{RO}}$ queries to the DDH oracle, which is well below the number in $\mathcal{G}_{\mathrm{A.5}}$; it further makes at most n_s many Decaps query in the OW-CCA game.

Unless $\mathcal{A}$ queries the random oracle on $\mathsf{CDH}(epk^{\mathsf{DH}}, sspk^{\mathsf{DH}}_{V^*})$ in the DH_3 position *and* ss^* in the ss position, $\mathcal{B}_6$ provides a perfect simulation of $\mathcal{G}_{\mathrm{D.4}}$. If $\mathcal{A}$ does make such a random oracle query, then $\mathcal{B}_6$ will detect this and win *both* its GapDH game and its OW-CCA game. We can thus bound the game hop by the *minimum* advantage against the two games:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{D.4}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \min(\epsilon_{\mathsf{GDH}}, \epsilon_{\mathsf{CCA}}) + \mathsf{Adv}^{\mathcal{G}_{\mathrm{D.5}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game D.6 (Replacing the session key). We now replace the session key of the test session π^* with a uniformly sampled key. Since Game $\mathcal{G}_{\mathrm{D.5}}$ has ruled out that the adversary queries the RO on the master secret $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$ of the test session π^*, the adversary has no chance of detecting this change. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{D.5}}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_{\mathrm{D.6}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

To conclude this proof case, observe that in Game $\mathcal{G}_{\mathrm{D.6}}$ the session key of the test session π^* is now a uniformly random key, independent of b_{test}. As before, $\mathcal{A}$ can neither reveal the session key of the test session π^* nor of any partnered session, nor does any other session derive the same session key by Game $\mathcal{G}_3$. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{D.6}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq 0.$$

Case E ($\mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{peerE}}(\pi^*) \wedge \mathsf{type}(\pi^*) = \mathsf{full}$).

In this proof case, the test session performs a full handshake and the $\mathsf{clean}_{\mathsf{EE}}$ predicate and its $\mathsf{clean}_{\mathsf{peerE}}$ sub-predicate ensure for the test session π^* that

1. the test session's owner's randomness is not revealed, and
2. the randomness of the session (contributively) partnered to the test session is not revealed.

Similar to Case D, we get a *hybrid* guarantee here: both the ephemeral/ephemeral Diffie–Hellman combination being uncompromised *and* the ephemeral KEM key being uncompromised are, on their own, sufficient to ensure key indistinguishability for the test session.

Game E.0. This case begins with Game $\mathcal{G}_3$ conditioned on $\mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{peerE}}(\pi^*) \wedge \pi^*.\mathsf{type} = \mathsf{full}$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.0}}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_3[\mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{peerE}}(\pi^*) \wedge \pi^*.\mathsf{type}=\mathsf{full}]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game E.1 (Guess initiator and responder sessions). We guess the initiator and responder sessions π^*_{i} resp. π^*_{r} involved in the test session π^* (i.e., both the test session itself and its sid- resp. cid-partner), overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of sessions n_s squared:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.0}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_s^2 \cdot \mathsf{Adv}^{\mathcal{G}_{\mathrm{E.1}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game E.2 (GapDH + OW-CCA). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the adversary queries the random oracle on a value formatted like a master secret, beginning with $\mathsf{CDH}(epk^{\mathsf{DH}}_{U^*}, epk^{\mathsf{DH}}_{V^*})$, i.e., the shared DH key between the ephemeral keys $epk^{\mathsf{DH}}_{U^*}$ and $epk^{\mathsf{DH}}_{V^*}$ of the guessed initiator session π^*_{i}, resp. responder session π^*_{r}, and ending with the KEM shared secret ss^* resulting from π^*_{i} encapsulating against $epk^{\mathsf{KEM}}_{V^*}$. The probability of such an abort can be bounded by the *minimum* of the advantages of the following reduction $\mathcal{B}_7$ playing *simultaneously* against the $(t, \epsilon_{\mathsf{GDH}}, q_{\mathrm{DDH}})$-hardness of the GapDH problem in $(\mathbb{G}, g, q)$ and $(t, \epsilon_{\mathsf{CCA}}, n_s)$–OW-CCA security of KEM.

