



Migrating to Hybrid Cryptography in Practice: The TutaCrypt Protocol and Its Security

Christian Holler , Tibor Jager and Tom Neuschulten

University of Wuppertal, Wuppertal, Germany

Abstract.

In this work, we study the hybrid key establishment protocol TUTACRYPT as a concrete case of post-quantum cryptographic migration in practice. The protocol was developed by the end-to-end encrypted email provider Tuta and has been deployed to more than ten million users worldwide. We present the protocol in a form that enables rigorous cryptographic analysis and define two Bellare–Rogaway-style security models that precisely characterize the provided security guarantees. The two models capture that the security properties achieved in the pre-quantum setting are slightly stronger than in the post-quantum setting. We then give reduction-based security proofs that clarify under which assumptions the construction achieves security, and how its guarantees degrade if either the classical or the post-quantum component is compromised.

Our results illustrate how formally grounded cryptographic models can capture real-world migration strategies and hybrid deployments. Such analyses help to understand the security properties of deployed cryptographic systems and help move cryptographic migration from best practice toward principled, verifiable design.

Keywords: Protocols · key exchange · hybrid cryptography · post-quantum transition · provable security

1 Introduction

The ongoing transition to post-quantum cryptography raises fundamental challenges for the design and deployment of secure communication systems. While post-quantum cryptographic schemes aim to provide security against adversaries equipped with quantum computers, most such schemes are relatively new and have not yet undergone the same level of scrutiny and real-world testing as classical cryptographic primitives. As a result, many deployed systems adopt *hybrid* cryptographic constructions that combine classical and post-quantum components. The goal of such hybrids is to hedge against cryptanalytic uncertainty: security against classical adversaries should follow solely from well-established classical assumptions, while security against quantum adversaries should rely exclusively on post-quantum assumptions, without depending on quantum-insecure primitives.

This pragmatic approach is increasingly driven by the threat of *store-now-break-later* attacks, in which adversaries record encrypted traffic today with the intention of decrypting it once sufficiently powerful quantum computers become available. For data with long-term confidentiality requirements, protection against quantum adversaries is therefore required already today.

E-mail: holler@uni-wuppertal.de (Christian Holler), jager@uni-wuppertal.de (Tibor Jager), neuschulten@uni-wuppertal.de (Tom Neuschulten)



Application Setting. In this work, we study a hybrid key establishment protocol deployed by the end-to-end encrypted email provider Tuta (<https://tuta.com>). The design of the TUTACRYPT protocol is influenced by the following requirements and application context:

One-pass key exchange. Email is an application that requires a protocol where the sender produces a single message that allows the receiver to derive a shared key without any further interaction. We formally model this as a *one-pass key exchange* (OPKE) protocol. See [BMS20, BJS15, HJLS17, GHJL17, DJSS18] for examples of other protocols with the same requirement, but various different syntactical and security properties.

Centralized infrastructure. Unlike the traditional federated email ecosystem, Tuta operates a centralized infrastructure and provides both client and server software. This allows coordinated cryptographic updates across communication partners, similar in spirit to centralized messaging systems such as Signal or WhatsApp.

End-to-end encryption. An important requirement is that the system is designed such that Tuta itself has no access to the confidential contents of user emails, and all cryptographic protections are enforced end-to-end.

High-level description of the TutaCrypt protocol. At a high level, the protocol combines classical Diffie–Hellman techniques with a post-quantum key encapsulation mechanism. We refer to Section 4 for a detailed description.

Public and secret keys. Each user’s public key consists of two components: a classical ElGamal public key g^u and a post-quantum public key pk_{PQ} for a key encapsulation mechanism (KEM, instantiated with ML-KEM/Kyber [SAB⁺22] in TUTACRYPT). The corresponding secret keys are the discrete logarithm u and the secret key sk_{PQ} belonging to pk_{PQ} .

Encryption. To encrypt an email, the sender generates a TUTACRYPT ciphertext $c = (g^r, c_{PQ})$, where g^r is a fresh Diffie–Hellman share with random r and an ML-KEM/Kyber ciphertext c_{PQ} that encapsulates a random secret k_{PQ} . It derives a shared key by computing

$$k \leftarrow H(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, g^{uv}, g^{vr}, k_{PQ}), \quad (1)$$

where H is a key derivation function, g^u and g^v are the sender’s and the recipient’s ElGamal public keys, respectively, and $pk_{V,PQ}$ is the recipient’s KEM public key. Note that the session key k depends on three shared secrets:

- The shared secret k_{PQ} encapsulated in the KEM ciphertext,
- the “ElGamal KEM” shared secret g^{vr} that combines the sender’s ephemeral randomness r with the receiver’s long-term secret key v , and
- the “Diffie–Hellman non-interactive key exchange (NIKE)” [FHKP13] shared secret g^{uv} that depends on the sender’s and the receiver’s long-term public keys g^u and g^v .

The shared key k is then used to encrypt the actual email. Since we focus on the key establishment protocol, the email encryption scheme is out of scope here.

Decryption. When receiving an encrypted email, the recipient uses its secret key $sk_V = (v, sk_{V,PQ})$ to compute the three shared secrets (k_{PQ}, g^{vr}, g^{uv}) from $c = (g^r, c_{PQ})$, and then derives the shared key as in (1).

This construction is intended to ensure that confidentiality against classical adversaries relies only on the security of classical Diffie–Hellman assumptions, while confidentiality against quantum adversaries relies solely on the post-quantum security of the KEM. The construction can be viewed as a special type of *KEM combiner* [GHP18], except that the syntax is different because the sender’s secret key is used for key derivation at the sender, and its public key for key derivation at the recipient, which is not the case for general KEMs. This is why we chose the abstraction of a *one-pass key exchange* protocol for TUTACRYPT, instead of a KEM. The main consequence of this syntactical difference is that TUTACRYPT achieves slightly stronger security in the pre-quantum setting (namely, what we will define as *strong* OPKE security), while a (combined) KEM may only achieve what we will call *weak* OPKE security. While the TUTACRYPT protocol (1) shares similarities with the QSF framework developed in [BCD⁺24], there are also notable differences, e.g., TUTACRYPT uses a static-static DH key exchange in addition to the ephemeral-static one and the inputs to the KDF differ. Furthermore, the authors analyze the QSF framework in a KEM setting, whereas our security model revolves around a multi-user setting with corruptions.

Contributions. Our main goal is to precisely understand and explain the security guarantees achieved by this hybrid construction. In this sense, our work fits into a long line of research that aims to understand the exact security of cryptographic protocols after they have been widely deployed, including TLS versions before 1.3 [JK02, MSW08, JKSS12, KPW13, GKS13, BFK⁺14], SSH [BKN02], and protocols used in WhatsApp [DFG⁺23, ADJ25] and Signal [CCD⁺20, DJK⁺25, JK25, FG25], for example.

We develop two Bellare–Rogaway-style security models for one-pass key exchange that capture active adversaries, adaptive corruptions of users, and an adaptive Test-query. The need for two models arises because the protocol achieves different guarantees depending on whether the classical or post-quantum setting is considered:

- In the classical setting, the use of the sender’s long-term secret key in key derivation provides a form of implicit authentication. This allows us to prove a *strong* OPKE security, where the adversary may be active in the tested session, unless the sender is corrupted.
- In contrast, in the post-quantum setting, security relies exclusively on the KEM component, and we cannot make any assumptions about a secure sender secret, because it is based on a quantum-insecure discrete logarithm. As a result, only *weak* OPKE security can be achieved, which differs from the strong notion by requiring the adversary to remain passive in the tested session, even if the sender is uncorrupted.

We formalize this distinction and prove matching security guarantees for each case.

Concrete security. Our security proofs are reduction-based and carefully track the concrete security loss. In the pre-quantum setting, we are able to use rerandomizability to improve the concrete security loss of the reduction. To this end, we employ the “count-corruptions-not-users” technique of Bellare, Riepel, Tessaro, and Zhang [BRTZ24], based on a Coron-style [Cor00] partitioning argument, which yields a loss that depends quadratically only on the number of *corrupted* users, instead of the total number of users. This requires a careful embedding of the given Gap-Diffie–Hellman problem instance and leads to significantly improved concrete security bounds for large deployments in the pre-quantum setting.

In principle, one could adopt the same argument for our post-quantum security proof by relying on a multi-user, multi-challenge IND-CCA security notion for the KEM. However, TUTACRYPT is specified for and deployed with ML-KEM/Kyber [SAB⁺22], which is

currently not known to achieve multi-user, multi-challenge IND-CCA security tightly. Consequently, the concrete protocol would not enjoy any improved concrete security from such an approach. It remains an interesting open problem whether a similar Coron-style improvement can also be achieved for TUTACRYPT in the post-quantum setting.

(Quantum) random oracles. In the classical setting, we rely on the classical random oracle model [BR93] in order to achieve active protocol security from primitives such as the Diffie–Hellman NIKE or the ElGamal-based KEMs. In contrast, our post-quantum security proof reduces the security of TUTACRYPT to the IND-CCA security of the underlying KEM (which is ML-KEM in the actual deployment of TUTACRYPT). This reduction is carried out entirely in the standard model. Hence, we avoid the explicit use of quantum random oracles [BDF⁺11] in our security proofs. However, note that quantum random oracles are used in the IND-CCA security proof of ML-KEM/Kyber from [SAB⁺22], when reducing its security to the hardness of computational problems in module lattices. Thus, we implicitly inherit their use, too, when considering a reduction from breaking TUTACRYPT to the hardness of module lattice problems.

Outline. In Section 2 we introduce the cryptographic preliminaries and assumptions used throughout the paper. Section 3 formalizes the notion of one-pass key exchange and presents our Bellare–Rogaway-style security models. In Section 4 we describe the TutaCrypt protocol in detail and prove its security under classical and post-quantum assumptions.

2 Preliminaries

Definition 1 (GapDH and 2-GapDH). Let $\mathbb{G}$ be a cyclic group of prime order $q \in \mathbb{P}$ with generator $g \in \mathbb{G}$. Let $\text{DDH}(g^x, g^y, g^z)$ be an oracle that returns 1 if and only if $xy \equiv z \pmod{q}$. Let $\text{GapDH}_x(\cdot, \cdot) = \text{DDH}(g^x, \cdot, \cdot)$. Consider the following experiment: The challenger $\mathcal{C}$ samples $x, y \xleftarrow{\$} \mathbb{Z}_q$ and initiates $\mathcal{A}^{\text{GapDH}_x(\cdot, \cdot)}(g^x, g^y)$. The advantage of an adversary $\mathcal{A}$ in solving the *Gap-Diffie–Hellman* problem is defined as

$$\text{Adv}_{\mathbb{G}, g}^{\text{GapDH}}(\mathcal{A}) = \Pr \left[\mathcal{A}^{\text{GapDH}_x(\cdot, \cdot)}(g^x, g^y) = g^{xy} \right].$$

Additionally, let $\text{GapDH}_y(\cdot, \cdot) = \text{DDH}(\cdot, g^y, \cdot)$. The *2-Gap-Diffie–Hellman* problem is defined as the same experiment, but initiating $\mathcal{A}^{\text{GapDH}_x(\cdot, \cdot), \text{GapDH}_y(\cdot, \cdot)}(g^x, g^y)$ instead, i.e., the adversary gains access to two DDH-oracles, one fixing g^x and one fixing g^y . The advantage of an adversary $\mathcal{A}$ in solving the *2-Gap-Diffie–Hellman* problem is defined as

$$\text{Adv}_{\mathbb{G}, g}^{2\text{-GapDH}}(\mathcal{A}) = \Pr \left[\mathcal{A}^{\text{GapDH}_x(\cdot, \cdot), \text{GapDH}_y(\cdot, \cdot)}(g^x, g^y) = g^{xy} \right].$$

Note that this assumption, where the DDH-oracles fix one exponent each, is significantly weaker than an assumption that provides the adversary with access to a full-fledged DDH-oracle $\text{DDH}(\cdot, \cdot, \cdot)$ fixing no parameter, which is still considered hard in an appropriate group $\mathbb{G}$.

Definition 2 (KEM). A *key-encapsulation mechanism* (KEM) is a tuple of PPT algorithms (KGen , Encaps , Decaps) that do the following:

1. The *key-generation algorithm* KGen takes the security parameter 1^λ as input and outputs a public-/private-key pair (pk, sk) .
2. The *encapsulation algorithm* Encaps takes as input a public key pk . It outputs a ciphertext c and a key $k \in \{0, 1\}^{\ell(\lambda)}$, where ℓ is the key length. We denote this by $(c, k) \leftarrow \text{Encaps}(pk)$.

3. The *decapsulation algorithm* Decaps takes as input a secret key sk and a ciphertext c and outputs a key k or an error symbol $\perp$. We write this as $k = \text{Decaps}(sk, c)$.

We say that a KEM is $(1 - \varepsilon)$ -correct if

$$\forall (pk, sk) \leftarrow \text{KGen}(1^\lambda): (c, k) \leftarrow \text{Encaps}(pk) \implies \text{Decaps}(sk, c) = k$$

holds with probability at least $1 - \varepsilon$.

Definition 3. The IND-CCA security game for a key-encapsulation mechanism $\text{KEM} = (\text{KGen}, \text{Encaps}, \text{Decaps})$ with key space $\mathcal{K}$ is the following:

$G_{\mathcal{A}, \text{KEM}}^{\text{CCA}}(\lambda)$	$\text{Decaps}^*(sk, c)$
1 : $(pk, sk) \xleftarrow{\$} \text{KGen}(1^\lambda)$	1 : if $c \neq c^*$
2 : $d \xleftarrow{\$} \{0, 1\}$	2 : return $\text{Decaps}(sk, c)$
3 : $(k_0, c^*) \xleftarrow{\$} \text{Encaps}(pk)$	3 : else
4 : $k_1 \xleftarrow{\$} \mathcal{K}$	4 : return $\perp$
5 : $d^* \xleftarrow{\$} \mathcal{A}^{\text{Decaps}^*(sk, \cdot)}(pk, k_d, c^*)$	5 : endif
6 : return $\llbracket d = d^* \rrbracket$	

The advantage of a PPT adversary $\mathcal{A}$ in the CCA-security game for a key-encapsulation mechanism KEM is defined as

$$\text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{A}) = |2 \cdot \Pr[G_{\mathcal{A}, \text{KEM}}^{\text{CCA}}(\lambda) = 1] - 1|.$$

Definition 4 (PRF). Let D , K and T be finite sets and let $\mathcal{F} = \{f: D \rightarrow T\}$ be the set of all functions mapping from D to T . The PRF-security game for a function $F: D \times K \rightarrow T$ is the following:

$G_{\mathcal{A}, F}^{\text{PRF}}(\lambda)$	$\mathcal{O}(x)$
1 : $k \xleftarrow{\$} K$	1 : if $d = 0$
2 : $f \xleftarrow{\$} \mathcal{F}$	2 : return $F(x, k)$
3 : $d \xleftarrow{\$} \{0, 1\}$	3 : else
4 : $d^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\cdot)}()$	4 : return $f(x)$
5 : return $\llbracket d = d^* \rrbracket$	5 : endif

The advantage of a PPT adversary $\mathcal{A}$ in the pseudorandom function (PRF) security game for a function $F: D \times K \rightarrow T$ is defined as

$$\text{Adv}_F^{\text{PRF}}(\mathcal{A}) = |2 \cdot \Pr[G_{\mathcal{A}, F}^{\text{PRF}}(\lambda) = 1] - 1|.$$

3 One-Pass Key Exchange and Its Security

Definition 5. A *one-pass key exchange* (OPKE) protocol consists of the following three probabilistic algorithms.

