Post-quantum Cryptographic Analysis of SSH

Benjamin Benčina
Royal Holloway, University of London, UK
Email: benjamin.bencina.2022@live.rhul.ac.uk

Benjamin Dowling
King's College London, UK
Email: benjamin.dowling@kcl.ac.uk

Varun Maram
SandboxAQ, UK
Email: varun.maram@sandboxaq.com

Keita Xagawa
Technology Innovation Institute, UAE
Email: keita.xagawa@tii.ae

***Abstract*—The Secure Shell (SSH) protocol is one of the first security protocols on the Internet to upgrade itself to resist attacks against future quantum computers, with the default adoption of the "quantum (otherwise, classically)" secure *hybrid* key exchange in OpenSSH from April 2022. However, there is a lack of a comprehensive security analysis of this quantum-resistant version of SSH in the literature: related works either focus on the hybrid key exchange *in isolation* and do not consider security of the overall protocol, or analyze the protocol in security models which are not appropriate for SSH, especially in the "post-quantum" setting.**

In this paper, we remedy the state of affairs by providing a thorough post-quantum cryptographic analysis of SSH. We follow a "top-down" approach wherein we first prove security of SSH in a more appropriate model, namely, our post-quantum extension of the so-called *authenticated and confidential channel establishment (ACCE)* protocol security model; our extension which captures "*harvest now, decrypt later*" attacks could be of independent interest. Then we establish the cryptographic properties of SSH's underlying primitives, as concretely instantiated in practice, based on our protocol-level ACCE security analysis: for example, we prove relevant cryptographic properties of "*Streamlined NTRU Prime*", a key encapsulation mechanism (KEM) which is used in recent versions of OpenSSH and TinySSH, in the *quantum random oracle model*, and address open problems related to its analysis in the literature. Notably, our ACCE security analysis of post-quantum SSH relies on the weaker notion of *IND-CPA* security of the ephemeral KEMs used in the hybrid key exchange. This is in contrast to prior works which rely on the *stronger* assumption of *IND-CCA* secure ephemeral KEMs. Hence we conclude the paper with a discussion on potentially replacing IND-CCA secure KEMs in current post-quantum implementations of SSH with simpler and faster IND-CPA secure counterparts, and also provide the corresponding benchmarks.

1. Introduction

To counter the threat of quantum computers breaking most currently deployed cryptography, there is an ongoing effort to transition important communication protocols on the Internet from using quantum-vulnerable cryptographic algorithms to their quantum-resilient counterparts. The Secure Shell (SSH) protocol — which is widely used for remote access to servers, as well as virtual private network (VPN) access — is one such protocol. In fact, recent versions of OpenSSH [1], the most prominent SSH implementation in practice, use quantum-secure key exchange methods *by default*. Because of the widespread adoption of these "post-quantum" transitioned protocols, it is important to formally analyze their cryptographic properties in appropriate quantum security models.

In post-quantum implementations of SSH (and other protocols such as the Transport Layer Security (TLS) protocol and Signal), the key exchange mechanism technically follows the so-called *hybrid* paradigm wherein the quantum-secure key exchange method (or technically, *key encapsulation mechanism (KEM)*) does not replace but instead is used in combination with the quantum-vulnerable method (e.g., that based on the Diffie-Hellman (DH) setting) such that the resulting mechanism is secure as long as one of these two methods is; in a way, these methods act as "backstops" against any potential weaknesses in their counterparts. And if we look at the handful of prior works which analyze the post-quantum security of such SSH implementations, most of them (e.g., [2], [3]) only focus on this hybrid key exchange primitive *in isolation*, and do not extend their analysis to capture security of the entire protocol.

To the best of our knowledge, the work which comes close to providing a protocol-level cryptographic analysis of post-quantum SSH is by Blanchet and Jacomme [4]. More specifically, the authors take a machine-aided approach to provide formal security guarantees for post-quantum versions of SSH (and also TLS). However in their security proofs, SSH was analyzed using close variants of the standard *authenticated key exchange (AKE)* models [5]. And as explained in [6], AKE models are *not* appropriate for modeling security

of SSH for various reasons.[1]

Finally, if we go back to the hybrid key exchange used in post-quantum implementations of SSH — notably, OpenSSH and TinySSH [7] — the quantum-secure KEM is instantiated with "*Streamlined NTRU Prime*" [8], a scheme which was submitted to NIST's post-quantum cryptography (PQC) standardization project; in fact, Streamlined NTRU Prime was used in the default key exchange method in OpenSSH from version 9.0 (OpenSSH also uses the new NIST PQC standard "*ML-KEM*" [9] from version 9.9). However as pointed out in [10], Streamlined NTRU Prime evades a formal security proof in the post-quantum setting — particularly, in the so-called "*Quantum Random Oracle Model (QROM)*" [11] — because of certain design choices (we will expand upon this in Subsection 4.2.1 below). This only adds to the list of issues with regards to a thorough cryptographic analysis of post-quantum SSH.

1.1. Our Contributions

In this work, we provide a *comprehensive* post-quantum cryptographic analysis of SSH in appropriate security models, thereby addressing the above issues in the literature. We follow a "top-down" approach in our analysis wherein we first establish post-quantum security of SSH at a protocol level, while relying on certain cryptographic properties of the underlying primitives; we then prove that concrete post-quantum instantiations of these primitives in practice do satisfy the corresponding properties. Our results can be summarized as follows:

Post-quantum ACCE security of SSH. As pointed above, AKE models are not suitable to model security of SSH. Instead, as explained in [6], the notion of so-called *authenticated and confidential channel establishment (ACCE)* security is more appropriate. The authors of [6] also establish ACCE security of SSH — albeit in the *classical* Diffie-Hellman setting. To analyze SSH in the post-quantum setting, we first extend the ACCE security definition to capture the highly relevant "*harvest now, decrypt later*" attacks; we believe our extension is of independent interest. Then we formally prove security of SSH in our enhanced post-quantum ACCE model. We also capture *forward secrecy* in our proof, unlike the prior classical analysis of SSH in [6].