The reduction embeds the GapDH challenge as the ephemeral keys in the sessions π^*_{i} and π^*_{r}, and the KEM public key pk_{KEM} from the OW-CCA game as ephemeral key $epk^{\mathsf{KEM}}_{V^*}$ in π^*_{r}. It uses the KEM challenge ciphertext ct^* in π^*_{i}. Similar to Game $\mathcal{G}_{\mathrm{D.5}}$, the reduction does not need to answer corresponding RevealRand queries on π^*_{i} or π^*_{r} due to $\mathsf{clean}_{\mathsf{EE}} \wedge \mathsf{clean}_{\mathsf{peerE}}$. Furthermore, as for Game $\mathcal{G}_{\mathrm{D.5}}$, patching the Send oracle is relatively easy: Except for queries to the test session or to π^*_{r} with a modified initiator ephemeral key,[14] the reduction can compute all DH shared secrets itself. For the latter query, $\mathcal{B}_7$ uses a single call to the DDH oracle of GapDH resp. Decaps oracle of OW-CCA to simulate. In the test session, the reduction uses a freshly sampled key as session key. If the DDH oracle accepts a query, then the

[14]This can happen if the test session is a responder session and the adversary mauls the initiator public key on the wire.

reduction returns the DH_3 and ss from the corresponding RO query to its GapDH and OW-CCA games after finishing the simulation for $\mathcal{A}$.

The reduction ensures consistency with the random oracle by checking RO queries against the above mentioned patches. It queries the DDH oracle possibly once for π^*_{r} and when checking RO queries for the DH_4 position, so overall a maximum of $q_{\mathrm{RO}} + 1$ queries, which is well below the number in Game $\mathcal{G}_{\mathrm{A.5}}$. It further makes at most one Decaps query in the OW-CCA game.

Unless $\mathcal{A}$ queries the random oracle on $\mathsf{CDH}(epk^{\mathsf{DH}}_{U^*}, epk^{\mathsf{DH}}_{V^*})$ in the DH_4 position and ss^* in the ss position, $\mathcal{B}_7$ provides a perfect simulation of $\mathcal{G}_{\mathrm{E.1}}$. If $\mathcal{A}$ does make such a random oracle query, then $\mathcal{B}_7$ will detect this and win *both* its GapDH game and its OW-CCA game. We can thus bound the game hop by the *minimum* advantage against the two games:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.1}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq \min(\epsilon_{\mathsf{GDH}}, \epsilon_{\mathsf{CCA}}) + \mathsf{Adv}^{\mathcal{G}_{\mathrm{E.2}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game E.3 (Replacing the session key). We now replace the session key of the test session π^* with a uniformly sampled key. Since Game $\mathcal{G}_{\mathrm{E.2}}$ has ruled out that the adversary queries the RO on the master secret $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$ of the test session π^*, the adversary has no chance of detecting this change. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.2}}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_{\mathrm{E.3}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

To conclude this proof case, observe that in Game $\mathcal{G}_{\mathrm{E.3}}$ the session key of the test session π^* is now a uniformly random key, independent of b_{test}. As before, $\mathcal{A}$ can neither reveal the session key of the test session π^* nor of any partnered session, nor does any other session derive the same session key by Game $\mathcal{G}_3$. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{E.3}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq 0.$$

Case F ($\mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{sigE}}(\pi^*) \wedge \mathsf{type}(\pi^*) = \mathsf{full}$).

In this proof case, the test session performs a full handshake and the $\mathsf{clean}_{\mathsf{EE}}$ predicate and its $\mathsf{clean}_{\mathsf{sigE}}$ sub-predicate ensure for the test session π^* that

1. the test session's owner's randomness is not revealed, and
2. if π^* is an initiator, the responder's long-term (signing) key was uncompromised upon acceptance and a potential responder partner session's randomness was not revealed.

Via the second point, we can guarantee that initiator and responder agree on the ephemeral KEM key, given signatures are unforgeable. The KEM encapsulation then ensures key indistinguishability for the test session.