$\text{OPKE.KGen}(1^\lambda)$ takes as input the security parameter 1^λ , and outputs a public key pk and a secret key sk .

$\text{OPKE.Wrap}(sk, pk)$ takes as input the secret key sk of the sender and the public key pk of the receiver. The output is (c, k) with ciphertext c and key k .

$\text{OPKE.Unwrap}(sk, pk, c)$ takes as input the secret key sk of the receiver, the public key pk of the sender and a ciphertext c . The output is a key k or an error $\perp$.

We say that an OPKE protocol is $(1 - \varepsilon)$ -correct if

$$\Pr \left[\begin{array}{l} (sk_i, pk_i) \leftarrow \text{OPKE.KGen}(1^\lambda) \\ (sk_j, pk_j) \leftarrow \text{OPKE.KGen}(1^\lambda) \\ (c, k_a) \leftarrow \text{OPKE.Wrap}(sk_i, pk_j) \\ k_b \leftarrow \text{OPKE.Unwrap}(sk_j, pk_i, c) \end{array} : k_a = k_b \right] \geq 1 - \varepsilon.$$

Security. Let $\mathcal{I} = \{U_1, \dots, U_N\}$ be the set of identities. We denote $\mathcal{C} \subseteq \mathcal{I}$ to be the set of corrupted identities, which is initialized to the empty set $\mathcal{C} = \emptyset$. Furthermore, we use the following session variables:

- $\text{st}_{exec} \in \{\text{accepted}, \text{rejected}\}$ denotes the current execution state.
- $\text{id} \in \mathcal{I}$ denotes the identity of the session.
- $\text{pid} \in \mathcal{I}$ denotes the identity of the intended communication partner.
- $\text{role} \in \{\text{sender}, \text{receiver}\}$ denotes the identity's role during the session.
- $\text{trans} \in \{0, 1\}^* \cup \{\perp\}$ denotes the transcript of all sent and received messages for the respective session.
- $\text{key} \in \{0, 1\}^* \cup \{\perp\}$ denotes the session key derived/generated during the session.
- $\text{revealed} \in \{\text{false}, \text{true}\}$ denotes whether the session key has been revealed at some point.

For the i^{th} session of user with identity U we write π_U^i . A session variable var of π_U^i is denoted via $\pi_U^i.\text{var}$.

Definition 6. We say that two sessions π_U^i and π_V^j are *partnered* if all of the following hold:

- $\pi_U^i.\text{trans} = \pi_V^j.\text{trans}$ (transcripts match),
- $\pi_U^i.\text{id} = \pi_V^j.\text{pid}$ and $\pi_U^i.\text{pid} = \pi_V^j.\text{id}$ (U and V are each other's intended partners),
- $\pi_U^i.\text{role} \neq \pi_V^j.\text{role}$ (sessions have opposite roles).

We assume the adversary $\mathcal{A}$ controls the full network, meaning that the adversary is responsible for the transport of the messages. This allows the adversary to arbitrarily modify, drop, or reorder messages. Interaction with the key exchange protocol and session is done via the following queries.

NewSessionS(U, V) initializes a new sender session π_U^i for identity $U \in \mathcal{I}$, with the intended partner $V \in \mathcal{I}$. The party U derives the session key k and computes a ciphertext c via $(c, k) \leftarrow \text{OPKE.Wrap}(sk_U, pk_V)$. Finally, the challenger initializes the session variables $\pi_U^i.\text{st}_{exec} = \text{accepted}$, $\pi_U^i.\text{role} = \text{sender}$, $\pi_U^i.\text{id} = U$, $\pi_U^i.\text{pid} = V$, $\pi_U^i.\text{trans} = c$, $\pi_U^i.\text{key} = k$, $\pi_U^i.\text{revealed} = \text{false}$. It returns the ciphertext c .

We define $Q_S \in \text{poly}(\lambda)$ to be the maximal number of **NewSessionS**($U, \cdot$) queries per user.

NewSessionR(U, V, m) initializes a new receiver session π_V^j for identity $V \in \mathcal{I}$, with the intended partner $U \in \mathcal{I}$. Furthermore, the party with V executes $k \leftarrow \text{OPKE.Unwrap}(sk_V, pk_U, m)$. Finally, the session variables are set to $\pi_V^j.\text{role} = \text{receiver}$, $\pi_V^j.\text{id} = V$, $\pi_V^j.\text{pid} = U$, $\pi_V^j.\text{trans} = m$, $\pi_V^j.\text{key} = k$. If $k \neq \perp$, then $\pi_V^j.\text{st}_{\text{exec}} = \text{accepted}$ and $\pi_V^j.\text{revealed} = \text{false}$, else $\pi_V^j.\text{st}_{\text{exec}} = \text{rejected}$.

We define $Q_R \in \text{poly}(\lambda)$ to be the maximal number of possible **NewSessionR**($\cdot, V, \cdot$) queries per user.

Reveal(π_U^i) reveals the session key of π_U^i : If $\pi_U^i.\text{key} \neq \perp$, then set $\pi_U^i.\text{revealed} = \text{true}$ and return k , else return $\perp$.

Corrupt(U) returns the long-term secret key sk_U of identity U and sets $\mathcal{C} \leftarrow \mathcal{C} \cup \{U\}$.

Test(π_A^t) enables the adversary to challenge the key of session π_A^t . If $\pi_A^t.\text{key} = \perp$, return $\perp$. Otherwise, a secret bit $d_{\text{test}} \in \{0, 1\}$ is chosen uniformly at random. If $d_{\text{test}} = 0$, return $\pi_A^t.\text{key}$, else return a uniformly random key drawn from $\{0, 1\}^\lambda$, where $\{0, 1\}^\lambda$ is the session key space of the protocol. This query may be asked only once throughout the security experiment.

Intuition of the security definition. Some restrictions on the adversary's queries are required, in order to rule out certain trivial and generic attacks. Intuitively, we want to achieve the following for one-pass key exchange:

1. The adversary should not be allowed to query **Reveal**(π_A^t) for the tested session. The same goes for revealing a partnered session, as it would derive that same session key. Following that train of thought, we need to prohibit revealing sessions with the same transcript as the test session, due to the following reason. Consider the sequence of queries:
 - (a) The adversary forges a valid ciphertext c on its own.
 - (b) Then it queries **NewSessionR**(A, B, c) *twice*, yielding two sessions π_B^1 and π_B^2 with the same transcript.
 - (c) Querying **Reveal**(π_B^1) now gives $\mathcal{A}$ access to the same key that also π_B^2 computes.

Note that these two sessions are neither partnered themselves, because $\pi_B^1.\text{role} = \text{receiver} = \pi_B^2.\text{role}$, nor partners of some other session, because there was no sender session involved.

2. The receiver identity involved in the test session must never be corrupted during the experiment, since its secret key is sufficient to derive any session key of this party, there is no ephemeral randomness. The sender, however, may be corrupted.
3. We distinguish between strong and weak OPKE security:
 - (a) **Weak OPKE security:** If the tested session is a receiver session, we demand that the adversary has only been *passive* in the tested session. That is, the ciphertext received by this session is not generated by the adversary himself, but it was obtained from a **NewSessionS**-query.
 - (b) **Strong OPKE security:** Here we have the same requirement as in weak OPKE security, but only in case the sender is corrupted. More precisely:
If the tested session is a receiver session and the sender has been corrupted at any point during the experiment, then we demand that the adversary has only been *passive* in the tested session. That is, the ciphertext received by this session is not generated by the adversary himself.

These restrictions prevent the adversary from simply generating the ciphertext of the tested session itself and then trivially computing the session key (possibly using the sender's long-term key).

Definition 7. Let OPKE be a one-pass key exchange protocol and $\mathcal{A}$ an efficient adversary interacting with OPKE via the oracle queries defined as above in the following game:

- The challenger generates keys for all parties $U \in \mathcal{I}$ as $(sk_U, pk_U) \leftarrow \text{OPKE.KGen}(1^\lambda)$.
- $\mathcal{A}$ receives pk_U for all $U \in \mathcal{I}$ and has access to the oracles `NewSessionS`, `NewSessionR`, `Reveal`, `Corrupt`, `Test`.
- Eventually $\mathcal{A}$ stops and outputs a guessing bit $d \in \{0, 1\}$.

The adversary wins if $d = d_{\text{test}}$ and the following conditions hold:

1. For any session π_U^i that shares the same transcript with the test session π_A^t , i.e., with $\pi_U^i.\text{trans} = \pi_A^t.\text{trans}$, we require that $\pi_U^i.\text{revealed} = \text{false}$. In particular, this means the adversary is not allowed to query `Reveal` on the test session π_A^t or any partnered session π_B^s if it exists.
2. For the test session π_A^t with $\pi_A^t.\text{id} = A$ and $\pi_A^t.\text{pid} = B$ we require that:
 - (i) If $\pi_A^t.\text{role} = \text{sender}$, then $B \notin \mathcal{C}$.
 - (ii) If $\pi_A^t.\text{role} = \text{receiver}$, then $A \notin \mathcal{C}$.
3. (a) **Weak OPKE security:** If $\pi_A^t.\text{role} = \text{receiver}$, there exists a session π_B^s with $\pi_B^s.\text{role} = \text{sender}$ that is partnered with π_A^t . This game is the *weak* OPKE security game, denoted $G_{\mathcal{A}, \text{OPKE}}^{\text{wOPKE}}$.
- (b) **Strong OPKE security:** If $\pi_A^t.\text{role} = \text{receiver}$ and $B \in \mathcal{C}$, there exists a session π_B^s with $\pi_B^s.\text{role} = \text{sender}$ that is partnered with π_A^t . This game is the *strong* OPKE security game, denoted $G_{\mathcal{A}, \text{OPKE}}^{\text{sOPKE}}$.

If the adversary wins, the output of the game is 1, otherwise 0. We define the advantage of a PPT adversary $\mathcal{A}$ in the wOPKE security game as

$$\text{Adv}_{\text{OPKE}}^{\text{wOPKE}}(\mathcal{A}) = |2 \cdot \Pr[G_{\mathcal{A}, \text{OPKE}}^{\text{wOPKE}}(\lambda) = 1] - 1|$$

and the advantage in the sOPKE security game as

$$\text{Adv}_{\text{OPKE}}^{\text{sOPKE}}(\mathcal{A}) = |2 \cdot \Pr[G_{\mathcal{A}, \text{OPKE}}^{\text{sOPKE}}(\lambda) = 1] - 1|.$$

OPKE vs. KEMs. The difference between OPKE and a KEM is that the Wrap algorithm of OPKE receives the initiator's *secret* key as input. This does not fit the syntax of a KEM, where the encapsulation algorithm takes only public values as input.

On permitting reveals only for different transcripts. Note that our security model prohibits revealing *any* session with the same transcript as the test session. This restriction could be relaxed, e.g., by allowing to reveal sessions with the same transcript, but only if different parties are involved. We believe that the TUTACRYPT protocol achieves also this slightly stronger notion, but only if one additionally argues that different parties generate the same key pair with only negligible probability.

Single vs. Multiple Test-queries. Note that our security model considers a *single* Test-query. It is also interesting to ask whether multiple Test-queries are possible, both from a theoretical and a concrete security perspective. In the post-quantum setting (i.e., the proof based on KEM security), we believe that proving security in a single-bit-guessing model, as in [JKRS21] for example, with more than a constant number of Test-queries is very difficult, even when assuming a multi-user-multi-instance security IND-CCA notion for the KEM. This is because this would require guessing correctly *all* tested sessions, which cannot be done by a polynomial-time reduction with sufficient success probability (unless the number of tested sessions is constant). We could probably achieve security in a multi-bit-guessing model, as in Bader et al. [BHJ⁺15], Gjøsteen and Jager [GJ18], and Liu et al. [LLGW20], however, [JKRS21] explained that single-bit guessing is the “right” model when considering tight security. The same holds in the 2-Gap-DH setting, except that we use explicit re-randomization instead of a multi-instance IND-CCA security notion. Also here, we would have to guess all tested sessions correctly, and we get the same exponential security loss in the number of Test-queries as above.

The main focus of our paper is to understand the security of TUTACRYPT in a realistic model and applying known techniques in a careful and technically sound manner, such as Coron’s guessing [Cor02], in order to achieve accurate security bounds for the considered model. We argue that a security model with a single Test-query is also interesting, e.g., it reflects the case where the adversary aims to attack one particular session.