Analysis of SSH's hybrid KEX, and Streamlined NTRU Prime. Our post-quantum ACCE security analysis of SSH relies on the underlying hybrid key exchange (KEX) satisfying the standard notion of *IND-CPA* security. We hence establish IND-CPA security of the specific hybrid KEX methods used in post-quantum SSH in the appropriate QROM setting; prior works which focus on such hybrid KEX (e.g., [3]) consider the *classical* ROM [12] in contrast. And as mentioned earlier, most post-quantum implementations of SSH in practice instantiate the quantum-secure component of their hybrid KEX with Streamlined NTRU Prime. In the context of SSH, we therefore tightly prove IND-CPA security of Streamlined NTRU Prime in the QROM. At the same time, we also describe a way to establish the stronger notion of *IND-CCA* security for Streamlined NTRU Prime in the QROM, thereby addressing an open problem in [10]. Such an IND-CCA security proof provides assurance against attacks which may exploit ephemeral keys being *reused* in bad hybrid KEX implementations in SSH.

QROM security of SSH PRF. Following the hybrid key exchange in the handshake phase, session keys in post-quantum SSH are derived using a specific function denoted as $\mathsf{PRF}_{\mathsf{SSH}}$ in this paper. Our ACCE security analysis of SSH in the post-quantum setting also relies on $\mathsf{PRF}_{\mathsf{SSH}}$ being a secure *pseudorandom function (PRF)*. Now $\mathsf{PRF}_{\mathsf{SSH}}$ is a relatively complicated construction (see Figure 5 for a detailed description) which involves nested invocations of a hash function. And as pointed out in [6], analyzing the PRF security of $\mathsf{PRF}_{\mathsf{SSH}}$ using standard assumptions on the underlying hash function (such as *collision resistance*) is either not possible or highly challenging. Hence in their classical ACCE security analysis of SSH, the authors of [6] make a non-standard "monolithic" assumption that $\mathsf{PRF}_{\mathsf{SSH}}$ is a secure PRF, and do not go further. At the same time, they mention that if the hash function is (heuristically) modeled as a *random oracle* (i.e., in the classical ROM [12] setting), then it can be proven that $\mathsf{PRF}_{\mathsf{SSH}}$ is a secure PRF. Hence to complement our post-quantum ACCE security analysis, we model the hash function appropriately as a *quantum random oracle* — namely, where attackers can query such functions in *quantum superposition* — and formally establish the security of $\mathsf{PRF}_{\mathsf{SSH}}$ in the QROM.

Benchmarking IND-CPA v/s IND-CCA secure KEMs in post-quantum SSH. Going back to the protocol-level analysis of post-quantum SSH by Blanchet and Jacomme [4], they rely on the stronger assumption of IND-CCA security of the *ephemeral* quantum-secure KEM in the handshake phase; the authors mention they need IND-CCA security to prove forward secrecy in their model. But as mentioned above, the AKE-like model used in [4] is not suitable to capture security of SSH, and we believe the IND-CCA security requirement is an artifact of their proofs. In contrast, our post-quantum ACCE security analysis of SSH suggests that the simpler IND-CPA secure ephemeral KEMs are sufficient — even for forward secrecy.[2] Now in post-quantum implementations of SSH, the quantum-secure KEMs (e.g., Streamlined NTRU Prime, ML-KEM) are instantiated with certain "transforms" (we describe them in more detail in Subsection 4.2.1) to achieve IND-CCA security; these transforms however result in some performance overheads — e.g., due to extra hash calls. In light of our ACCE analysis, we instantiate the above ephemeral KEMs without these IND-CCA transforms and evaluate the resulting performance in the post-quantum SSH protocol. Even though we notice a significant runtime improvement when such IND-CPA

1. One reason is that AKE security requires indistinguishability of session keys in the protocol following the initial key exchange; however in SSH, an adversary can trivially distinguish real session keys from random using so-called "key confirmation" messages sent during the handshake phase.

2. Our result also follows some recent works, e.g., [13], [14], which claim that IND-CPA secure KEMs are sufficient in post-quantum TLS; however, their new security proofs for TLS incur a significant loss in tightness in contrast to our analysis of post-quantum SSH.

secure KEMs are used *in isolation*, when compared to their transformed IND-CCA secure counterparts, we observe that the resulting improvements in the overall protocol are minimal.[3] At the same time, we are concretely saving on the number of hash calls during the protocol execution on both the server and client sides, and this should result in reduced financial and ecological costs: for example, as mentioned in Meta's post-quantum transition update at the "Real World PQC Workshop" [15], an extra hash "*can cost hundreds of thousands or even millions of dollars a year*". So all in all, we hope our benchmarks serve as a guide to implementers of the post-quantum SSH protocol with regards to reducing the associated costs.

2. Preliminaries

2.1. Notations

For a positive integer n, we denote $[n]$ to be the set $\{1, 2, \ldots, n\}$. We denote $\lambda \in \mathbb{N}$ to be the security parameter, unless stated otherwise (but we omit writing λ when it is clear from context). For a finite set S, we write $x \leftarrow\!\!\$\ S$ to denote that x is uniformly at random sampled from S, unless stated otherwise; also $|S|$ denotes the size of S. $x \,\|\, y$ denotes the concatenation of bit strings x and y. For probabilistic algorithms we use $y \leftarrow \mathcal{A}(x)$ to denote a (randomized) output of $\mathcal{A}$ on input x; we also sometimes specify the randomness r used in $\mathcal{A}$ as $y \leftarrow \mathcal{A}(x; r)$.

2.2. ACCE

An Authenticated and Confidential Channel Establishment (ACCE) protocol is a two-phase communication protocol consisting of a handshake phase (establishing secrets and authenticating parties) and a communication phase (using secrets to authentically encrypt messages). The original ACCE SSH analysis [6] defines two variants of ACCE protocols: a standard ACCE formulation, and a multi-ciphersuite ACCE protocol. In what follows we extend the standard ACCE formalism to capture post-quantum security. Each execution of the protocol is called a *session* and will maintain and update the following *per-session variables*.

Definition 2.1 (Per-session variables)**.** *Let π denote the following collection of* per-session variables*:*

- $\rho \in \{\texttt{init}, \texttt{resp}\}$*: The party's role in this session.*
- $\mathsf{pid} \in \{1, \ldots, n_P, \perp\}$*: The identifier of the alleged peer of this session, or $\perp$ for unauthenticated.*
- $\mathsf{status} \in \{\texttt{active}, \texttt{reject}, \texttt{accept}\}$*: The status.*
- *k: A session key, or $\perp$. Note that k consists of two sub-keys: bi-directional authenticated encryption keys k_e and k_d (which themselves may consist of encryption and MAC sub-keys).*
- sid*: A session identifier defined by the protocol.*
- *st_e, st_d: State for the stateful authenticated encryption and decryption algorithms.*
- *Any additional state specific to the protocol.*
- *Any additional state for the security experiment.*

We now define and formalise the algorithms comprising an ACCE protocol.