Game F.0. This case begins with Game $\mathcal{G}_3$ conditioned on $\mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{sigE}}(\pi^*) \wedge \mathsf{type}(\pi^*) = \mathsf{full}$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{F.0}}}_{\mathsf{PQXDH}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_3[\mathsf{clean}_{\mathsf{EE}}(\pi^*) \wedge \mathsf{clean}_{\mathsf{sigE}}(\pi^*) \wedge \mathsf{type}(\pi^*)=\mathsf{full}]}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game F.1 (Guess responder identity V^*). We first guess the identity V^* of the responder to the test session, overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_{\mathrm{F.0}}}_{\mathsf{PQXDH}}(\mathcal{A}) \leq n_p \cdot \mathsf{Adv}^{\mathcal{G}_{\mathrm{F.1}}}_{\mathsf{PQXDH}}(\mathcal{A}).$$

Game F.2 (Signature unforgeability). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the test session π^* is an initiator session and accepts using an ephemeral KEM public key $epk_{V^*}^{\mathsf{KEM}}$ that was not generated by V^*. This ensures that the test session accepts with an ephemeral KEM public key of which the adversary does not know the secret key, since the adversary cannot reveal V^*'s session randomness. Similar to Game $\mathcal{G}_{A.2}$, the probability of such an abort can be bounded by the advantage of a reduction against the $(t, \epsilon_{\mathsf{SIG}}, 2n_{ss} + n_s)$-unforgeability of SIG. The proof is identical to the one for Game $\mathcal{G}_{A.2}$. Thus,

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.1}}(\mathcal{A}) \leq \epsilon_{\mathsf{SIG}} + \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.2}}(\mathcal{A}).$$

Game F.3 (Guess initiator and responder sessions). We guess the initiator and responder sessions π_{i}^* resp. π_{r}^* involved in the test session π^* (i.e., both the test session itself and its partner), overwriting the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of sessions n_s squared:

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.2}}(\mathcal{A}) \leq n_s^2 \cdot \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.3}}(\mathcal{A}).$$

Game F.4 (OW-CCA). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the adversary queries the random oracle on a value formatted like a master secret and ending with the KEM shared secret ss^* resulting from π_{i}^* encapsulating against $epk_{V^*}^{\mathsf{KEM}}$. The probability of such an abort can be bounded by the advantage of the following reduction $\mathcal{B}_8$ playing against $(t, \epsilon_{\mathsf{CCA}}, n_s)$–OW-CCA security of KEM.

The reduction embeds the challenge pk_{KEM}, ct^* from the OW-CCA game as ephemeral key $epk_{V^*}^{\mathsf{KEM}}$ of V^*, and as KEM ciphertext in π_{r}^*. The reduction does not need to answer a RevealRand query on π_{r}^* due to $\mathsf{clean}_{\mathsf{EE}}$. When $\mathcal{A}$ halts, the reduction guesses $i \in [q_{\mathrm{RO}}]$ and returns the ss position of the ith RO query (assuming this query is formatted like a master secret) as the target shared secret.

Unless $\mathcal{A}$ queries the random oracle on ss^* in the ss position, $\mathcal{B}_8$ provides a perfect simulation of Game $\mathcal{G}_{F.3}$. If $\mathcal{A}$ does make such a random oracle query, then $\mathcal{B}_8$ wins its OW-CCA game if it guesses the ith RO query correctly. We can thus bound the game hop by the advantage against OW-CCA times a loss of q_{RO}:

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.3}}(\mathcal{A}) \leq q_{\mathrm{RO}} \cdot \epsilon_{\mathsf{CCA}} + \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.4}}(\mathcal{A}).$$

Game F.5 (Replacing the session key). We now replace the session key of the test session π^* with a uniformly sampled key. Since Game $\mathcal{G}_{F.4}$ has ruled out that the adversary queries the RO on the master secret $DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$ of the test session π^*, the adversary has no chance of detecting this change. Thus,

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.4}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.5}}(\mathcal{A}).$$

To conclude this proof case, observe that in Game $\mathcal{G}_{F.5}$ the session key of the test session π^* is now a uniformly random key, independent of b_{test}. As before, $\mathcal{A}$ can neither reveal the session key of the test session π^* nor of any partnered session, nor does any other session derive the same session key by Game $\mathcal{G}_3$. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

$$\mathsf{Adv}_{\mathsf{PQXDH}}^{\mathcal{G}_{F.5}}(\mathcal{A}) \leq 0.$$

Collecting the bounds from all cases yields the overall theorem bound. $\square$

6 Discussion and Conclusion

In this work, we provided a reductionist security analyis of Signal's PQXDH in a "maximum-exposure", game-based security model, augmented compared to prior versions [CCD+17, BFG+22, BFG+21] to capture the KEM component but also key signing, and gave a fully-parameterized, concrete security bound for PQXDH.