4 The TutaCrypt Protocol and Its Security

4.1 Protocol Description

$\Pi_{\text{OPKE}}.\text{KGen}(1^\lambda)$	$\Pi_{\text{OPKE}}.\text{Wrap}(sk_U, pk_V)$
1 : $(x, g^x) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$	1 : $(g^v, pk_{V,\text{PQ}}) := pk_V$
2 : $(sk_{\text{PQ}}, pk_{\text{PQ}}) \leftarrow \Pi_{\text{PQ}}.\text{KGen}(1^\lambda)$	2 : $(u, sk_{U,\text{PQ}}) := sk_U$
3 : $pk := (g^x, pk_{\text{PQ}})$	3 : $(r, g^r) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$
4 : $sk := (x, sk_{\text{PQ}})$	4 : $(k_{\text{PQ}}, c_{\text{PQ}}) \leftarrow \Pi_{\text{PQ}}.\text{Encaps}(pk_{V,\text{PQ}})$
5 : return (pk, sk)	5 : $k \leftarrow \text{H}(g^u, g^r, g^v, pk_{V,\text{PQ}}, c_{\text{PQ}}, (g^v)^u, (g^v)^r, k_{\text{PQ}})$
	6 : $c := (g^r, c_{\text{PQ}})$
	7 : return (c, k)
	$\Pi_{\text{OPKE}}.\text{Unwrap}(sk_V, pk_U, c)$
	1 : $(g^r, c_{\text{PQ}}) := c$
	2 : $(v, sk_{V,\text{PQ}}) := sk_V$
	3 : $(g^u, pk_{U,\text{PQ}}) := pk_U$
	4 : $k_{\text{PQ}} \leftarrow \Pi_{\text{PQ}}.\text{Decaps}(sk_{V,\text{PQ}}, c_{\text{PQ}})$
	5 : if $k_{\text{PQ}} \neq \perp$
	6 : $k \leftarrow \text{H}(g^u, g^r, g^v, pk_{V,\text{PQ}}, c_{\text{PQ}}, (g^u)^v, (g^r)^v, k_{\text{PQ}})$
	7 : else
	8 : $k \leftarrow \perp$
	9 : endif
	10 : return k

Figure 1: The three algorithms $\Pi_{\text{OPKE}}.\text{KGen}$, $\Pi_{\text{OPKE}}.\text{Wrap}$ and $\Pi_{\text{OPKE}}.\text{Unwrap}$ of the TUTACRYPT protocol.

The TUTACRYPT protocol works as follows (see Figure 1 for a precise algorithmic description).

Key generation. The key generation algorithm $\Pi_{\text{OPKE}}.\text{KGen}(1^\lambda)$ first generates a Diffie–Hellman key pair $(x, g^x) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$ with random x , and a key pair $(sk_{\text{PQ}}, pk_{\text{PQ}}) \leftarrow \Pi_{\text{PQ}}.\text{KGen}(1^\lambda)$ for a KEM (specifically, TUTACRYPT uses ML-KEM [SAB⁺22, Nat24], but we will view this as a generic IND-CCA-secure KEM in the security analysis). The public key is $pk = (g^x, pk_{\text{PQ}})$ and the secret key is $sk = (x, sk_{\text{PQ}})$.

Sender-side message generation and key derivation. Algorithm $\Pi_{\text{OPKE}}.\text{Wrap}$ takes as input the sender’s secret key $sk_U = (u, sk_{U,\text{PQ}})$ and the recipient’s public key $pk_V = (g^v, pk_{V,\text{PQ}})$. It generates an ephemeral Diffie–Hellman key share (r, g^r) . Using the recipient’s public key $pk_{V,\text{PQ}}$, it computes a KEM key/ciphertext pair $(k_{\text{PQ}}, c_{\text{PQ}})$. It derives the shared session key as

$$k \leftarrow \text{H}(g^u, g^r, g^v, pk_{V,\text{PQ}}, c_{\text{PQ}}, (g^v)^u, (g^r)^r, k_{\text{PQ}}),$$

where H is a cryptographic hash function and sends the ciphertext $c = (g^r, c_{\text{PQ}})$.

Receiver-side key derivation. Algorithm $\Pi_{\text{OPKE}}.\text{Unwrap}$ receives as input the receiver’s secret key $sk_V = (v, sk_{V,\text{PQ}})$, the sender’s public key $pk_U = (g^u, pk_{U,\text{PQ}})$, and ciphertext $c = (g^r, c_{\text{PQ}})$. It uses $sk_{V,\text{PQ}}$ to obtain the key k_{PQ} encapsulated in c_{PQ} and derives the shared session key as

$$k \leftarrow \text{H}(g^u, g^r, g^v, pk_{V,\text{PQ}}, c_{\text{PQ}}, (g^u)^v, (g^r)^v, k_{\text{PQ}}).$$

Comparison to Signal’s PQXDH Protocol. On the one hand, the TUTACRYPT protocol shares a number of similarities to Signal’s PQXDH handshake protocol [KS24], as both use a combination of Diffie–Hellman shared secrets and a PQ-KEM shared secret as security-critical inputs to the KDF (the values $g^u, g^r, g^v, pk_{V,\text{PQ}}$ and c_{PQ} are either public information or part of the ciphertext/transcript). On the other hand, there are some major differences. While TUTACRYPT does not use any explicit authentication primitive, PQXDH makes use of an identity key pair $(IK_{U,sk}, IK_{U,pk}) =: (\hat{u}, g^{\hat{u}})$ per identity $U \in \mathcal{I}$, instantiated with another DH key pair on curve25519 or curve448. This key pair actually serves a double purpose. First, it is used to sign the so-called “pre-keys” that are stored on the server, which would correspond to the public keys $pk_U = (g^u, pk_{U,\text{PQ}})$ of TUTACRYPT. Second, the identity key pair is used for the computation of two DH secrets, namely $g^{v\hat{u}}$ and $g^{\hat{v}r}$ for some $U, V \in \mathcal{I}$. The value g^{rv} in our notation is an input to the KDF that both protocols have in common. In addition, PQXDH makes use of another DH value $g^{r\rho}$, where (ρ, g^ρ) is an ephemeral one-time pre-key pair of the recipient that has been generated in advance, in order to contribute randomness to the key derivation by the recipient’s side in an asynchronous setting. However, this value is optional, as the recipient might run out of the pre-generated one-time key pairs before being online again to generate new ones. The final session key of PQXDH is then calculated (in our notation) as

$$k \leftarrow \begin{cases} \text{H}(g^{\hat{u}v} \parallel g^{r\hat{v}} \parallel g^{rv} \parallel g^{r\rho} \parallel k_{\text{PQ}}), & \text{if a one-time pre-key } g^\rho \text{ exists on the server,} \\ \text{H}(g^{\hat{u}v} \parallel g^{r\hat{v}} \parallel g^{rv} \parallel k_{\text{PQ}}), & \text{otherwise.} \end{cases}$$

In summary, the PQXDH protocol uses signatures to authenticate the long-term public keys of users, but also uses the signing key pair for the computation of DH shared secrets, resulting in three resp. four of these to be inputs of the KDF evaluation. Meanwhile, TUTACRYPT does not use signatures and therefore also cannot provide DH shared secrets

involving the corresponding key material and ends up with two DH shared secrets. Furthermore, TUTACRYPT does not involve randomness of the recipient's side into the key derivation, making it lose out on certain flavors of forward-security in the pre-quantum setting. An analysis of Signal's protocol has been conducted in [FG25].

4.2 Weak OPKE Security under Post-Quantum Assumptions

In this section, we will prove that TUTACRYPT satisfies weak OPKE security in the sense of Definition 7. Note that the “weak” in this definition is only used to distinguish the security notion from the stronger notion achieved in the pre-quantum setting (see Section 4.3). That is, it refers only to the fact that the adversary is required to be passive in the tested session, which is inherent for protocols where the session key is obtained essentially by a sender encapsulating a secret under a public-key KEM and sending the ciphertext to a receiver.

Theorem 1. *Let $\mathbb{G}$ be a cyclic group with generator $g \in \mathbb{G}$ and let KEM be a key-encapsulation mechanism with public key space $\mathcal{PK}_{PQ}$, ciphertext space $\mathcal{C}_{PQ}$ and key space $\mathcal{K}_{PQ}$. If KEM is CCA-secure and $(1 - \varepsilon)$ -correct and H is a pseudorandom function according to Definition 4 with domain $D = \mathbb{G}^3 \times \mathcal{PK}_{PQ} \times \mathcal{C}_{PQ} \times \mathbb{G}^2$ and key space $K = \mathcal{K}_{PQ}$ (i.e., keyed in the last argument), then the protocol TUTACRYPT is a weakly secure OPKE in the sense of Definition 7. More precisely, from every efficient adversary $\mathcal{A}$, we can construct efficient adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\text{Adv}_{\text{TC}}^{\text{wOPKE}}(\mathcal{A}) \leq N^2 \cdot Q_S \cdot \left(\text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + \text{Adv}_H^{\text{PRF}}(\mathcal{B}_2) + 2\varepsilon \right),$$

and the runtime of $\mathcal{A}$ is approximately the joint runtimes of $\mathcal{B}_1$ and $\mathcal{B}_2$.

Proof. We show the claim by using a sequence of games [Sho04]. Let S_i be the event that the adversary $\mathcal{A}$ wins in Game i and $\text{Adv}_i(\mathcal{A}) = |2 \cdot \Pr[S_i] - 1|$ the corresponding advantage.

Game 0. This is the standard security experiment for weak OPKE security. The advantage of $\mathcal{A}$ is

$$\text{Adv}_0(\mathcal{A}) = |2 \cdot \Pr[S_0] - 1|.$$

Game 1. Game 1 works the same way as Game 0, except that the challenger guesses the sender and receiver identity of the tested session, as well as the specific `NewSessionS`-query for that sender. More explicitly, the challenger picks random $X, Y \xleftarrow{s} \mathcal{I}$ with $|\mathcal{I}| = N$ and $\ell \xleftarrow{s} \{1, \dots, Q_S\}$ and proceeds as in Game 0, until the `Test`-oracle is queried. Let π_A^t be the session such that $\mathcal{A}$ queries `Test`(π_A^t), and let $\pi_A^t.\text{pid} = B$. Let π_B^s denote the partner session of π_A^t , if it exists. We have to consider two cases:

1. $\pi_A^t.\text{role} = \text{sender}$: We have $\pi_B^s.\text{role} = \text{receiver}$ if the session exists. If $A = X$, $B = Y$ and π_X^t is the ℓ^{th} sender session of X , proceed with the experiment;
2. $\pi_A^t.\text{role} = \text{receiver}$: We have $\pi_B^s.\text{role} = \text{sender}$. If $B = X$, $A = Y$ and π_X^s is the ℓ^{th} sender session of X , proceed with the experiment;

if neither case is true, the challenger aborts and outputs a random bit. Note that, in the second case, the partnered sender session must exist due to Definition 7. Since the experiment runs exactly like Game 0, unless the challenger guessed wrong, $\mathcal{A}$ can only distinguish the experiments in that case. The advantage is therefore

$$\text{Adv}_0(\mathcal{A}) = N^2 \cdot Q_S \cdot \text{Adv}_1(\mathcal{A}).$$

Game 2. Game 2 proceeds exactly like Game 1, except that a random KEM-key is drawn, which is independent of the key encapsulated in the ciphertext that corresponds to the tested session. More precisely, let $pk_X = (g^x, pk_{X,PQ})$ and $pk_Y = (g^y, pk_{Y,PQ})$. At the beginning of the experiment, the challenger executes $(k'_{PQ}, c'_{PQ}) \leftarrow \Pi_{PQ}.\text{Encaps}(pk_{Y,PQ})$ and samples a random key $\tilde{k} \xleftarrow{\$} \mathcal{K}_{PQ}$. When $\text{NewSessionS}(X, Y)$ is queried and this is the ℓ^{th} sender session of X , the session key is calculated as

$$k \leftarrow H(g^x, g^r, g^y, pk_{Y,PQ}, c'_{PQ}, g^{xy}, g^{yr}, \tilde{k}),$$

where $(r, g^r) \leftarrow \Pi_{DH}.\text{KGen}(1^\lambda)$. On the receiver side, whenever $\text{NewSessionR}(U, Y, (g', c'_{PQ}))$ for some $U \in \mathcal{I}$ with $pk_U = (g^u, pk_{U,PQ})$ and $g^{r'} \in \mathbb{G}$ is queried, the session key is calculated as

$$k \leftarrow H(g^u, g^{r'}, g^y, pk_{Y,PQ}, c'_{PQ}, g^{uy}, g^{yr'}, \tilde{k}).$$

We will show in Lemma 1 below that

$$\text{Adv}_1(\mathcal{A}) \leq \text{Adv}_2(\mathcal{A}) + \text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + 2\varepsilon.$$

Game 3. In Game 3, we change the **Test**-oracle such that, instead of setting

$$k_0 \leftarrow \pi_A^t.\text{key} = H(g^x, g^r, g^y, pk_{Y,PQ}, c_{PQ}, g^{xy}, g^{yr}, \tilde{k}),$$

we sample a uniformly random key $k_0 \xleftarrow{\$} \{0,1\}^\lambda$, where $\{0,1\}^\lambda$ is the key space of TUTCrypt. In Lemma 2 we will show that, due to H being a PRF, the adversary's advantage is bounded by

$$\text{Adv}_2(\mathcal{A}) \leq \text{Adv}_3(\mathcal{A}) + \text{Adv}_H^{\text{PRF}}(\mathcal{B}_2).$$

As k_0 and k_1 are now *both* random, independent of the bit d_{test} used in $\text{Test}(\pi_A^t)$, the adversary can do no better than guessing, i.e., $\Pr[S_3] = 1/2$ or, equivalently, $\text{Adv}_3(\mathcal{A}) = 0$. In total, we obtain

$$\begin{aligned} \text{Adv}_{\text{TC}}^{\text{wOPKE}}(\mathcal{A}) &= \text{Adv}_0(\mathcal{A}) = N^2 \cdot Q_S \cdot \text{Adv}_1(\mathcal{A}) \\ &\leq N^2 \cdot Q_S \cdot (\text{Adv}_2(\mathcal{A}) + \text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + 2\varepsilon) \\ &\leq N^2 \cdot Q_S \cdot (\text{Adv}_3(\mathcal{A}) + \text{Adv}_H^{\text{PRF}}(\mathcal{B}_2) + \text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + 2\varepsilon) \\ &= N^2 \cdot Q_S \cdot (\text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + \text{Adv}_H^{\text{PRF}}(\mathcal{B}_2) + 2\varepsilon). \end{aligned}$$

□

It remains to prove the following two lemmas.

Lemma 1. *We have*

$$\text{Adv}_1(\mathcal{A}) \leq \text{Adv}_2(\mathcal{A}) + \text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + 2\varepsilon.$$

In particular, assuming that KEM is IND-CCA-secure, Game 1 and Game 2 in the proof of Theorem 1 are computationally indistinguishable.