Definition 2.2 (ACCE protocol)**.** *An* ACCE protocol *is a tuple of algorithms. The key generation algorithm* $\mathsf{KeyGen}() \xrightarrow{\$} (sk, pk)$ *outputs a long-term secret key / public key pair. The handshake algorithms* Algl_ℓ *and* AlgR_ℓ*, $\ell = 1, \ldots,$ take as input (sk, pk) and an incoming message m, update per-session variables π, and output an outgoing message m'. The handshake algorithms eventually set the variables for the peer identifier $\pi.\mathsf{pid}$, the session status $\pi.\mathsf{status}$, the session key $\pi.k$, and the session identifier $\pi.\mathsf{sid}$. There are also* stateful authenticated encryption and decryption algorithms $\mathsf{Enc}(\pi.k_e, m, \pi.st_e) \xrightarrow{\$} (C, \pi.st_e)$ *and* $\mathsf{Dec}(\pi.k_d, C, \pi.st_d) \rightarrow (m', \pi.st_d)$*. All algorithms are assumed to take as implicit input any global protocol parameters, including the list of all trusted peer public keys.*

2.3. Execution Environment

We now define the experiments used to capture the security of ACCE protocols. Our ACCE experiments mostly follow those of the original ACCE SSH analysis [6], but we extend the execution environment of ACCE protocols by introducing a new adversarial query $\mathsf{OExecute}$, which is used to execute ACCE protocols between two honest parties. This allows us to capture Harvest-Now-Decrypt-Later attacks for our post-quantum ACCE security experiment.

Parties and long-term key generation. The execution environment consists of n_P parties, $P_1, \ldots, P_{n_P}$, each a potential protocol participant. Each party P_i generates long-term private key / public key pairs (sk_i, pk_i) using $\mathsf{KeyGen}()$.

Sessions. Each party can execute multiple sessions of the protocol, either concurrently or subsequently. We will denote the s-th session of a protocol at party P_i by π_i^s, where $s \in \{1, \ldots, n_S\}$. We overload the notation so that π_i^s also denotes the per-session variables π for this session. Each session within a party has read access to the party's long-term keys. The per-session variables $\pi_i^j.(\mathsf{pid}, \mathsf{status}, k, \mathsf{sid})$ are initialized to $(\perp, \texttt{active}, \perp, \perp)$.

For the purposes of defining ciphertext indistinguishability and integrity, each session upon initialization chooses a uniform random bit $\pi_i^s.b \xleftarrow{\$} \{0, 1\}$. Each session also maintains additional variables for stateful encryption/decryption as required in Figure 1.

Adversary interaction. The adversary controls all communications between parties: it directs parties to initiate sessions, delivers messages to parties, and can reorder, alter,

3. It is worth pointing out that similar benchmarks done in the literature in the context of post-quantum TLS (e.g., in [13], [14]) only focus on "primitive-level" performance improvements in KEMs, when the above IND-CCA transforms are removed, but do not consider improvements in the overall protocol.

delete, and create messages. The adversary can also compromise certain long-term and per-session values of parties. The adversary interacts with parties using the following queries.

The first two queries model normal, unencrypted communication of parties during session establishment.

- $\mathsf{OSend}(i, s, m) \to m'$: The adversary sends message m to session π_i^s. Party P_i processes message m according to the protocol specification and its per-session state π_i^s, updates its per-session state, and optionally outputs an outgoing message m'. There is a distinguished initialization message which allows the adversary to activate the session with certain information. In particular, the initialization message consists of: the role ρ the party is meant to play in this session; and optionally the identity pid of the intended partner of this session. This query returns $\perp$ if the session has set status $=$ `accept` and no more protocol messages are transmitted over the unencrypted channel.
- $\mathsf{OExecute}(i, s, j, t) :\to \vec{m}$: Sessions π_i^s and π_j^t execute the channel establishment protocol without modification from the adversary. Internally, this is achieved by the challenger calling $\mathsf{OSend}(i, s, \rho \| \mathsf{pid}) \to m$, $\mathsf{OSend}(j, t, m) \to m' \ldots$ until both sessions have set status $=$ `accept` and returned $\perp$.

The next two queries model adversarial compromise of long-term and per-session secrets.

- $\mathsf{OReveal}(i, s) \to k$: Returns session key $\pi_i^s.k$.
- $\mathsf{OCorrupt}(i) \to sk$: Returns party P_i's long-term secret key sk_i, and can now impersonate P_i in later sessions.

The final two queries model communication over the encrypted channel. The adversary can cause plaintexts to be encrypted as outgoing ciphertexts, and can cause ciphertexts to be delivered and decrypted as incoming plaintexts.

- $\mathsf{OEncrypt}(i, s, m_0, m_1) \to C$: This query takes as input two messages m_0 and m_1. If $\pi_i^s.k = \perp$, the query returns $\perp$. Otherwise, it proceeds as in Figure 1, depending on the random bit $\pi_i^s.b$ sampled by π_i^s at the beginning of the game and the state variables of π_i^s.
- $\mathsf{ODecrypt}(i, s, C) \to m$ or $\perp$: This query takes as input a ciphertext C. If $\pi_i^s.k = \perp$, the query returns $\perp$. Otherwise, it proceeds as in Figure 1. Note in particular that decryption can be buffered, meaning a decryption state may be maintained containing unprocessed bytes of a partial ciphertext.

Together, these two oracles model the *buffered stateful authenticated encryption (BSAE)* notion, which simultaneously captures (i) indistinguishability under chosen ciphertext attack, (ii) integrity of ciphertexts, and (iii) buffered in-order delivery of ciphertexts. The hidden bit $\pi_i^s.b$ is leaked to the adversary if any of these goals is violated.

$\mathsf{OEncrypt}(i, s, m_0, m_1)$	$\mathsf{ODecrypt}(i, s, C)$
1 : $u \leftarrow u + 1$	1 : $(j, t) \leftarrow \pi_i^s.\mathsf{pid}$, $v \leftarrow v + 1$
2 : $(C^{(0)}, st_e^0) \leftarrow \mathsf{Enc}(k_e, m_0, st_e^0)$	2 : $(m, st_d) \leftarrow \mathsf{Dec}(k_d, C, st_d)$
3 : $(C^{(1)}, st_e^1) \leftarrow \mathsf{Enc}(k_e, m_1, st_e^1)$	3 : **if** $m = \perp_p$ **then return** $\perp$
4 : **if** $C^{(0)} = \perp$ or $C^{(1)} = \perp$ **then**	4 : **if** $b = 0$ **then return** $\perp$
5 : **return** $\perp$	5 : **if** $v > \pi_j^t.u$ or $C \neq \pi_j^t.C[v]$ **then**
6 : $C[u] \leftarrow C^{(b)}$	6 : $\mathsf{phase} \leftarrow 1$
7 : **return** $C^{(b)}$	7 : **if** $\mathsf{phase} = 1$ **then return** m
	8 : **return** $\perp$

Figure 1. OEncrypt and ODecrypt queries in the multi-ciphersuite ACCE security experiment.