Our bound relies (among other things) on the $LEAK^{+}$-BIND-SS-$\{CT, PK\}$ binding property of the KEM. While we show that this property is satisfied by both Kyber and ML-KEM, the current and future KEMs in PQXDH may not satisfy this property; so achieving PQXDH-like security without relying on a binding property is of general interest. We can indeed forgo this assumption (and the corresponding advantage term), if in the key derivation of PQXDH, we include the KEM public key and ciphertext (or, ideally, the whole session context as is good practice). This supports and complements a proposal discussed in the tool-based formal verification of PQXDH by BJKS [BJK23, BJKS23], to make sure the key agreement in the initial handshake does not rely on the follow-up AEAD encryption (which would indeed violate key indistinguishability as we prove it). As a side note, including context into the key derivation also supports tighter security proofs [CCG+19].

Our reduction for PQXDH has an additional case (compared to the X3DH analysis [CCD+17]), where key indistinguishability hinges solely on the signed ephemeral KEM key. This guarantees security against an *active* adversary who later gets quantum powers. Hence, PQXDH protects against an even stronger class of attacks than the "harvest now, decrypt later" attack which motivated the design.

Like prior Signal analyses [CCD+17, BJKS23, BJK23], we model the long-term DH and signing keys as separate ones; we leave it to future work to more accurately reflect the implementation which re-uses the same keys for both purposes.

Acknowledgments

We thank Ehren Kret and Rolfe Schmidt for discussions on the PQXDH protocol, and Franziskus Kiefer for insights on the KEM re-encapsulation attack. R.F. was supported by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

[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. (Cited on page 3.)

[BC] Bruno Blanchet and Vincent Cheval. Proverif: Cryptographic protocol verifier in the formal model. https://bblanche.gitlabpages.inria.fr/proverif/. (Cited on page 2.)

[BCD+24] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karolin Varner, and Bas Westerbaan. X-wing. *IACR Communications in Cryptology*, 1(1), 2024. (Cited on page 3.)

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. (Cited on page 5.)

[BFG$^+$20] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. Towards post-quantum security for Signal's X3DH handshake. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 404–430, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Heidelberg, Germany. (Cited on page 3.)

[BFG$^+$21] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. Cryptology ePrint Archive, Report 2021/769, 2021. `https://eprint.iacr.org/2021/769`. (Cited on pages 2, 15, 16, 17, 18, 19, and 37.)

[BFG$^+$22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 3–34, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany. (Cited on pages 2, 3, 15, 16, 17, 18, 19, and 37.)

[BJK23] Karthikean Barghavan, Charlie Jacomme, and Franziskus Kiefer. Formal analysis of the PQXDH protocol, 2023. `https://github.com/Inria-Prosecco/pqxdh-analysis`. (Cited on pages 2, 3, and 37.)

[BJKS23] Karthikean Barghavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. An analysis of Signal's PQXDH, October 2023. `https://cryspen.com/post/pqxdh/`. (Cited on pages 2, 3, and 37.)

[Bla] Bruno Blanchet. Cryptoverif: Cryptographic protocol verifier in the computational model. `https://bblanche.gitlabpages.inria.fr/CryptoVerif/`. (Cited on page 2.)

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany. (Cited on page 15.)

[CCD$^+$17] Katriel Cohn-Gordon, Cas J. F. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. In *IEEE European Symposium on Security and Privacy, EuroS&P 2017*, pages 451–466, 2017. (Cited on pages 1, 2, 5, 15, 17, 18, 21, 22, 25, 29, 30, and 37.)

[CCG$^+$19] Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 767–797, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. (Cited on page 37.)

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. (Cited on page 3.)

[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. Version 1.0.6 (April 3, 2024), https://eprint.iacr.org/2023/1933. (Cited on pages 3, 7, and 8.)