Proof. To see that Game 2 is indistinguishable from Game 1, we construct an adversary $\mathcal{B}_1$ against the IND-CCA security of the PQ-KEM. $\mathcal{B}_1$ receives as input a public key pk of the KEM and a challenge (k_d, c^*) , where k_0 is the real KEM key and k_1 is random. In Game 1 we may assume that $\mathcal{B}_1$ guessed the correct parties $X = A, Y = B$ and the ℓ^{th} $\text{NewSessionS}(A, \cdot)$ -query (if the test session π_A^t is a sender session) resp. $X = B, Y = A$

and the ℓ^{th} $\text{NewSessionS}(B, \cdot)$ -query (if the test session is a receiver session partnered with π_B^s), as otherwise the game is aborted. $\mathcal{B}_1$ keeps track of the set of corrupted parties, which is initialized as $\mathcal{C} \leftarrow \emptyset$. It generates key pairs for all parties, except Y (the guessed receiver identity):

$$\forall U \in \mathcal{I} \setminus \{Y\}: (pk_U, sk_U) \leftarrow \Pi_{\text{OPKE}}.\text{KGen}(1^\lambda).$$

$\mathcal{B}_1$ generates a long-term DH key pair $(y, g^y) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$ for Y and sets $pk_Y := (g^y, pk)$, where pk is the challenge public key that $\mathcal{B}_1$ received from its challenger.

With the generated key material for all identities, except Y , $\mathcal{B}_1$ is able to respond to all oracle queries that do not involve Y . Thus, in the sequel we only have to explain how the queries involving Y are handled. To simplify notation, in the sequel we let $\mathcal{M} := \mathbb{G}^3 \times \mathcal{PK}_{\text{PQ}} \times \mathcal{C}_{\text{PQ}} \times \mathbb{G}^2 \times \mathcal{K}_{\text{PQ}}$, so that we have $\text{H}: \mathcal{M} \rightarrow \{0, 1\}^\lambda$.

NewSessionS(X, Y): Parse the public key of Y as $(g^y, pk) := pk_Y$, the secret key of X as $(x, sk_{X, \text{PQ}}) := sk_X$ and generate ephemeral values $(r, g^r) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$. We have to consider two cases:

1. If this is *not* the ℓ^{th} sender session of X , the reduction $\mathcal{B}_1$ executes the protocol honestly, i.e., $(k_{\text{PQ}}, c_{\text{PQ}}) \leftarrow \Pi_{\text{PQ}}.\text{Encaps}(pk)$, computes the session key via $k \leftarrow \text{H}(g^x, g^r, g^y, pk, c_{\text{PQ}}, g^{xy}, g^{yr}, k_{\text{PQ}})$ and sets $c := (g^r, c_{\text{PQ}})$.
2. If this *is* the ℓ^{th} sender session of X , the reduction uses the challenge (k_d, c^*) to compute $k \leftarrow \text{H}(g^x, g^r, g^y, pk, c^*, g^{xy}, g^{yr}, k_d)$ and sets $c := (g^r, c^*)$.

$\mathcal{B}_1$ then proceeds like the original experiment, i.e., it assigns the session variables $\pi_X^i.\text{st}_{\text{exec}} = \text{accepted}$, $\pi_X^i.\text{role} = \text{sender}$, $\pi_X^i.\text{id} = X$, $\pi_X^i.\text{pid} = Y$, $\pi_X^i.\text{trans} = c$, $\pi_X^i.\text{key} = k$, $\pi_X^i.\text{revealed} = \text{false}$. Finally, the ciphertext c is returned.

NewSessionS(U, Y) for $U \in \mathcal{I} \setminus \{X\}$: This query can be answered with an honest execution of $\Pi_{\text{OPKE}}.\text{Wrap}(sk_U, pk_Y)$, as it does not involve the KEM secret key of Y . The session variables are assigned accordingly and the ciphertext c is returned.

NewSessionS(Y, V) for $V \in \mathcal{I}$: This query can be answered with an honest execution of $\Pi_{\text{OPKE}}.\text{Wrap}(sk_Y, pk_V)$, as it does not involve the KEM secret key of Y . Note that, in particular, we could have $V = Y$. Finally, the ciphertext c is returned.

NewSessionR(Y, V, m) for $V \in \mathcal{I} \setminus \{Y\}, m \in \mathbb{G} \times \mathcal{C}_{\text{PQ}}$: This query is answered with an honest execution of $\Pi_{\text{OPKE}}.\text{Unwrap}(sk_V, pk_Y, m)$, as it does not involve the KEM secret key of Y .

NewSessionR(U, Y, m) for $U \in \mathcal{I}, m \in \mathbb{G} \times \mathcal{C}_{\text{PQ}}$: Parse the message as $(g^r, c_{\text{PQ}}) := m$ and the public key of U as $(g^u, pk_{U, \text{PQ}})$. The reduction $\mathcal{B}_1$ cannot execute the decapsulation of c_{PQ} by itself, because it does not have the corresponding secret key.

If $c_{\text{PQ}} \neq c^*$, then $\mathcal{B}_1$ makes use of the decapsulation oracle provided by the challenger in the CCA-security experiment by querying for the decryption of c_{PQ} and then proceeding like the original experiment.

If $c_{\text{PQ}} = c^*$, $\mathcal{B}_1$ makes use of the challenge (k_d, c^*) by defining the session key as

$$k \leftarrow \text{H}(g^u, g^r, g^y, pk, c^*, g^{uy}, g^{yr}, k_d).$$

Apart from this, $\mathcal{B}_1$ proceeds like the original experiment.

Reveal(π_U^i) for $U \in \mathcal{I}, i \in \{1, \dots, Q_S + Q_R\}$: Since we may assume that the reduction has guessed X, Y and ℓ correctly, we have $\{X, Y\} = \{A, B\}$ and the test session is π_A^t with partner session π_B^s (if it exists).

Sessions which have the same transcript as the test session make π_A^t an invalid test session due to Definition 7, so we do not have to consider **Reveal**-queries for such sessions.

Thus, we have to consider sessions which have a different transcript than that of the test session. Recall that, as described above, for all such sessions, $\mathcal{B}_1$ simulates the **NewSessionS**(U, V)- and **NewSessionR**(U, V, m)-queries in a way such that also a key k is derived. Therefore $\mathcal{B}_1$ is able to answer corresponding **Reveal**-queries by setting $\pi_U^i.\text{revealed} = \text{true}$ and returning the session key k .

Corrupt(Y): Since we may assume that the reduction has guessed X and Y correctly, we have $\{X, Y\} = \{A, B\}$, and the test session is π_A^t with $\pi_A^t.\text{pid} = B$. So this query cannot happen, because it would make π_A^t an invalid test session.

Finally, we need to answer the **Test**-query **Test**(π_A^t). This is done by flipping a fair coin $d_{\text{test}} \xleftarrow{\$} \{0, 1\}$, sampling $k_1 \xleftarrow{\$} \{0, 1\}^\lambda$ and returning

$$k^* \leftarrow \begin{cases} \pi_A^t.\text{key} = H(g^x, g^r, g^y, pk, c^*, g^{xy}, g^{yr}, k_d), & d_{\text{test}} = 0, \\ k_1, & d_{\text{test}} = 1 \end{cases}$$

to $\mathcal{A}$, who then answers with a bit $d^* \in \{0, 1\}$. The reduction just forwards d^* to the challenger, whose output is 1 if $d^* = d$ and 0 otherwise. Let C be the event that $\text{Decaps}(sk, c^*) \neq k_0$ and $\mathcal{A}$ queries **NewSessionR**($U, Y, (g^*, c^*)$) at least once for some $U \in \mathcal{I}$ and $g^* \in \mathbb{G}$, i.e., the reduction reuses the challenge (k_d, c^*) for key derivation instead of executing the decapsulation. If k_d is the “real” key encapsulated in the ciphertext, this is only a perfect simulation of Game 1 if $\text{Decaps}(sk, c^*) = k_d$, so we have to take the $(1 - \varepsilon)$ -correctness of KEM into account here. Meanwhile Game 2 is perfectly simulated if k_d is random. Therefore, we have

$$\Pr[S_1 \mid \overline{C}] = \Pr[d^* = d_{\text{test}} \mid d = 0], \quad \Pr[S_2] = \Pr[d^* = d_{\text{test}} \mid d = 1].$$

Due to the fact that Game 1 and Game 2 only differ in case of $d_{\text{test}} = 0$, so the case $d_{\text{test}} = 1$ takes a random $k_1 \xleftarrow{\$} \{0, 1\}^\lambda$ in both games, the view of $\mathcal{A}$ is the same in that case and it holds that $\Pr[d^* = 1 \mid d_{\text{test}} = 1 \wedge d = 0] = \Pr[d^* = 1 \mid d_{\text{test}} = 1 \wedge d = 1]$. Using the (reverse) triangle inequality and plugging in the terms for $\Pr[S_1 \mid \overline{C}]$ and $\Pr[S_2]$, we obtain

$$\begin{aligned} |\text{Adv}_1(\mathcal{A}) - \text{Adv}_2(\mathcal{A})| &= |2 \cdot \Pr[S_1] - 1| - |2 \cdot \Pr[S_2] - 1| \\ &\leq |2 \cdot \Pr[S_1] - 1 - (2 \cdot \Pr[S_2] - 1)| \\ &= 2 \cdot |\Pr[S_1] - \Pr[S_2]| \\ &= 2 \cdot |\Pr[S_1 \mid C] \cdot \Pr[C] + \Pr[S_1 \mid \overline{C}] \cdot \Pr[\overline{C}] - \Pr[S_2]| \\ &= 2 \cdot |\Pr[S_1 \mid C] \cdot \Pr[C] + \Pr[S_1 \mid \overline{C}] \cdot (1 - \Pr[C]) - \Pr[S_2]| \\ &= 2 \cdot |(\Pr[S_1 \mid C] - \Pr[S_1 \mid \overline{C}]) \cdot \Pr[C] + \Pr[S_1 \mid \overline{C}] - \Pr[S_2]| \\ &\leq 2 \cdot |\Pr[S_1 \mid C] - \Pr[S_1 \mid \overline{C}]| \cdot \Pr[C] + 2 \cdot |\Pr[S_1 \mid \overline{C}] - \Pr[S_2]| \\ &\leq 2 \cdot 1 \cdot \Pr[\text{Decaps}(sk, c^*) \neq k_0] + 2 \cdot |\Pr[S_1 \mid \overline{C}] - \Pr[S_2]| \\ &\leq 2 \cdot \varepsilon + 2 \cdot |\Pr[d^* = d_{\text{test}} \mid d = 0] - \Pr[d^* = d_{\text{test}} \mid d = 1]| \\ &= 2 \cdot \frac{1}{2} |\Pr[d^* = 0 \mid d_{\text{test}} = 0 \wedge d = 0] + \Pr[d^* = 1 \mid d_{\text{test}} = 1 \wedge d = 0] \\ &\quad - (\Pr[d^* = 0 \mid d_{\text{test}} = 0 \wedge d = 1] + \Pr[d^* = 1 \mid d_{\text{test}} = 1 \wedge d = 1])| + 2\varepsilon \\ &= \left| \frac{\Pr[d^* = 0 \wedge d = 0 \mid d_{\text{test}} = 0]}{\Pr[d = 0]} - \left(1 - \frac{\Pr[d^* = 1 \wedge d = 1 \mid d_{\text{test}} = 0]}{\Pr[d = 1]} \right) \right| + 2\varepsilon \end{aligned}$$

$$\begin{aligned}
&= |2 \cdot (\Pr[d^* = 0 \wedge d = 0 \mid d_{\text{test}} = 0] + \Pr[d^* = 1 \wedge d = 1 \mid d_{\text{test}} = 0]) - 1| + 2\varepsilon \\
&= |2 \cdot \Pr[d^* = d \mid d_{\text{test}} = 0] - 1| + 2\varepsilon \\
&= |2 \cdot \Pr[\mathcal{G}_{\mathcal{B}_1, \text{KEM}}^{\text{CCA}}(\lambda) = 1] - 1| + 2\varepsilon \\
&= \text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + 2\varepsilon.
\end{aligned}$$

Thus, we get

$$\text{Adv}_1(\mathcal{A}) \leq \text{Adv}_2(\mathcal{A}) + \text{Adv}_{\text{KEM}}^{\text{CCA}}(\mathcal{B}_1) + 2\varepsilon.$$

□

Lemma 2. *We have*

$$\text{Adv}_2(\mathcal{A}) \leq \text{Adv}_3(\mathcal{A}) + \text{Adv}_{\text{H}}^{\text{PRF}}(\mathcal{B}_2).$$

In particular, assuming that H is a pseudorandom function, Game 2 and Game 3 in the proof of Theorem 1 are computationally indistinguishable.

Proof. We will construct an efficient adversary $\mathcal{B}_2$ for the PRF experiment with H. $\mathcal{B}_2$ proceeds almost identical to Game 1. In particular it generates key pairs for all identities by computing

$$\forall U \in \mathcal{I}: (pk_U, sk_U) \leftarrow \Pi_{\text{OPKE}}.\text{KGen}(1^\lambda).$$

Since all relevant key material is generated by $\mathcal{B}_2$, it can answer all queries of `NewSessionS`, `NewSessionR`, `Reveal` and `Corrupt` by $\mathcal{A}$ with an honest execution.

The only difference is how the `Test`-oracle query is handled. When $\mathcal{A}$ queries $\text{Test}(\pi_A^t)$, the reduction answers as follows. We have $\pi_A^t.\text{pid} = B$ with $\pi_A^t.\text{trans} = (g^r, c_{\text{PQ}})$ and $(pk_A, sk_A) = ((g^a, pk_{A,\text{PQ}}), (a, sk_{A,\text{PQ}}))$ as well as $(pk_B, sk_B) = ((g^b, pk_{B,\text{PQ}}), (b, sk_{B,\text{PQ}}))$.

- If $\pi_A^t.\text{role} = \text{sender}$, we set $x \leftarrow (g^a, g^r, g^b, pk_{B,\text{PQ}}, c_{\text{PQ}}, g^{ab}, g^{br})$,
- else we set $x \leftarrow (g^b, g^r, g^a, pk_{A,\text{PQ}}, c_{\text{PQ}}, g^{ba}, g^{ar})$.