Note that $b, C[], k_d, k_e, st_d, st_e, u, v$ denote the values stored in the per-session variables π_i^s. Although $\pi_i^s.\mathsf{pid}$ only contains the party identifier j, once π_i^s has accepted every session π_i^s has a unique matching session π_j^t known to the challenger. The ODecrypt query accounts for buffering in the third line; this is the difference from ACCE's original stateful length-hiding definition [16], [17].

2.4. Post-quantum Security Definitions

The security of ACCE protocols require that (i) the handshake is a secure authentication protocol, and (ii) the encrypted channel provides authenticated and confidential communication Security in ACCE protocols is captured through a game played between an adversary and a challenger. When considering authentication, the adversary's goal is to cause a session to accept without an *honest matching partner*. Additionally, we consider the security of the channel protocol, where an adversary's goal is to distinguish between the encryption of distinct plaintext messages.

The post-quantum security of ACCE protocols are captured slightly different, since we are specifically interested in *Harvest-Now-Decrypt-Later* attacks in which classical adversaries can interact with the sessions while the sessions are online, but quantum adversaries can only interact with the transcripts of previously completed sessions. Thus we consider only classical adversaries against the standard ACCE game.

In the post-quantum ACCE experiment quantum adversaries can no longer interact with the handshake protocol, but is instead given completed handshake transcripts via the OExecute query. The quantum adversary is still allowed to issue OCorrupt and OReveal queries, but is limited to interacting with the sessions via the OEncrypt and ODecrypt queries. Thus we begin with defining the standard ACCE security experiment.

ACCE security experiment. The security experiment is played between an adversary $\mathcal{A}$ and a challenger who implements all parties according to the ACCE execution environment. After the challenger initializes long-term keys, the adversary receives the long-term public keys of all parties, then interacts with the challenger. In the standard ACCE game, the PPT adversary interacts with the sessions by using OSend, OReveal, OCorrupt, OEncrypt, and ODecrypt queries. In the post-quantum ACCE game, the QPT adversary interacts with the sessions by using OExecute, OReveal, OCorrupt, OEncrypt and ODecrypt queries. Finally, the adversary outputs a triple (i, s, b') and terminates. We begin by defining when sessions match.

Definition 2.3 (Matching sessions)**.** *We say that session π_j^t* matches *π_i^s if $\pi_i^s.\rho \neq \pi_j^t.\rho$; $\pi_i^s.c = \pi_j^t.c$; and $\pi_i^s.\mathsf{sid}$ prefix-matches $\pi_j^t.\mathsf{sid}$, meaning that (i) if π_i^s sent the last message in $\pi_i^s.\mathsf{sid}$, then $\pi_j^t.\mathsf{sid}$ is a prefix of $\pi_i^s.\mathsf{sid}$, or (ii) if π_j^t sent the last message in $\pi_i^s.\mathsf{sid}$, then $\pi_i^s.\mathsf{sid} = \pi_j^t.\mathsf{sid}$.*

Note that for previous analyses of SSH, session IDs consisted of a single value (the hashed session ID) and thus not only prefix-matched, but were identical: $\pi_i^s.\mathsf{sid} = \pi_j^t.\mathsf{sid}$. This is due to the abbreviated handshake option of SSH where clients can send `KEXINIT` and `KEX_INIT_DH` messages simultaneously. We do not capture this abbreviated handshake option in our analysis of SSH, and clients cannot send both `KEXINIT` and `KEX_KEM_INIT` simultaneously.

Next we describe server-only authentication definitions, based on the existence of matching sessions. For server-only authentication, we are only concerned about clients accepting without a matching server session, since the client does not authenticate to the server and so it is trivial for an adversary to impersonate an unauthenticated client. We note that our analysis would follow the ACCE analysis of SSH [6] with minor modifications to prove mutual authentication modes.

Definition 2.4 (Authentication)**.** *Let π_i^s be a session. We say that π_i^s* accepts maliciously *if*

- $\pi_i^s.\mathsf{status} = \mathtt{accept}$*; and*
- $\pi_i^s.\mathsf{pid} = j \neq \perp$*, where no* $\mathsf{OCorrupt}(j)$ *query was issued before π_i^s accepted,*

but there is no unique session π_j^t which matches π_i^s.

Define $\mathsf{Adv}^{\textit{acce-so-auth}}_{\mathsf{ACCE}}(\mathcal{A})$ *as the probability that, when a PPT $\mathcal{A}$ terminates in the ACCE experiment for* ACCE*, there exists an initiator session (i.e., with $\pi_i^s.\rho = \mathtt{init}$) that has accepted maliciously.*

Channel security is defined by the ability to break the confidentiality or integrity of the channel. Formally, this is defined as the ability of the adversary to guess the bit b used in the OEncrypt and ODecrypt queries of an uncompromised session. "Uncompromised" means that the adversary did not reveal the session key at either the session or any matching session, and that that adversary did not corrupt the long-term keys of either party in the session before they transitioned to the channel phase.