[CHDN$^+$24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. K-Waay: Fast and deniable Post-Quantum X3DH without ring signatures. In *33rd USENIX Security Symposium (USENIX Security 24)*, Philadelphia, PA, August 2024. USENIX Association. (Cited on page 3.)

[DFGS15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1197–1210, Denver, CO, USA, October 12–16, 2015. ACM Press. (Cited on page 16.)

[DG22] Samuel Dobson and Steven D. Galbraith. Post-quantum signal key agreement from SIDH. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 422–450. Springer, 2022. (Cited on page 3.)

[FG14] Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 1193–1204, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. (Cited on page 15.)

[FJ24] Rune Fiedler and Christian Janson. A deniability analysis of Signal's initial handshake PQXDH. *Proc. Priv. Enhancing Technol.*, 2024(4), 2024. (Cited on page 2.)

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. (Cited on page 10.)

[HKKP22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for Signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. *Journal of Cryptology*, 35(3):17, July 2022. (Cited on page 3.)

[JK18] Tibor Jager and Rafael Kurek. Short digital signatures and ID-KEMs via truncation collision resistance. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 221–250, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. (Cited on page 5.)

[JKN21] Tibor Jager, Rafael Kurek, and David Niehues. Efficient adaptively-secure IB-KEMs and VRFs via near-collision resistance. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 596–626, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. (Cited on page 5.)

[KS23] Ehren Kret and Rolfe Schmidt. The PQXDH key agreement protocol, September 2023. Revision 1, https://signal.org/docs/specifications/pqxdh/. (Cited on pages 1 and 2.)

[KS24] Ehren Kret and Rolfe Schmidt. The PQXDH key agreement protocol, January 2024. Revision 3, https://signal.org/docs/specifications/pqxdh/. (Cited on pages 1, 2, 3, and 21.)

[LT11] Gaëtan Leurent and Søren S. Thomsen. Practical near-collisions on the compression function of BMW. In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 238–251, Lyngby, Denmark, February 13–16, 2011. Springer, Heidelberg, Germany. (Cited on page 5.)

[MMP$^+$23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. (Cited on page 3.)

[MP16] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol, November 2016. https://signal.org/docs/specifications/x3dh/. (Cited on page 1.)

[MvV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997. (Cited on page 5.)

[NIS23] NIST. Module-lattice-based key-encapsulation mechanism standard, August 2023. FIPS 203 (draft). https://doi.org/10.6028/NIST.FIPS.203.ipd. (Cited on pages 3, 6, 7, and 10.)

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany. (Cited on page 5.)

[Per16] Trevor Perrin. The XEdDSA and VXEdDSA signature schemes, October 2016. https://signal.org/docs/specifications/xeddsa/. (Cited on page 21.)

[PS14] Inna Polak and Adi Shamir. Using random error correcting codes in near-collision attacks on generic hash-functions. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014: 15th International Conference in Cryptology in India*, volume 8885 of *Lecture Notes in Computer Science*, pages 219–236, New Delhi, India, December 14–17, 2014. Springer, Heidelberg, Germany. (Cited on page 5.)

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. (Cited on page 3.)

[SAB$^+$] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-Kyber. https://pq-crystals.org/kyber/. (Cited on pages 3, 6, 7, and 10.)

[Sch24] Sophie Schmieg. Unbindable kemmy schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK. Cryptology ePrint Archive, Paper 2024/523, 2024. https://eprint.iacr.org/2024/523. (Cited on pages 3, 7, and 8.)

[Sig] Signal: Technical information. `https://signal.org/docs/`. (Cited on page 1.)

[VGIK20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the Signal protocol. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20: 18th International Conference on Applied Cryptography and Network Security, Part II*, volume 12147 of *Lecture Notes in Computer Science*, pages 188–209, Rome, Italy, October 19–22, 2020. Springer, Heidelberg, Germany. (Cited on page 2.)

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. (Cited on page 5.)

[YCW14] Hongbo Yu, Jiazhe Chen, and Xiaoyun Wang. Partial-collision attack on the round-reduced compression function of Skein-256. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 263–283, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany. (Cited on page 5.)