Now $\mathcal{B}_2$ uses the PRF-oracle $\mathcal{O}(\cdot)$ with input x and receives $y := \mathcal{O}(x)$ from the PRF experiment. Then $\mathcal{B}_2$ flips a fair coin $d_{\text{test}} \xleftarrow{\$} \{0, 1\}$, samples $k_1 \xleftarrow{\$} \{0, 1\}^\lambda$ and sets

$$k^* \leftarrow \begin{cases} y, & d_{\text{test}} = 0, \\ k_1, & d_{\text{test}} = 1, \end{cases}$$

which is sent to $\mathcal{A}$, who will eventually answer with $d^* \in \{0, 1\}$. Finally, $\mathcal{B}_2$ forwards d^* to the challenger.

Note that $\mathcal{B}_2$ answers all queries of $\mathcal{A}$ correctly and perfectly simulates Game 2 if the PRF is being used, i.e., $\mathcal{O}(\cdot) = \text{H}(\cdot, \tilde{k})$, while Game 3 is perfectly simulated if a random function is being used. As $\mathcal{B}_2$ just forwards the output of $\mathcal{A}$, a difference in the advantages between the two games corresponds to the advantage of $\mathcal{B}_2$ in the PRF security game. With the same methodology as in the proof of Lemma 1 we thus get

$$\text{Adv}_2(\mathcal{A}) \leq \text{Adv}_3(\mathcal{A}) + \text{Adv}_{\text{H}}^{\text{PRF}}(\mathcal{B}_2).$$

□

4.3 Strong OPKE Security under Pre-Quantum Assumptions

In this section, we will prove that TUTCrypt satisfies strong OPKE security in the sense of Definition 7, assuming hardness of the 2-Gap-Diffie-Hellman problem as defined in Definition 1. Note that this is a pre-quantum assumption, because it requires the hardness of the discrete logarithm problem, which can be efficiently solved on a quantum computer by Shor's algorithm [Sho94].

Theorem 2. *Let $\mathbb{G}$ be a cyclic group of order $q \in \mathbb{P}$ with generator $g \in \mathbb{G}$. If the 2-GapDH-assumption holds in $\mathbb{G}$, then the protocol TutaCrypt is a strongly secure OPKE in the random oracle model. More precisely, from every efficient adversary $\mathcal{A}$, we can construct an efficient adversary $\mathcal{B}$ such that*

$$\text{Adv}_{\text{TC}}^{\text{sOPKE}}(\mathcal{A}) \leq 4 \cdot \exp(1) \cdot (Q_C + 1)^2 \cdot \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B}),$$

where Q_C denotes the number of **Corrupt**-queries. The runtime of $\mathcal{A}$ is approximately that of $\mathcal{B}$.

Proof. We show the claim by using a sequence of games. Let S_i be the event that the adversary $\mathcal{A}$ wins in Game i and $\text{Adv}_i(\mathcal{A}) = |2 \cdot \Pr[S_i] - 1|$ the advantage.

Game 0. This is the standard security experiment for sOPKE security, so we have

$$\text{Adv}_{\text{TC}}^{\text{sOPKE}}(\mathcal{A}) = \text{Adv}_0(\mathcal{A}).$$

Game 1. Game 1 works the same way as Game 0, except that the challenger guesses a subset of identities for possible corruptions, as well as subsets for sender and receiver of the **Test**-query, in order to implement the Coron-style *count-corruptions-not-users* technique [Cor00, BRTZ24]. More explicitly, let $\text{D}(N, Q_C)$ be a biased-coin sampler with $\Pr[\zeta = 0 \mid \zeta \leftarrow \text{D}(N, Q_C)] = \frac{1}{Q_C+1}$ (which, according to [BRTZ24, Theorem 3.2], is the optimal biased-coin sampler). At the beginning of the experiment, the challenger flips a biased coin for every identity, i.e., $\forall U \in \mathcal{I}: \zeta_U \leftarrow \text{D}(N, Q_C)$ and sets

$$\mathcal{I}_{\overline{\text{corr}}} := \{U \in \mathcal{I} \mid \zeta_U = 0\}, \quad \mathcal{I}_{\text{corr}} := \{U \in \mathcal{I} \mid \zeta_U = 1\}.$$

The set $\mathcal{I}_{\text{corr}}$ then contains all identities that may be corrupted during the game, while identities from $\mathcal{I}_{\overline{\text{corr}}}$ may not. Furthermore, it flips a fair coin $\zeta'_U \xleftarrow{\$} \{0, 1\}$ for every identity in $\mathcal{I}_{\overline{\text{corr}}}$ and splits it in two more sets:

$$\mathcal{I}_x := \{U \in \mathcal{I}_{\overline{\text{corr}}} \mid \zeta'_U = 0\}, \quad \mathcal{I}_y := \{U \in \mathcal{I}_{\overline{\text{corr}}} \mid \zeta'_U = 1\}.$$

This leaves us with the partitioning

$$\mathcal{I} = \mathcal{I}_{\overline{\text{corr}}} \dot{\cup} \mathcal{I}_{\text{corr}} = \mathcal{I}_x \dot{\cup} \mathcal{I}_y \dot{\cup} \mathcal{I}_{\text{corr}}.$$

Intuitively, $\mathcal{I}_x$ and $\mathcal{I}_y$ will be the sets of identities, whose public DH keys are a re-randomization of g^x or g^y , respectively.

We want the test session to contain one DH share with g^x and g^y each. This happens if one identity is in $\mathcal{I}_x$ and the other in $\mathcal{I}_y$ or the receiver is in $\mathcal{I}_{\overline{\text{corr}}}$ (i.e., either in $\mathcal{I}_x$ or $\mathcal{I}_y$) and the sender is in $\mathcal{I}_{\text{corr}}$, where we embed the opposing element of $\{g^x, g^y\}$ in the ephemeral DH share, depending on which one the receiver's public key contains. Since embedding into the randomness of the ciphertext is only possible with *passive* adversaries $\mathcal{A}_{\text{pass}}$ that are passive in the test session, and thus we have to embed into both long-term keys in case of *active* adversaries, we split the analysis in two parts.

We start with adversaries $\mathcal{A}_{\text{act}}$ that are *active* in the test session π_A^t with $\pi_A^t.\text{pid} = B$ and let

$$E_{\text{act}} \Leftrightarrow (A \in \mathcal{I}_x \wedge B \in \mathcal{I}_y) \vee (A \in \mathcal{I}_y \wedge B \in \mathcal{I}_x),$$

which captures the case of active adversaries. Now let

$$G_{\text{act}} \Leftrightarrow E_{\text{act}} \wedge \mathcal{C} \subseteq \mathcal{I}_{\text{corr}}$$

denote the event that E_{act} occurs and all identities for which corruption queries are asked fall into the set $\mathcal{I}_{\text{corr}}$. This is a “good” case for our reduction. If G_{act} does not occur, the challenger aborts and outputs a random bit, i.e., G_{act} is the event that we do *not* abort, so we get $\Pr[S_1 \mid \overline{G_{\text{act}}}] = 1/2$. Since the experiment runs exactly like Game 0, unless $\overline{G_{\text{act}}}$ occurs, $\mathcal{A}_{\text{act}}$ can only distinguish the experiments in that case. This gives us $\Pr[S_1 \mid G_{\text{act}}] = \Pr[S_0 \mid G_{\text{act}}] = \Pr[S_0]$. We will now further examine the event G_{act} . The probability of E_{act} can be calculated as

$$\begin{aligned} \Pr[E_{\text{act}}] &= \Pr[(A \in \mathcal{I}_x \wedge B \in \mathcal{I}_y) \vee (A \in \mathcal{I}_y \wedge B \in \mathcal{I}_x)] \\ &= \Pr[A \in \mathcal{I}_x \wedge B \in \mathcal{I}_y] + \Pr[A \in \mathcal{I}_y \wedge B \in \mathcal{I}_x] \\ &= \frac{1}{(Q_C + 1)^2} \cdot \frac{1}{2^2} + \frac{1}{(Q_C + 1)^2} \cdot \frac{1}{2^2} \\ &= \frac{1}{2(Q_C + 1)^2}. \end{aligned}$$

The third equality comes from the distributions of the independent (biased-)coin flips in the beginning of the experiment, e.g., we get

$$\begin{aligned} \Pr[A \in \mathcal{I}_x \wedge B \in \mathcal{I}_y] &= \Pr[A \in \mathcal{I}_x] \cdot \Pr[B \in \mathcal{I}_y] \\ &= \Pr[\zeta_A = 0 \wedge \zeta'_A = 0] \cdot \Pr[\zeta_B = 0 \wedge \zeta'_B = 1] \\ &= \Pr[\zeta_A = 0] \cdot \Pr[\zeta'_A = 0] \cdot \Pr[\zeta_B = 0] \cdot \Pr[\zeta'_B = 1] \\ &= \frac{1}{Q_C + 1} \cdot \frac{1}{2} \cdot \frac{1}{Q_C + 1} \cdot \frac{1}{2} = \frac{1}{(Q_C + 1)^2} \cdot \frac{1}{2^2}. \end{aligned}$$

The other probability in that line can be calculated analogously. Further, we have

$$\Pr[\mathcal{C} \subseteq \mathcal{I}_{\text{corr}}] = \Pr[\forall U \in \mathcal{C}: \zeta_U = 1] = \left(1 - \frac{1}{Q_C + 1}\right)^{Q_C}.$$

Since the (biased-)coin flips in the beginning are independent of each other, putting this together yields a bound for the probability of G_{act} :

$$\begin{aligned} \Pr[G_{\text{act}}] &= \Pr[E_{\text{act}} \wedge \mathcal{C} \subseteq \mathcal{I}_{\text{corr}}] = \Pr[E_{\text{act}}] \cdot \Pr[\mathcal{C} \subseteq \mathcal{I}_{\text{corr}}] \\ &= \frac{1}{2(Q_C + 1)^2} \cdot \left(1 - \frac{1}{Q_C + 1}\right)^{Q_C} \\ &\geq \frac{1}{2 \cdot \exp(1) \cdot (Q_C + 1)^2}, \end{aligned}$$

where we used the fact that the sequence $a_n := \left(1 - \frac{1}{n+1}\right)^n$ is monotonically decreasing and convergent with $\lim_{n \rightarrow \infty} a_n = 1/\exp(1)$ [BRTZ24, Theorem 3.2]. Finally, we obtain the probability

$$\begin{aligned} \text{Adv}_1(\mathcal{A}_{\text{act}}) &= |2\Pr[S_1] - 1| \\ &= |2\Pr[S_1 \mid G_{\text{act}}] \cdot \Pr[G_{\text{act}}] + 2\Pr[S_1 \mid \overline{G_{\text{act}}}] \cdot \Pr[\overline{G_{\text{act}}}] - 1| \\ &= \left| 2\Pr[S_1 \mid G_{\text{act}}] \cdot \Pr[G_{\text{act}}] + 2 \cdot \frac{1}{2} \cdot \Pr[\overline{G_{\text{act}}}] - 1 \right| \\ &= |2\Pr[S_1 \mid G_{\text{act}}] \cdot \Pr[G_{\text{act}}] + (1 - \Pr[G_{\text{act}}]) - 1| \\ &= |2\Pr[S_0 \mid G_{\text{act}}] \cdot \Pr[G_{\text{act}}] - \Pr[G_{\text{act}}]| \\ &= \Pr[G_{\text{act}}] \cdot |2\Pr[S_0] - 1| \\ &= \Pr[G_{\text{act}}] \cdot \text{Adv}_0(\mathcal{A}_{\text{act}}) \end{aligned}$$

$$\geq \frac{1}{2 \cdot \exp(1) \cdot (Q_C + 1)^2} \cdot \text{Adv}_0(\mathcal{A}_{\text{act}}).$$

We proceed with an analysis for adversaries $\mathcal{A}_{\text{pass}}$ that are *passive* in the test session. Therefore, let π_U^i be any session with $\pi_U^i.\text{id} = U$ and $\pi_U^i.\text{pid} = V$ for any $U, V \in \mathcal{I}$. We define the two functions

$$S(\pi_U^i) := \begin{cases} U, & \text{if } \pi_U^i.\text{role} = \text{sender}, \\ V, & \text{if } \pi_U^i.\text{role} = \text{receiver}, \end{cases} \quad R(\pi_U^i) := \begin{cases} V, & \text{if } \pi_U^i.\text{role} = \text{sender}, \\ U, & \text{if } \pi_U^i.\text{role} = \text{receiver}. \end{cases}$$

Analogously, let

$$\begin{aligned} E_{\text{pass}} &\Leftrightarrow S(\pi_A^t) \in \mathcal{I}_{\text{corr}} \wedge R(\pi_A^t) \in \mathcal{I}_{\overline{\text{corr}}}, \\ G_{\text{pass}} &\Leftrightarrow E_{\text{pass}} \wedge \mathcal{C} \subseteq \mathcal{I}_{\text{corr}}. \end{aligned}$$

If G_{pass} does not occur, the challenger aborts and outputs a random bit. Similar to the first part of the analysis, we get $\Pr[S_1 \mid G_{\text{pass}}] = \Pr[S_0 \mid G_{\text{pass}}] = \Pr[S_0]$ and we can calculate

$$\begin{aligned} \Pr[E_{\text{pass}}] &= \Pr[S(\pi_A^t) \in \mathcal{I}_{\text{corr}} \wedge R(\pi_A^t) \in \mathcal{I}_{\overline{\text{corr}}}] \\ &= \left(1 - \frac{1}{Q_C + 1}\right) \cdot \frac{1}{Q_C + 1}. \end{aligned}$$

We obtain the bound

$$\begin{aligned} \Pr[G_{\text{pass}}] &= \Pr[E_{\text{pass}} \wedge \mathcal{C} \subseteq \mathcal{I}_{\text{corr}}] = \Pr[E_{\text{pass}}] \cdot \Pr[\mathcal{C} \subseteq \mathcal{I}_{\text{corr}}] \\ &= \left(1 - \frac{1}{Q_C + 1}\right) \cdot \frac{1}{Q_C + 1} \cdot \left(1 - \frac{1}{Q_C + 1}\right)^{Q_C} \\ &\geq \frac{Q_C}{\exp(1) \cdot (Q_C + 1)^2}. \end{aligned}$$

For the advantage of $\mathcal{A}_{\text{pass}}$, we have

$$\text{Adv}_1(\mathcal{A}_{\text{pass}}) = \Pr[G_{\text{pass}}] \cdot \text{Adv}_0(\mathcal{A}_{\text{pass}}) \geq \frac{Q_C}{\exp(1) \cdot (Q_C + 1)^2} \text{Adv}_0(\mathcal{A}_{\text{pass}}).$$

According to Lemma 3, which will be proven below, we get the upper bound $\text{Adv}_1(\mathcal{A}) \leq 2 \cdot \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B})$. Applying the bounds derived from all the games consecutively yields

$$\begin{aligned} \text{Adv}_{\text{TC}}^{\text{OPKE}}(\mathcal{A}) &= \text{Adv}_0(\mathcal{A}) \\ &\leq \max\{\text{Adv}_0(\mathcal{A}_{\text{act}}), \text{Adv}_0(\mathcal{A}_{\text{pass}})\} \\ &\leq 2 \cdot \exp(1) \cdot (Q_C + 1)^2 \cdot \text{Adv}_1(\mathcal{A}) \\ &\leq 4 \cdot \exp(1) \cdot (Q_C + 1)^2 \cdot \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B}). \end{aligned}$$

□

Lemma 3. Let $H: \mathcal{M} \rightarrow \{0, 1\}^\lambda$ be modeled as a random oracle, where $\mathcal{M} = \mathbb{G}^3 \times \mathcal{PK}_{PQ} \times \mathcal{C}_{PQ} \times \mathbb{G}^2 \times \mathcal{K}_{PQ}$.¹ Then in the proof of Theorem 2, we get the inequality

$$\text{Adv}_1(\mathcal{A}) \leq 2 \cdot \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B}).$$

¹In practice, a hash function maps from arbitrary finite bit strings to bit strings of a certain length, i.e., we would have $H: \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ and not necessarily only inputs from $\mathcal{M}$. We omit this distinction here for better readability. If the input does not match the desired format of $\mathcal{M}$, the random oracle would just return $H(m) \xleftarrow{\$} \{0, 1\}^\lambda$.