Definition 2.5 (Channel security)**.** *Suppose $\mathcal{A}$ with access to* OSend*,* OReveal*,* OCorrupt*,* OEncrypt*,* ODecrypt *outputs (i, s, b') in the ACCE experiment. We say that $\mathcal{A}$* answers the encryption challenge correctly *if*

- $\pi_i^s.\mathsf{status} = \mathtt{accept}$*;*
- *no* $\mathsf{OCorrupt}(j)$ *query was ever issued before $\pi_i^s.\mathsf{status} \leftarrow \mathtt{accept}$ such that $\pi_i^s.\mathsf{pid} = j$ and π_i^s does not have a matching session;*
- *no* $\mathsf{OReveal}(i, s)$ *query was issued;*
- *no* $\mathsf{OReveal}(j, t)$ *query was issued for any π_j^t that matches π_i^s; and*
- $\pi_i^s.b = b'$*.*

Define $\mathsf{Adv}^{\textit{acce-so-aenc}}_{\mathsf{ACCE}}(\mathcal{A})$ *as $|p - 1/2|$, where p is the probability that a PPT $\mathcal{A}$ answers the encryption challenge correctly and either $\pi_i^s.\rho = \mathtt{init}$ or both $\pi_i^s.\rho = \mathtt{resp}$ and there exists a session that matches π_i^s.*

Definition 2.6 (ACCE security)**.** *An ACCE protocol* ACCE *is ϵ-*ACCE*-secure against a PPT adversary $\mathcal{A}$ if, we have that $\mathsf{Adv}^{\textit{acce-so-auth}}_{\mathsf{ACCE}}(\mathcal{A}) \leq \epsilon$ and $\mathsf{Adv}^{\textit{acce-so-aenc}}_{\mathsf{ACCE}}(\mathcal{A}) \leq \epsilon$.*

We now turn to defining post-quantum ACCE security in the Harvest-Now-Decrypt-Later setting. The definition modifies the above security experiment by allowing the adversary to be a quantum polynomial time (QPT) adversary, replacing the OSend query with the OExecute query, and only considering channel security.

Definition 2.7 (Post-quantum channel security)**.** *Suppose $\mathcal{Q}$ with access to* OExecute*,* OReveal*,* OCorrupt*,* OEncrypt*,* ODecrypt *outputs (i, s, b') in the ACCE experiment. We say that $\mathcal{Q}$* answers the encryption challenge correctly *if*

- $\pi_i^s.\mathsf{status} = \mathtt{accept}$*;*
- *no* $\mathsf{OCorrupt}(j)$ *query was ever issued before $\pi_i^s.\mathsf{status} \leftarrow \mathtt{accept}$ such that $\pi_i^s.\mathsf{pid} = j$ and π_i^s does not have a matching session;*
- *no* $\mathsf{OReveal}(i, s)$ *query was issued;*
- *no* $\mathsf{OReveal}(j, t)$ *query was issued for any π_j^t that matches π_i^s; and*
- $\pi_i^s.b = b'$*.*

Define $\mathsf{Adv}^{\textit{pq-acce-aenc}}_{\mathsf{ACCE}}(\mathcal{Q})$ *as $|p - 1/2|$, where p is the probability that a QPT $\mathcal{Q}$ answers the encryption challenge correctly and either $\pi_i^s.\rho = \mathtt{init}$ or both $\pi_i^s.\rho = \mathtt{resp}$ and there exists a session that matches π_i^s.*

Definition 2.8 (Post-quantum ACCE security)**.** *An ACCE protocol* ACCE *is ϵ-*post-quantum-ACCE*-secure against a* QPT *adversary $\mathcal{Q}$ if, we have that $\mathsf{Adv}^{\textit{pq-acce-aenc}}_{\mathsf{ACCE}}(\mathcal{Q}) \leq \epsilon$.*

Note that unlike the original ACCE analysis of the SSH protocol [6] our analysis captures forward secrecy: in both experiments the adversary is allowed to corrupt the long-term secrets of either session after the session has accepted. In the post-quantum ACCE security game, while the handshake is executed without interference from an active adversary, the quantum adversary is allowed to corrupt the long-term keys of either party at any point, capturing forward secrecy.

2.5. Quantum Random Oracle Model (QROM)

We refer the reader to [18] for the basics of quantum computation and information. The QROM is an idealized model where hash functions are modeled as random oracles that are publicly and *quantumly* accessible. Namely, an adversary $\mathcal{A}$ is allowed to query a random oracle $\mathsf{O}\colon \{0,1\}^m \to \{0,1\}^n$ on an arbitrary quantum superposition of inputs, where we use the mapping $|x\rangle|y\rangle \to |x\rangle|y \oplus \mathsf{O}(x)\rangle$ with input register $x \in \{0,1\}^m$ and output register $y \in \{0,1\}^n$. We also use the notation $\mathcal{A}^{|\mathsf{O}\rangle}$ to denote that $\mathcal{A}$ has quantum access to oracle O. We sometimes omit the "ket" notation in $\mathcal{A}^{|\mathsf{O}\rangle}$ and

instead just write $\mathcal{A}^{\mathsf{O}}$ when it is clear from context. For a more detailed description of the QROM, we refer to [11].

Now the following lemma, which is a generalization of the so-called *One-Way To Hiding (OW2H) lemma* in the QROM literature [19], allows us to "program" quantum random oracles.

Lemma 2.1 (Generalized OW2H, simplified [20, Theorem 3])**.** *Let $S \subseteq \mathcal{X}$ be a random subset. Let $\mathsf{G}, \mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be random oracles satisfying $\mathsf{G}(x) = \mathsf{H}(x)$ for every $x \notin S$. Let z be a random bit string. ($S, \mathsf{G}, \mathsf{H}, z$) may have arbitrary joint distribution. Let $\mathcal{A}$ be a quantum oracle algorithm making at most q quantum queries to its corresponding oracle (either G or H)*[4]*. Let $\mathcal{B}^{|\mathsf{H}\rangle}$ be an oracle algorithm that on input z does the following: picks $i \leftarrow\$ [q]$, runs $\mathcal{A}^{|\mathsf{H}\rangle}(z)$ until (just before) the i-th query, measures a query input register in the computational basis, and outputs the measurement outcome t (if $\mathcal{A}$ makes less than i queries, the measurement outcomes are taken to be $\perp \notin X$). Let*

$$P_{\text{left}} = \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}\rangle}(z)],$$
$$P_{\text{right}} = \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{G}\rangle}(z)],$$
$$P_{\text{guess}} = \Pr[t \in S : t \leftarrow \mathcal{B}^{|\mathsf{H}\rangle}(z)].$$

Then, $|P_{\text{left}} - P_{\text{right}}| \leq 2q\sqrt{P_{\text{guess}}}$. The same result also holds with $\mathcal{B}^{|\mathsf{G}\rangle}$ instead of $\mathcal{B}^{|\mathsf{H}\rangle}$ in the definition of P_{guess}.

A consequence of the above lemma is that a quantum random oracle "behaves" as a secure *pseudorandom function (PRF)* even with respect to adversaries that have quantum access to said random oracle. More formally, we have the following (the proof can be found in [21]).