Proof. To show our claim, we construct an adversary $\mathcal{B}$ that attempts to solve the 2-GapDH-problem. $\mathcal{B}$ receives as input two group elements $g^x, g^y \in \mathbb{G}$ and has access to the DDH oracles $\text{GapDH}_x(\cdot, \cdot) = \text{DDH}(g^x, \cdot, \cdot)$ and $\text{GapDH}_y(\cdot, \cdot) = \text{DDH}(\cdot, g^y, \cdot)$ as defined in Definition 1.

$\mathcal{B}$ proceeds exactly like Game 1 of the proof of Theorem 2 and generates the keys for all identities as follows. The identities in $\mathcal{I}_{\text{corr}}$ get the full key material:

$$\forall U \in \mathcal{I}_{\text{corr}}: (sk_U, pk_U) \leftarrow \Pi_{\text{OPKE}}.\text{KGen}(1^\lambda).$$

This allows to answer the queries $\text{Corrupt}(U)$ for $U \in \mathcal{I}_{\text{corr}}$. The identities in $\mathcal{I}_{\text{corr}}$ generate KEM keys:

$$\forall U \in \mathcal{I}_{\text{corr}}: (sk_{U,\text{PQ}}, pk_{U,\text{PQ}}) \leftarrow \Pi_{\text{PQ}}.\text{KGen}(1^\lambda)$$

and their DH keys are set in the following way:

$$\begin{aligned} \forall U \in \mathcal{I}_x: x_U &\stackrel{\$}{\leftarrow} \mathbb{Z}_q, pk_U^{\text{DH}} := g^x \cdot g^{x_U} = g^{x+x_U}, \\ \forall U \in \mathcal{I}_y: y_U &\stackrel{\$}{\leftarrow} \mathbb{Z}_q, pk_U^{\text{DH}} := g^y \cdot g^{y_U} = g^{y+y_U}. \end{aligned}$$

Store the sets $\mathcal{X} := \{x_U \mid U \in \mathcal{I}_x\}$ and $\mathcal{Y} := \{y_U \mid U \in \mathcal{I}_y\}$. We initiate the set of corrupted identities as $\mathcal{C} \leftarrow \emptyset$ as well as an empty set $\mathcal{L}_H \leftarrow \emptyset$ for consistent answers from the random oracle. In addition to that, we need three more sets $\mathcal{L}_{01}, \mathcal{L}_{10}, \mathcal{L}_{11}$ for the reduction to be able to map hash values to the correct **NewSessionS**- or **NewSessionR**-oracle and viceversa, because the reduction does not always know all the required DH values for key derivation. The three sets distinguish which of the DH values (static-static, ephemeral-static or both) are unknown to $\mathcal{B}$, where a “1” in the subscript indicates that $\mathcal{B}$ does *not* know the corresponding value, and “0” indicates that $\mathcal{B}$ knows it, with the left subscript referring to the static-static case. We have

$$\begin{aligned} \mathcal{L}_H &\subseteq \mathbb{G}^3 \times \mathcal{PK}_{\text{PQ}} \times \mathcal{C}_{\text{PQ}} \times \mathbb{G}^2 \times \mathcal{K}_{\text{PQ}} \times \{0, 1\}^\lambda = \mathcal{M} \times \{0, 1\}^\lambda, \\ \mathcal{L}_{01}, \mathcal{L}_{10} &\subseteq \mathbb{G}^3 \times \mathcal{PK}_{\text{PQ}} \times \mathcal{C}_{\text{PQ}} \times \mathbb{G} \times \mathcal{K}_{\text{PQ}} \times \{0, 1\}^\lambda, \\ \mathcal{L}_{11} &\subseteq \mathbb{G}^3 \times \mathcal{PK}_{\text{PQ}} \times \mathcal{C}_{\text{PQ}} \times \mathcal{K}_{\text{PQ}} \times \{0, 1\}^\lambda, \end{aligned}$$

where the last coordinate will always be the assigned hash value. Furthermore, we initialize a set $\mathcal{D} \leftarrow \emptyset$, where we keep track of the values used for re-randomization. We then run $\mathcal{A}(1^\lambda, (pk_U)_{U \in \mathcal{I}})$ and give oracle access to all oracles as described in Definition 7 with the addition of a random oracle $\mathcal{H}$ answering hash queries. However, we ignore the output bit from $\mathcal{A}$, since the solution to the challenge has to be contained in a random oracle query, in order for $\mathcal{A}$ to win with non-zero advantage. Oracle queries of $\mathcal{A}$ involving only identities from $\mathcal{I}_{\text{corr}}$ are answered with an honest execution by $\mathcal{B}$, because all key material is known for these. Hence, we will now describe how $\mathcal{B}$ answers oracle queries involving at least one identity from $\mathcal{I}_{\text{corr}} = \mathcal{I}_x \cup \mathcal{I}_y$.

NewSessionS(U, V) **for** $U, V \in \mathcal{I}_{\text{corr}}$: Parse $(g^u, pk_{U,\text{PQ}}) := pk_U$, $(g^v, pk_{V,\text{PQ}}) := pk_V$, initialize a new sender session π_U^i , and set $\pi_U^i.\text{role} = \text{sender}$, $\pi_U^i.\text{id} = U$, $\pi_U^i.\text{pid} = V$. Start by generating ephemeral values $(r, g^r) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$ and using the recipient’s public key to encapsulate some key material $(k_{\text{PQ}}, c_{\text{PQ}}) \leftarrow \Pi_{\text{PQ}}.\text{Encaps}(pk_{V,\text{PQ}})$. Since both input identities are from the set $\mathcal{I}_{\text{corr}}$, $\mathcal{B}$ has neither of the secret DH keys u, v and cannot compute g^{u^2} , g^{v^2} or g^{uv} , as this would be equivalent to solving the GapDH-challenge. We proceed by filtering out all existing entries of the form $(g^u, g^r, g^v, pk_{V,\text{PQ}}, c_{\text{PQ}}, \alpha_n, g^{rv}, k_{\text{PQ}}, h_n) \in \mathcal{L}_H$ for $\alpha_n \in \mathbb{G}$ and querying the DDH oracle of the challenger. We need to distinguish four cases:

1. $U, V \in \mathcal{I}_x$: Use the set $\mathcal{X}$ to write $g^u = g^{x+x_U}$ and $g^v = g^{x+x_V}$ and check if $\exists n_0: \text{GapDH}_x(g^{x+x_U}, \alpha_{n_0} \cdot (g^x)^{-x_V} \cdot g^{-x_U x_V}) = 1$;

2. $U \in \mathcal{I}_x, V \in \mathcal{I}_y$: Use the sets $\mathcal{X}, \mathcal{Y}$ to write $g^u = g^{x+x_U}$ and $g^v = g^{y+y_V}$ and check if $\exists n_0: \text{GapDH}_y(g^{x+x_U}, \alpha_{n_0} \cdot (g^x)^{-y_V} \cdot g^{-x_U y_V}) = 1$;
3. $U \in \mathcal{I}_y, V \in \mathcal{I}_x$: Use the sets $\mathcal{X}, \mathcal{Y}$ to write $g^u = g^{y+y_U}$ and $g^v = g^{x+x_V}$ and check if $\exists n_0: \text{GapDH}_x(g^{y+y_U}, \alpha_{n_0} \cdot (g^y)^{-x_V} \cdot g^{-y_U x_V}) = 1$;
4. $U, V \in \mathcal{I}_y$: Use the set $\mathcal{Y}$ to write $g^u = g^{y+y_U}$ and $g^v = g^{y+y_V}$ and check if $\exists n_0: \text{GapDH}_y(g^{y+y_U}, \alpha_{n_0} \cdot (g^y)^{-y_V} \cdot g^{-y_U y_V}) = 1$;

if such an n_0 exists in one of these cases, set $k \leftarrow h_{n_0}$. If not, check for an entry in $\mathcal{L}_{10}$, i.e., if $\exists h_{10}: (g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, g^{rv}, k_{PQ}, h_{10}) \in \mathcal{L}_{10}$, set $k \leftarrow h_{10}$. Otherwise, draw a random $k \xleftarrow{\$} \{0, 1\}^\lambda$ and store

$$\mathcal{L}_{10} \leftarrow \mathcal{L}_{10} \cup \{(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, g^{rv}, k_{PQ}, k)\}.$$

The ciphertext is set to $c := (g^r, c_{PQ})$. Next, we finalize the session variables by setting $\pi_U^i.\text{key} = k$, $\pi_U^i.\text{st}_{exec} = \text{accepted}$, $\pi_U^i.\text{trans} = c$ and $\pi_U^i.\text{revealed} = \text{false}$. We return the ciphertext c .

NewSessionS(U, V) **for** $U \in \mathcal{I}_{\text{corr}}, V \in \mathcal{I}_{\text{corr}}$: Since $V \in \mathcal{I}_{\text{corr}}$, the reduction knows the secret DH key v , as well as the generated randomness r . Therefore, it is able to compute g^{uv} and g^{rv} and this query can be answered with an honest execution of $\Pi_{\text{OPKE}}.\text{Wrap}(sk_U, pk_V)$. The ciphertext is $c := (g^r, c_{PQ})$. We return the ciphertext c .

NewSessionS(U, V) **for** $U \in \mathcal{I}_{\text{corr}}, V \in \mathcal{I}_{\text{corr}}$: Parse $(g^u, pk_{U,PQ}) := pk_U$, $(g^v, pk_{V,PQ}) := pk_V$, initialize a new sender session π_U^i , and set $\pi_U^i.\text{role} = \text{sender}$, $\pi_U^i.\text{id} = U$, $\pi_U^i.\text{pid} = V$. Since $U \in \mathcal{I}_{\text{corr}}$, the reduction knows the secret DH key u , which would enable an honest execution, but due to the event E_{pass} in Game 1, testing such a session would not lead to an abort, so we also want to embed the DH-challenge here. Therefore, we use the recipient's public key to encapsulate some key material $(k_{PQ}, c_{PQ}) \leftarrow \Pi_{\text{PQ}}.\text{Encaps}(pk_{V,PQ})$ and generate $(\delta, g^\delta) \leftarrow \Pi_{\text{DH}}.\text{KGen}(1^\lambda)$, which will be used for re-randomization. Since $\mathcal{B}$ does not have the secret DH key v of V , he is not able to compute $g^{(x+\delta)v}$ or $g^{(y+\delta)v}$, as this would be equivalent to solving the GapDH-challenge. We distinguish two cases:

1. $V \in \mathcal{I}_x$: Use the set $\mathcal{X}$ to write $g^v = g^{x+x_V}$, the ciphertext is set to $c := (g^{y+\delta}, c_{PQ})$ and the re-randomization is being tracked by $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\delta, g^y)\}$. Filter out all entries of the form

$$(g^u, g^{y+\delta}, g^{x+x_V}, pk_{V,PQ}, c_{PQ}, g^{u(x+x_V)}, \beta_n, k_{PQ}, h_n) \in \mathcal{L}_H \text{ for } \beta_n \in \mathbb{G}.$$

- If $\exists n_0: \text{GapDH}_x(g^{y+\delta}, \beta_{n_0} \cdot (g^y)^{-x_V} \cdot g^{-\delta x_V}) = 1$, set $k \leftarrow h_{n_0}$.
- Else if $\exists h_{01}: (g^u, g^{y+\delta}, g^{x+x_V}, pk_{V,PQ}, c_{PQ}, g^{u(x+x_V)}, k_{PQ}, h_{01}) \in \mathcal{L}_{01}$, set $k \leftarrow h_{01}$.
- Else draw a random $k \xleftarrow{\$} \{0, 1\}^\lambda$ and store

$$\mathcal{L}_{01} \leftarrow \mathcal{L}_{01} \cup \{(g^u, g^{y+\delta}, g^{x+x_V}, pk_{V,PQ}, c_{PQ}, g^{u(x+x_V)}, k_{PQ}, k)\}.$$

2. $V \in \mathcal{I}_y$: Use the set $\mathcal{Y}$ to write $g^v = g^{y+y_V}$, the ciphertext is set to $c := (g^{x+\delta}, c_{PQ})$ and the re-randomization is being tracked by $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\delta, g^x)\}$. Filter out all entries of the form

$$(g^u, g^{x+\delta}, g^{y+y_V}, pk_{V,PQ}, c_{PQ}, g^{u(y+y_V)}, \beta_n, k_{PQ}, h_n) \in \mathcal{L}_H \text{ for } \beta_n \in \mathbb{G}.$$

- If $\exists n_0: \text{GapDH}_y(g^{x+\delta}, \beta_{n_0} \cdot (g^x)^{-y_V} \cdot g^{-\delta y_V}) = 1$, set $k \leftarrow h_{n_0}$.
- Else if $\exists h_{01}: (g^u, g^{x+\delta}, g^{y+y_V}, pk_{V,PQ}, c_{PQ}, g^{u(y+y_V)}, k_{PQ}, h_{01}) \in \mathcal{L}_{01}$, set $k \leftarrow h_{01}$.

- Else draw a random $k \leftarrow^{\$} \{0, 1\}^\lambda$ and store

$$\mathcal{L}_{01} \leftarrow \mathcal{L}_{01} \cup \{(g^u, g^{x+\delta}, g^{y+y_V}, pk_{V,PQ}, c_{PQ}, g^{u(y+y_V)}, k_{PQ}, k)\}.$$

Next, we finalize the session variables by setting $\pi_U^i.\text{key} = k$, $\pi_U^i.\text{st}_{exec} = \text{accepted}$, $\pi_U^i.\text{trans} = c$ and $\pi_U^i.\text{revealed} = \text{false}$. We return the ciphertext c .

NewSessionR(U, V, m) **for** $U, V \in \mathcal{I}_{\text{corr}}, m \in \mathbb{G} \times \mathcal{C}_{PQ}$: Parse the identities' public keys as $(g^u, pk_{U,PQ}) := pk_U$, $(g^v, pk_{V,PQ}) := pk_V$, the message as $(g^r, c_{PQ}) := m$ and execute $k_{PQ} \leftarrow \Pi_{PQ}.\text{Decaps}(sk_{V,PQ}, c_{PQ})$. We initialize a new receiver session π_V^j , and set $\pi_V^j.\text{role} = \text{receiver}$, $\pi_V^j.\text{id} = V$, $\pi_V^j.\text{pid} = V$, $\pi_V^j.\text{trans} = m$. If $k_{PQ} = \perp$, which means the decapsulation was not successful, we set $\pi_V^j.\text{st}_{exec} = \text{rejected}$, $\pi_V^j.\text{key} \leftarrow \perp$. On a successful decapsulation we proceed by filtering out all existing entries of the form $(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, \alpha_n, \beta_n, k_{PQ}, h_n) \in \mathcal{L}_H$ for $\alpha_n, \beta_n \in \mathbb{G}$ and querying the DDH oracle of the challenger, where we distinguish four cases:

1. $U, V \in \mathcal{I}_x$: Use the set $\mathcal{X}$ to write $g^u = g^{x+x_U}$ and $g^v = g^{x+x_V}$ and check if

$$\begin{aligned} \exists n_1, n_2: & \text{GapDH}_x(g^{x+x_U}, \alpha_{n_1} \cdot (g^x)^{-x_V} \cdot g^{-x_U x_V}) = 1 \\ & \wedge \text{GapDH}_x(g^r, \beta_{n_2} \cdot (g^r)^{-x_V}) = 1; \end{aligned}$$

2. $U \in \mathcal{I}_x, V \in \mathcal{I}_y$: Use the sets $\mathcal{X}, \mathcal{Y}$ to write $g^u = g^{x+x_U}$ and $g^v = g^{y+y_V}$ and check if

$$\begin{aligned} \exists n_1, n_2: & \text{GapDH}_y(g^{x+x_U}, \alpha_{n_1} \cdot (g^y)^{-y_V} \cdot g^{-x_U y_V}) = 1 \\ & \wedge \text{GapDH}_y(g^r, \beta_{n_2} \cdot (g^r)^{-y_V}) = 1; \end{aligned}$$

3. $U \in \mathcal{I}_y, V \in \mathcal{I}_x$: Use the sets $\mathcal{X}, \mathcal{Y}$ to write $g^u = g^{y+y_U}$ and $g^v = g^{x+x_V}$ and check if

$$\begin{aligned} \exists n_1, n_2: & \text{GapDH}_x(g^{y+y_U}, \alpha_{n_1} \cdot (g^y)^{-x_V} \cdot g^{-y_U x_V}) = 1 \\ & \wedge \text{GapDH}_x(g^r, \beta_{n_2} \cdot (g^r)^{-x_V}) = 1; \end{aligned}$$

4. $U, V \in \mathcal{I}_y$: Use the set $\mathcal{Y}$ to write $g^u = g^{y+y_U}$ and $g^v = g^{y+y_V}$ and check if

$$\begin{aligned} \exists n_1, n_2: & \text{GapDH}_y(g^{y+y_U}, \alpha_{n_1} \cdot (g^y)^{-y_V} \cdot g^{-y_U y_V}) = 1 \\ & \wedge \text{GapDH}_y(g^r, \beta_{n_2} \cdot (g^r)^{-y_V}) = 1; \end{aligned}$$

if such n_1, n_2 exist in one of these cases, set

$$k \leftarrow H(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, \alpha_{n_1}, \beta_{n_2}, k_{PQ}).$$

If not, filter out all existing entries of the form

$$(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, \beta_n, k_{PQ}, h_n) \in \mathcal{L}_{10} \text{ for } \beta_n \in \mathbb{G}$$

and query the DDH oracle depending on the following case.

1. $V \in \mathcal{I}_x$: Use the set $\mathcal{X}$ to write $g^v = g^{x+x_V}$ and check if

$$\exists n_0: \text{GapDH}_x(g^r, \beta_{n_0} \cdot (g^r)^{-x_V}) = 1;$$

2. $V \in \mathcal{I}_y$: Use the set $\mathcal{Y}$ to write $g^v = g^{y+y_V}$ and check if

$$\exists n_0: \text{GapDH}_y(g^r, \beta_{n_0} \cdot (g^r)^{-y_V}) = 1;$$

if such an n_0 exists in one of the cases, set $k \leftarrow h_{n_0}$. If not, check for an entry in $\mathcal{L}_{11}$, i.e., if $\exists h_{11}: (g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, k_{PQ}, h_{11}) \in \mathcal{L}_{11}$, set $k \leftarrow h_{11}$. Otherwise, draw a random $k \xleftarrow{\$} \{0, 1\}^\lambda$ and store $\mathcal{L}_{11} \leftarrow \mathcal{L}_{11} \cup \{(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, k_{PQ}, k)\}$.

Next, finalize the session variables by setting $\pi_V^j.\text{key} = k$, $\pi_V^j.\text{st}_{exec} = \text{accepted}$, and $\pi_V^j.\text{revealed} = \text{false}$.

NewSessionR(U, V, m) **for** $U \in \mathcal{I}_{\text{corr}}, V \in \mathcal{I}_{\text{corr}}, m \in \mathbb{G} \times \mathcal{C}_{PQ}$: Parse $(g^r, c_{PQ}) := m$. This query can be answered with an honest execution of $\Pi_{\text{OPKE}}.\text{Unwrap}(sk_V, pk_U, m)$, as the reduction knows the secret DH key v of identity V , so it is therefore able to compute g^{uv} for all identities $U \in \mathcal{I}$ and g^{rv} .

NewSessionR(U, V, m) **for** $U \in \mathcal{I}_{\text{corr}}, V \in \mathcal{I}_{\text{corr}}, m \in \mathbb{G} \times \mathcal{C}_{PQ}$: Parse $(g^u, pk_{U,PQ}) := pk_U$, $(g^v, pk_{V,PQ}) := pk_V$, the secret key $(u, sk_{U,PQ}) := sk_U$ and $(g^r, c_{PQ}) := m$ and execute $k_{PQ} \leftarrow \Pi_{PQ}.\text{Decaps}(sk_{V,PQ}, c_{PQ})$. We initialize a new receiver session π_V^j , and set $\pi_V^j.\text{role} = \text{receiver}$, $\pi_V^j.\text{id} = V$, $\pi_V^j.\text{pid} = U$, $\pi_V^j.\text{trans} = m$. If $k_{PQ} = \perp$, which means the decapsulation was not successful, we set $\pi_V^j.\text{st}_{exec} = \text{rejected}$, $\pi_V^j.\text{key} \leftarrow \perp$. On a successful decapsulation we proceed by filtering out all existing entries of the form $(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, g^{uv}, \beta_n, k_{PQ}, h_n) \in \mathcal{L}_H$ for $\beta_n \in \mathbb{G}$ and querying the DDH oracle of the challenger, where we distinguish two cases:

1. $V \in \mathcal{I}_x$: Use the set $\mathcal{X}$ to write $g^v = g^{x+uv}$ and check if

$$\exists n_0: \text{GapDH}_x(g^r, \beta_{n_0} \cdot (g^r)^{-xv}) = 1;$$

2. $V \in \mathcal{I}_y$: Use the set $\mathcal{Y}$ to write $g^v = g^{y+uv}$ and check if

$$\exists n_0: \text{GapDH}_y(g^r, \beta_{n_0} \cdot (g^r)^{-yv}) = 1;$$

if such an n_0 exists in one of these cases, set $k \leftarrow h_{n_0}$. If not, check for an entry in $\mathcal{L}_{01}$, i.e., if $\exists h_{01}: (g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, g^{uv}, k_{PQ}, h_{01}) \in \mathcal{L}_{01}$, set $k \leftarrow h_{01}$. Otherwise, draw a random $k \xleftarrow{\$} \{0, 1\}^\lambda$ and store

$$\mathcal{L}_{01} \leftarrow \mathcal{L}_{01} \cup \{(g^u, g^r, g^v, pk_{V,PQ}, c_{PQ}, g^{uv}, k_{PQ}, k)\}.$$

Next, finalize the session variables by setting $\pi_V^j.\text{key} = k$, $\pi_V^j.\text{st}_{exec} = \text{accepted}$, and $\pi_V^j.\text{revealed} = \text{false}$.

Reveal(π_U^i) **for** $U \in \mathcal{I}, i \in \{1, \dots, Q_S + Q_R\}$: Sessions which have the same transcript as the test session make π_A^t an invalid test session due to Definition 7, so we do not have to consider **Reveal**-queries for such sessions.

Thus, we have to consider sessions which have a different transcript than that of the test session. Recall that, as described above, for all such sessions, $\mathcal{B}$ simulates the **NewSessionS**(U, V)- and **NewSessionR**(U, V, m)-queries in a way such that also a key k is derived. Therefore $\mathcal{B}$ is able to answer corresponding **Reveal**-queries by setting $\pi_U^i.\text{revealed} = \text{true}$ and returning the session key k .

Corrupt(U) **for** $U \in \mathcal{I}_{\text{corr}}$: Since we may assume that the event $G := G_{\text{act}} \vee G_{\text{pass}}$ occurs, so in particular $\mathcal{C} \subseteq \mathcal{I}_{\text{corr}}$, this case cannot happen.

H(m) **for** $m \in \mathcal{M}$: Parse $(g_0, g_1, g_2, pk_{PQ}, c_{PQ}, \alpha, \beta, k_{PQ}) := m$ and sample $h \xleftarrow{\$} \{0, 1\}^\lambda$. If one of the following cases applies, h gets overwritten:

- If $\exists h': (g_0, g_1, g_2, pk_{PQ}, c_{PQ}, \alpha, \beta, k_{PQ}, h') \in \mathcal{L}_H$, i.e., the input $m \in \mathcal{M}$ has been queried to the random oracle before and **H**(m) is already defined, then $\mathcal{B}$ replies consistently by setting $h \leftarrow h'$.

- Else if $g_2 \in \{g^{x+x_U} \mid x_U \in \mathcal{X}\}$, i.e., check if $\exists x_2 \in \mathcal{X}: g_2 = g^{x+x_2}$, then:
 - If $g_1 \in \{g^{y+\delta} \mid (\delta, g^y) \in \mathcal{D}\}$, i.e., check if $\exists (\delta_1, g^y) \in \mathcal{D}: g_1 = g^{y+\delta_1}$, then:
 - * If $\text{GapDH}_x(g^{y+\delta_1}, \beta \cdot (g^y)^{-x_2} \cdot g^{-\delta_1 x_2}) = 1$, set

$$\gamma \leftarrow \beta \cdot (g^x)^{-\delta_1} \cdot (g^y)^{-x_2} \cdot g^{-\delta_1 x_2}$$

and output γ as the 2-GapDH solution and terminate.

- If $g_0 \in \{g^{x+x_U} \mid x_U \in \mathcal{X}\}$, i.e., check if $\exists x_0 \in \mathcal{X}: g_0 = g^{x+x_0}$, then:
 - * If $\text{GapDH}_x(g_0, \alpha \cdot (g^x)^{-x_2} \cdot g^{-x_0 x_2}) = 1$, check for an entry in $\mathcal{L}_{10}$.
 - If $\exists h_{10}: (g_0, g_1, g_2, pk_{\text{PQ}}, c_{\text{PQ}}, \beta, k_{\text{PQ}}, h_{10}) \in \mathcal{L}_{10}$, set $h \leftarrow h_{10}$.
 - Else if

$$\text{GapDH}_x(g_1, \beta \cdot g_1^{-x_2}) = 1$$

$$\wedge \exists h_{11}: (g_0, g_1, g_2, pk_{\text{PQ}}, c_{\text{PQ}}, k_{\text{PQ}}, h_{11}) \in \mathcal{L}_{11},$$

set $h \leftarrow h_{11}$.

- Else if $g_0 \in \{g^{y+y_U} \mid y_U \in \mathcal{Y}\}$, i.e., check if $\exists y_0 \in \mathcal{Y}: g_0 = g^{y+y_0}$, then:
 - * If $\text{GapDH}_x(g_0, \alpha \cdot (g^y)^{-x_2} \cdot g^{-y_0 x_2}) = 1$, set

$$\gamma \leftarrow \alpha \cdot (g^x)^{-y_0} \cdot (g^y)^{-x_2} \cdot g^{-y_0 x_2}$$

and output γ and terminate.

- Else if $g_2 \in \{g^{y+y_U} \mid y_U \in \mathcal{Y}\}$, i.e., check if $\exists y_2 \in \mathcal{Y}: g_2 = g^{y+y_2}$, then:
 - If $g_1 \in \{g^{x+\delta} \mid (\delta, g^x) \in \mathcal{D}\}$, i.e., check if $\exists (\delta_1, g^x) \in \mathcal{D}: g_1 = g^{x+\delta_1}$, then:
 - * If $\text{GapDH}_y(g^{x+\delta_1}, \beta \cdot (g^x)^{-y_2} \cdot g^{-\delta_1 y_2}) = 1$, set

$$\gamma \leftarrow \beta \cdot (g^y)^{-\delta_1} \cdot (g^x)^{-y_2} \cdot g^{-\delta_1 y_2}$$

and output γ and terminate.