Lemma 2.2 (QRO is a secure PRF [21, Corollary 1])**.** *Let $\mathsf{H}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and $\mathsf{R}\colon \mathcal{X} \to \mathcal{Y}$ be two independent quantum random oracles. Define the (classical) oracles $\mathsf{F}_0 = \mathsf{H}(K, \cdot)$, where we have the "PRF key" $K \leftarrow\$ \mathcal{K}$, and $\mathsf{F}_1 = \mathsf{R}(\cdot)$. Consider an oracle distinguisher $\mathcal{A}^{|\mathsf{H}\rangle, \mathsf{F}_i}$ $(i \in \{0, 1\})$ that makes at most q_{H} quantum queries to H. Then we have $|\Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}\rangle, \mathsf{F}_0}] - \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}\rangle, \mathsf{F}_1}]| \leq \frac{2q_{\mathsf{H}}}{\sqrt{|\mathcal{K}|}}$.*

Finally, the following variant of the OW2H lemma, which is called the *double-sided* OW2H lemma, offers a much tighter bound than Lemma 2.1 on the indistinguishability of the programmed quantum random oracles; however the downside of this lemma is that it comes with limited applicability (e.g., the oracles can only be programmed on a single input). But looking ahead, it turns out that this double-sided OW2H lemma suffices to provide a tight security analysis of Streamlined NTRU Prime as used in OpenSSH.

Lemma 2.3 (Double-sided OW2H [21, Lemma 5])**.** *Let x^* be a random element in set $\mathcal{X}$. Let $\mathsf{G}, \mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be random oracles satisfying $\mathsf{G}(x) = \mathsf{H}(x)$ for every $x \neq x^*$. Let z be a random bit string. ($x^*, \mathsf{G}, \mathsf{H}, z$) may have arbitrary joint distribution. Let $\mathcal{A}$ be a quantum oracle algorithm that has access to either G or H. Then there exists a quantum oracle algorithm $\mathcal{B}^{|\mathsf{G}\rangle, |\mathsf{H}\rangle}(z)$ such that if,*

$$P_{\text{left}} = \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}\rangle}(z)]$$
$$P_{\text{right}} = \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{G}\rangle}(z)]$$
$$P_{\text{extract}} = \Pr[x^* \leftarrow \mathcal{B}^{|\mathsf{G}\rangle, |\mathsf{H}\rangle}(z)],$$

then we have $|P_{\text{left}} - P_{\text{right}}| \leq 2\sqrt{P_{\text{extract}}}$. Here whenever $\mathcal{A}$ queries its quantum oracle, $\mathcal{B}$ queries both G *and* H*, and at the end of $\mathcal{A}$'s execution, $\mathcal{B}$ outputs either x^* or $\perp$. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$,*

3. Post-quantum SSH Protocol

We provide the description of SSH's handshake protocol — the combination of the key-exchange (KEX) protocol in RFC 4253 [22] and the user authentication protocol in RFC 4252 [23] — in Figure 2, where we replace the DH-based KEX with KEM-based KEX. Intuitively speaking, after sending nonces (r_C and r_S) to each other and negotiating cryptographic algorithms, the client and server run a key-exchange protocol and compute session identifier and keys used for authenticated encryption. We note that the server generates a signature on the session identifier. Afterwards, they run the user authentication protocol in a secure channel protected by stateful authenticated encryption.

4. Security Analysis

In this section, we provide a comprehensive analysis of SSH in appropriate post-quantum security models by following a "top-down" approach. Namely, in Subsection 4.1, we first prove security of the SSH protocol in our post-quantum extension of the ACCE model wherein we rely on certain cryptographic properties of the underlying primitives. Then in the subsequent subsections, we prove that the primitives, and their real-world instantiations, satisfy these corresponding properties in the post-quantum setting: Subsection 4.2 covers the hybrid key exchange primitive and Subsection 4.3 covers the $\mathsf{PRF}_{\mathsf{SSH}}$ primitive (see Figure 5).

4.1. Post-quantum ACCE Security of SSH

We begin by proving the security of the post-quantum SSH protocol in server-only mode. The following theorem shows that, if the hash function H_c is collision-resistant, the signature scheme DSS is EUF-CMA-secure, $\mathsf{PRF}_{\mathsf{SSH}}$ is a secure PRF, the symmetric encryption is a secure BSAE scheme[5] and the key encapsulation mechanism KEM is IND-CPA-secure, then the post-quantum SSH protocol is a secure server-only ACCE protocol. Notably, our security proof relies on the *weaker* assumption of IND-CPA secure KEMs when compared to the recent analysis in [4] which requires IND-CCA secure KEMs.

4. The generalized OW2H lemma in [20] technically has "q" to be the *query-depth* of $\mathcal{A}$ which accounts for potential *parallel* oracle queries made by the algorithm. We will not be considering such parallel queries in this paper for the sake of simplicity. But at the same time, our subsequent analyses in the QROM can be extended to this parallel setting in a straightforward manner.