- If $g_0 \in \{g^{x+x_U} \mid x_U \in \mathcal{X}\}$, i.e., check if $\exists x_0 \in \mathcal{X}: g_0 = g^{x+x_0}$, then:
 - * If $\text{GapDH}_y(g_0, \alpha \cdot (g^x)^{-y_2} \cdot g^{-x_0 y_2}) = 1$, set

$$\gamma \leftarrow \alpha \cdot (g^x)^{-y_2} \cdot (g^y)^{-x_0} \cdot g^{-x_0 y_2}$$

and output γ and terminate.

- Else if $g_0 \in \{g^{y+y_U} \mid y_U \in \mathcal{Y}\}$, i.e., check if $\exists y_0 \in \mathcal{Y}: g_0 = g^{y+y_0}$, then:
 - * If $\text{GapDH}_y(g_0, \alpha \cdot (g^y)^{-y_2} \cdot g^{-y_0 y_2}) = 1$, check for an entry in $\mathcal{L}_{10}$.
 - If $\exists h_{10}: (g_0, g_1, g_2, pk_{\text{PQ}}, c_{\text{PQ}}, \beta, k_{\text{PQ}}, h_{10}) \in \mathcal{L}_{10}$, set $h \leftarrow h_{10}$.
 - Else if

$$\text{GapDH}_y(g_1, \beta \cdot g_1^{-y_2}) = 1$$

$$\wedge \exists h_{11}: (g_0, g_1, g_2, pk_{\text{PQ}}, c_{\text{PQ}}, k_{\text{PQ}}, h_{11}) \in \mathcal{L}_{11},$$

set $h \leftarrow h_{11}$.

Finally, store $\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{(g_0, g_1, g_2, pk_{\text{PQ}}, c_{\text{PQ}}, \alpha, \beta, k_{\text{PQ}}, h)\}$ and return h .

Test(π_A^t): If $\pi_A^t.\text{key} = \perp$, return $\perp$; otherwise, flip a fair coin $d_{\text{test}} \xleftarrow{\$} \{0, 1\}$. If $d_{\text{test}} = 0$, set $k^* \leftarrow \pi_A^t.\text{key}$, else set $k^* \xleftarrow{\$} \{0, 1\}^\lambda$. Return k^* .

Finally, $\mathcal{A}$ returns a bit $d^* \in \{0, 1\}$. In order for $\mathcal{A}$ to win with non-zero advantage, the random oracle must have been queried with the appropriate $m \in \mathcal{M}$, so that $\mathcal{B}$ sets $\gamma \leftarrow g^{xy}$, which is returned to the challenger of the 2-Gap-DH experiment. Note that the value for γ can be set using one of the δ , which is only generated in $\text{NewSessionS}(U, V)$ -queries for $U \in \mathcal{I}_{\text{corr}}, V \in \mathcal{I}_{\text{corr}}$. Even if we have $\pi_A^t.\text{role} = \text{receiver}$ and $B \in \mathcal{C} \subseteq \mathcal{I}_{\text{corr}}$, there must be a partnered sender session according to Definition 7, where the corresponding δ has been generated. If the value of γ has not been set during an RO query, $\mathcal{B}$ guesses a value before returning it to the challenger: $\gamma \xleftarrow{\$} \{0, 1\}^\lambda$. Note that $\mathcal{B}$ perfectly simulates Game 1 for $\mathcal{A}$. Let H be the event that $\mathcal{A}$ queries the appropriate $m \in \mathcal{M}$ to the random oracle. If H occurs, the reduction wins the 2-Gap-DH experiment, i.e., $\Pr[H] \leq \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B})$. If H does not occur, the adversary's capability of winning is reduced to guessing, as both keys in the Test-query appear random due to the RO. Therefore,

$$\Pr[S_1] = \Pr[S_1 \mid H] \cdot \Pr[H] + \Pr[S_1 \mid \overline{H}] \cdot \Pr[\overline{H}] \leq 1 \cdot \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B}) + \frac{1}{2} \cdot 1,$$

which yields

$$\text{Adv}_1(\mathcal{A}) = |2 \cdot \Pr[S_1] - 1| \leq \left| 2 \cdot \left(\text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B}) + \frac{1}{2} \right) - 1 \right| = 2 \cdot \text{Adv}_{\mathbb{G},g}^{2\text{-GapDH}}(\mathcal{B}).$$

□

5 Conclusion

We presented a formal cryptographic analysis of TUTACRYPT, a hybrid one-pass key establishment protocol that is deployed today to protect email communication of more than ten million users worldwide against store-now-break-later attacks. Such future quantum attacks are ruled out by our weak-OPKE-security proof, assuming that the underlying KEM remains secure against quantum attacks.

Store-now-break-later attacks usually consider a fully *passive* adversary, which only *observes* protocol executions, stores the transcripts, and then applies the quantum attack to the stored transcripts in the future. We show something slightly stronger, by allowing the quantum attacker to be *active* in protocol sessions, except for the tested session. Disallowing an active attack on the tested session is necessary, because the quantum-secure key exchange only uses a KEM and an active adversary can always encapsulate a key honestly, like an honest user, and then test the corresponding session.

By expressing the protocol in a precise cryptographic form and introducing two Bellare–Rogaway-style security models, we captured the distinct guarantees that arise in the classical and post-quantum threat settings. In more detail, we do not think that the different security levels achieved pose a threat to TUTACRYPT. The 2-GapDH-based proof additionally allows the adversary to compute the message that is sent to a receiver Test-session by itself, provided that the sender's long-term key is not corrupted. In the KEM-based proof we cannot allow the adversary to encapsulate the tested key by itself (which seems natural for KEM-based key transport). Hence, the 2-GapDH proof achieves a slightly stronger security notion. On the one hand, for most practical attacker models this probably only makes a minor difference (in particular, security against store-now-break-later is achieved, which is the main purpose of the hybrid design of TUTACRYPT), however, we still find it interesting to point it out because it provides a better understanding of the exact security of TUTACRYPT in the pre- and post-quantum setting. Our reduction-based proofs clarify exactly which computational hardness assumptions are required for each security notion, and the concrete security loss of the proofs. We view such results as a step towards bridging the gap between practical custom designs and principled, verifiable cryptographic design during the post-quantum transition.

Acknowledgments

We would like to thank Kai Gellert for fruitful discussions when developing the security model and thank the anonymous reviewers for their constructive comments, as well as the Tuta team for explaining and discussing the TUTACRYPT protocol and its design. This work is supported by the Bundesministerium für Forschung, Technologie und Raumfahrt (BMFTR) under the project number 16KIS1802 (PQDrive).

References

- [ADJ25] Martin R. Albrecht, Benjamin Dowling, and Daniel Jones. Formal analysis of multi-device group messaging in WhatsApp. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 242–271. Springer, Cham, May 2025. doi:[10.1007/978-3-031-91101-9_9](https://doi.org/10.1007/978-3-031-91101-9_9).
- [BCD⁺24] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, and Bas Westerbaan. X-Wing. *CiC*, 1(1):21, 2024. doi:[10.62056/a3qj89n4e](https://doi.org/10.62056/a3qj89n4e).
- [BDF⁺11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Berlin, Heidelberg, December 2011. doi:[10.1007/978-3-642-25385-0_3](https://doi.org/10.1007/978-3-642-25385-0_3).
- [BFK⁺14] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella Béguelin. Proving the TLS handshake secure (as it is). In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 235–255. Springer, Berlin, Heidelberg, August 2014. doi:[10.1007/978-3-662-44381-1_14](https://doi.org/10.1007/978-3-662-44381-1_14).
- [BHJ⁺15] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Berlin, Heidelberg, March 2015. doi:[10.1007/978-3-662-46494-6_26](https://doi.org/10.1007/978-3-662-46494-6_26).
- [BJS15] Florian Bergsma, Tibor Jager, and Jörg Schwenk. One-round key exchange with strong security: An efficient and generic construction in the standard model. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 477–494. Springer, Berlin, Heidelberg, March / April 2015. doi:[10.1007/978-3-662-46447-2_21](https://doi.org/10.1007/978-3-662-46447-2_21).
- [BKN02] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 1–11. ACM Press, November 2002. doi:[10.1145/586110.586112](https://doi.org/10.1145/586110.586112).
- [BMS20] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment, Second Edition*. Information Security and Cryptography. Springer, 2020. doi:[10.1007/978-3-662-58146-9](https://doi.org/10.1007/978-3-662-58146-9).
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:[10.1145/168588.168596](https://doi.org/10.1145/168588.168596).

- [BRTZ24] Mihir Bellare, Doreen Riepel, Stefano Tessaro, and Yizhao Zhang. Count corruptions, not users: Improved tightness for signatures, encryption and authenticated key exchange. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 326–360. Springer, Singapore, December 2024. doi:[10.1007/978-981-96-0888-1_11](https://doi.org/10.1007/978-981-96-0888-1_11).
- [CCD⁺20] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, October 2020. doi:[10.1007/s00145-020-09360-1](https://doi.org/10.1007/s00145-020-09360-1).
- [Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Berlin, Heidelberg, August 2000. doi:[10.1007/3-540-44598-6_14](https://doi.org/10.1007/3-540-44598-6_14).
- [Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Berlin, Heidelberg, April / May 2002. doi:[10.1007/3-540-46035-7_18](https://doi.org/10.1007/3-540-46035-7_18).
- [DFG⁺23] Gareth T. Davies, Sebastian H. Faller, Kai Gellert, Tobias Handirk, Julia Hesse, Máté Horváth, and Tibor Jager. Security analysis of the WhatsApp end-to-end encrypted backup protocol. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 330–361. Springer, Cham, August 2023. doi:[10.1007/978-3-031-38551-3_11](https://doi.org/10.1007/978-3-031-38551-3_11).
- [DJK⁺25] Yevgeniy Dodis, Daniel Jost, Shuichi Katsumata, Thomas Prest, and Rolfe Schmidt. Triple ratchet: A bandwidth efficient hybrid-secure signal protocol. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 302–331. Springer, Cham, May 2025. doi:[10.1007/978-3-031-91101-9_11](https://doi.org/10.1007/978-3-031-91101-9_11).
- [DJSS18] David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 425–455. Springer, Cham, April / May 2018. doi:[10.1007/978-3-319-78372-7_14](https://doi.org/10.1007/978-3-319-78372-7_14).
- [FG25] Rune Fiedler and Felix Günther. Security analysis of Signal’s PQXDH handshake. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part II*, volume 15675 of *LNCS*, pages 137–169. Springer, Cham, May 2025. doi:[10.1007/978-3-031-91823-0_5](https://doi.org/10.1007/978-3-031-91823-0_5).
- [FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Berlin, Heidelberg, February / March 2013. doi:[10.1007/978-3-642-36362-7_17](https://doi.org/10.1007/978-3-642-36362-7_17).
- [GHJL17] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 519–548. Springer, Cham, April / May 2017. doi:[10.1007/978-3-319-56617-7_18](https://doi.org/10.1007/978-3-319-56617-7_18).
- [GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769

- of *LNCS*, pages 190–218. Springer, Cham, March 2018. doi:[10.1007/978-3-319-76578-5_7](https://doi.org/10.1007/978-3-319-76578-5_7).
- [GJ18] Kristian Gjøsteen and Tibor Jäger. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Cham, August 2018. doi:[10.1007/978-3-319-96881-0_4](https://doi.org/10.1007/978-3-319-96881-0_4).
- [GKS13] Florian Giesen, Florian Kohlar, and Douglas Stebila. On the security of TLS renegotiation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 387–398. ACM Press, November 2013. doi:[10.1145/2508859.2516694](https://doi.org/10.1145/2508859.2516694).
- [HJLS17] Britta Hale, Tibor Jäger, Sebastian Lauer, and Jörg Schwenk. Simple security definitions for and constructions of 0-RTT key exchange. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 2017*, volume 10355 of *LNCS*, pages 20–38. Springer, Cham, July 2017. doi:[10.1007/978-3-319-61204-1_2](https://doi.org/10.1007/978-3-319-61204-1_2).
- [JK02] Jakob Jonsson and Burton S. Kaliski, Jr. On the security of RSA encryption in TLS. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 127–142. Springer, Berlin, Heidelberg, August 2002. doi:[10.1007/3-540-45708-9_9](https://doi.org/10.1007/3-540-45708-9_9).
- [JK25] Joseph Jaeger and Akshaya Kumar. Analyzing group chat encryption in MLS, session, signal, and matrix. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 272–301. Springer, Cham, May 2025. doi:[10.1007/978-3-031-91101-9_10](https://doi.org/10.1007/978-3-031-91101-9_10).
- [JKRS21] Tibor Jäger, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 117–146. Springer, Cham, October 2021. doi:[10.1007/978-3-030-77870-5_5](https://doi.org/10.1007/978-3-030-77870-5_5).
- [JKSS12] Tibor Jäger, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 273–293. Springer, Berlin, Heidelberg, August 2012. doi:[10.1007/978-3-642-32009-5_17](https://doi.org/10.1007/978-3-642-32009-5_17).
- [KPW13] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 429–448. Springer, Berlin, Heidelberg, August 2013. doi:[10.1007/978-3-642-40041-4_24](https://doi.org/10.1007/978-3-642-40041-4_24).
- [KS24] Ehren Kret and Rolfe Schmidt. The PQXDH Key Agreement Protocol. <https://signal.org/docs/specifications/pqxdh/>, January 2024. Accessed: 16.07.2026.
- [LLGW20] Xiangyu Liu, Shengli Liu, Dawu Gu, and Jian Weng. Two-pass authenticated key exchange with explicit authentication and tight security. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 785–814. Springer, Cham, December 2020. doi:[10.1007/978-3-030-64834-3_27](https://doi.org/10.1007/978-3-030-64834-3_27).

- [MSW08] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. A modular security analysis of the TLS handshake protocol. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 55–73. Springer, Berlin, Heidelberg, December 2008. doi:10.1007/978-3-540-89255-7_5.
- [Nat24] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard, 2024. URL: <https://doi.org/10.6028/NIST.FIPS.203>.
- [SAB⁺22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>.
- [Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365700.
- [Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. URL: <https://eprint.iacr.org/2004/332>.