5. All assumptions match [6] except the IND-CPA KEM assumption.

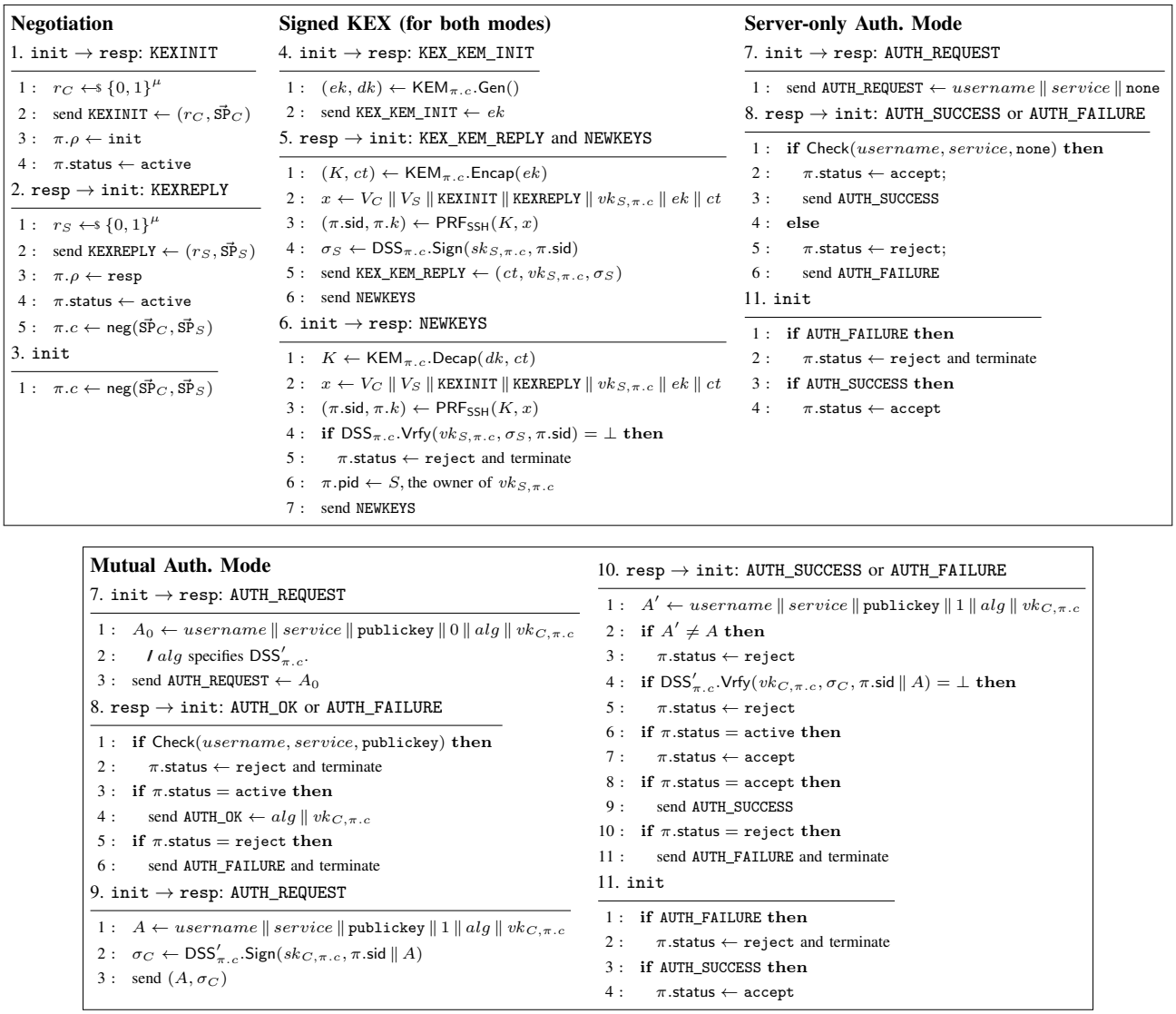

Figure 2. Description of SSH handshake protocol (RFC 4253 + RFC 4252) with KEM-based KEX. In steps 1 and 2, algorithm neg negotiates a ciphersuite to be used in the protocol. V_C and V_S are version strings. In steps 5 and 6, PRF_{SSH} is defined in Figure 5. In step 8, algorithm Check checks the accessibility of username for service via some authentication mechanism, `none` or `publickey`. We note that from steps 7–11, their protocol is run within BSAE using $\pi.k$.

Lemma 4.1 (SSH is ACCE-Authentication-secure)**.** *Let μ be the length of the nonces in* KEXINIT *and* KEXREPLY *($\mu = 128$), n_P the number of participating parties and n_S the maximum number of sessions per party. Thus, for algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_5$ given in the proof, and for all algorithms $\mathcal{A}$,*

$$\begin{aligned}\text{Adv}^{\textit{acce-so-auth}}_{\text{SSH}}(\mathcal{A}) \leq{}& \frac{(n_P n_S)^2}{2^\mu} + \text{Adv}^{\textit{coll-res}}_{\text{H}_c}(\mathcal{B}_1^{\mathcal{A}}) \\ &+ n_P^2 \cdot n_S \big(\text{Adv}^{\textit{euf-cma}}_{\text{DSS}}(\mathcal{B}_2^{\mathcal{A}}) \\ &+ \text{Adv}^{\textit{ind-cpa}}_{\text{KEM}}(\mathcal{B}_3^{\mathcal{A}}) + \text{Adv}^{\textit{prf}}_{\text{PRF}_{\text{SSH}}}(\mathcal{B}_4^{\mathcal{A}}) \\ &+ \text{Adv}^{\textit{bsae}}_{\text{BSAE}}(\mathcal{B}_5^{\mathcal{A}})\big).\end{aligned}$$

Sketch. Due to space limitations we give a proof sketch, and point readers to Appendix A for full details.

Our proof proceeds in a sequence of game hops. We begin by introducing a series of conditions that cause our proof to abort. This occurs if any nonces collide, if any hash collisions are output, if the challenger incorrectly guesses the first (client) session to accepts maliciously, and finally if the adversary manages to forge a valid signature from the client's session partner.

From this point we demonstrate that the adversary cannot modify any messages between the client and its communicating partner. We do so by showing that the adversary cannot forge valid ciphertexts for the BSAE scheme used between the client and the server. We replace keys used by the BSAE scheme (via a *ind-cpa* assumption on the KEM, and a PRF assumption on the PRF_{SSH}) with uniformly random keys. Finally, we replace the encryption and decryption of BSAE

ciphertexts with a *bsae* challenger: any adversary capable of forging messages between the client and server can be turned into an attacker against the *bsae* security of the BSAE scheme. This completes the authentication proof sketch. □

We now turn to proving the classical ACCE-channel-security of our post-quantum SSH protocol.

Lemma 4.2 (Channel security, server-only auth. mode)**.** *Let μ be the length of the nonces in* KEXINIT *and* KEXREPLY *($\mu = 128$), n_P the number of participating parties and n_S the maximum number of sessions per party. The algorithms $\mathcal{B}_3$, $\mathcal{B}_4$, and $\mathcal{B}_5$, explicitly given in the proof of the lemma, are such that, for all algorithms $\mathcal{A}$,*

$$\begin{aligned}\mathsf{Adv}^{\textit{acce-so-aenc}}_{\mathsf{SSH}}(\mathcal{A}) \leq\ & \mathsf{Adv}^{\textit{acce-so-auth}}_{\mathsf{SSH}}(\mathcal{A}) \\ & + n_P^2 n_S^2 \big(\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}}(\mathcal{B}_6^{\mathcal{A}}) \\ & + \mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{B}_7^{\mathcal{A}}) + \mathsf{Adv}^{\textit{bsae}}_{\mathsf{BSAE}}(\mathcal{B}_8^{\mathcal{A}})\big).\end{aligned}$$

Sketch. Due to space limitations we give a proof sketch, and point readers to the Appendix A for full details.

Our proof proceeds in a sequence of game hops. We begin by introducing a series of conditions that cause our proof to abort. This occurs if the adversary manages to cause our test session to accept maliciously (in the sense of Lemma 4.1), or if the challenger incorrectly guesses the test session.

From this point we demonstrate that the adversary cannot break the security of the ACCE channel. We do so by forwarding all OEncrypt/ODecrypt queries sent to the test session and its matching partner to a *bsae* challenger.

However, we must first replace the keys used by the BSAE scheme with uniformly random keys (via a *ind-cpa* assumption on the KEM, and a PRF assumption on the $\mathsf{PRF}_{\mathsf{SSH}}$). Now our reduction simply forwards all OEncrypt/ODecrypt queries to our *bsae* challenger: the output from the adversary is forwarded to the *bsae* challenger, and we win the *bsae* game with the same advantage as our adversary, completing the channel security proof sketch. □

Finally, we turn to proving the post-quantum ACCE security of the PQ-SSH protocol.

Theorem 4.1 (Post-quantum channel security)**.** *Let μ be the length of the nonces in* KEXINIT *and* KEXREPLY *($\mu = 128$), n_P the number of participating parties and n_S the maximum number of sessions per party. The algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$, explicitly given in the proof of the theorem, are such that, for all algorithms $\mathcal{Q}$,*

$$\begin{aligned}\mathsf{Adv}^{\textit{pq-acce-aenc}}_{\mathsf{SSH}}(\mathcal{Q}) \leq\ & n_P n_S \big(\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}}(\mathcal{B}_1^{\mathcal{Q}}) \\ & + \mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{B}_2^{\mathcal{Q}}) \\ & + \mathsf{Adv}^{\textit{bsae}}_{\mathsf{BSAE}}(\mathcal{B}_3^{\mathcal{Q}})\big).\end{aligned}$$

Proof. Let $\mathsf{break}^{(2)}_\delta$ be the event that occurs when $\mathcal{Q}$ answers the encryption challenge correctly in Game δ in the sense of Definition 2.7.

Game 0. This game equals the ACCE channel security experiment described in Section 2.4. Note that since we are capturing Harvest-Now-Decrypt-Later attacks, the adversary uses OExecute to generate sessions, and thus from now on, we always have a matching session for the session π_i^s where the adversary tries to guess the random bit.

Game 1. In this game, we guess the session for which the adversary outputs the bit b'. We guess two indices $(i, s) \in [n_P] \times [n_S]$ and abort if the adversary outputs (i^*, s^*, b') with $(i^*, s^*) \neq (i, s)$. This happens with probability $\frac{1}{n_P n_S}$. By the definition of the channel security experiment we have that there exists a unique partner session π_j^t which can be easily determined by the simulator. Thus we have: $\Pr(\mathsf{break}^{(2)}_0) \leq n_P n_S \cdot \Pr(\mathsf{break}^{(2)}_1)$.

Game 2. In this game we replace the value $K, ct = \mathsf{KEM.Encap}(ek)$, $K = \mathsf{KEM.Decap}(dk, ct)$ computed by π_i^s and π_j^t with a uniformly random value K^*. Specifically, we introduce a reduction $\mathcal{B}_1$ that interacts with $\mathcal{Q}$ and embeds an IND-CPA challenge into the test session's transcript. Note that in what follows, we assume WLOG that the test session π_i^s is the client. The alternative case (where π_i^s is the server) follows identically up to a change in notation.

At the beginning of the experiment, $\mathcal{B}_1$ initialises an IND-CPA KEM challenger. When π_i^s needs to compute a KEM public key pair, $\mathcal{B}_1$ replaces the honest computation of (ek, dk) with (ek^*, dk^*) output by the IND-CPA challenger. Similarly, when the server session π_j^t needs to encapsulate a shared secret, $\mathcal{B}_1$ replaces K, ct with the key output K^* and the ciphertext output ct^* from the IND-CPA challenger. Finally, when π_i^s needs to compute K, $\mathcal{B}_1$ replaces K with K^*. We note that by definition π_i^s and π_j^t communicate without any modification from the adversary.

Any adversary $\mathcal{Q}$ that can distinguish this game from the previous game can directly be used to construct an adversary $\mathcal{B}_1^{\mathcal{Q}}$ that can break the IND-CPA assumption: If the challenge bit b sampled by the IND-CPA KEM challenger is 0, then $K^* = \mathsf{Decap}(dk, ct)$ and we are in Game 1. If the challenge bit b sampled by the IND-CPA KEM challenger is 1, then K^* is instead uniformly random and independent of the protocol flow and we are in Game 2. Thus $\Pr(\mathsf{break}^{(2)}_1) \leq \Pr(\mathsf{break}^{(2)}_2) + \mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}}(\mathcal{B}_1^{\mathcal{Q}})$.

Game 3. In this game we replace the values $H, k_1, ..., k_6$ computed by π_i^s and π_j^t as $\mathsf{PRF}_{\mathsf{SSH}}(K^*, sid)$ with random values $H^*, k_1^*, ..., k_6^*$. Any adversary $\mathcal{Q}$ that can distinguish this game from the previous game can directly be used to construct an adversary $\mathcal{B}_2^{\mathcal{Q}}$ that can break the PRF assumption: let $S = H\|k_1\|...\|k_6$ be the output of $\mathsf{PRF}_{\mathsf{SSH}}$, and let $S^* = H^*\|k_1^*\|...\|k_6^*$ be a random string of the same length. For S we are in Game 2, and for S^* in Game 3. Thus $\Pr(\mathsf{break}^{(2)}_2) \leq \Pr(\mathsf{break}^{(2)}_3) + \mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{B}_2^{\mathcal{A}})$.

Final analysis. We now have that the keys $k_1^*, ..., k_6^*$ are uniformly random and independent from the protocol transcript. Thus any adversary $\mathcal{Q}$ that can guess (i^*, s^*, b') correctly can directly be used to construct an adversary $\mathcal{B}_3^{\mathcal{Q}}$ that breaks the BSAE scheme. Technically we exploit the fact that all keys for the encryption scheme are now independent from the handshake and embed a BSAE challenger to answer $\mathcal{Q}$'s queries for Enc, Dec. Now $\mathcal{B}_3$ simply forwards $\mathcal{Q}$'s output to the challenger and thus we have $\Pr(\mathsf{break}^{(2)}_3) \leq \mathsf{Adv}^{\textit{bsae}}_{\mathsf{BSAE}}(\mathcal{B}_3^{\mathcal{A}})$. □

4.2. Hybrid KEMs in SSH

Post-quantum implementations of SSH, e.g., OpenSSH [1], TinySSH [7], and OQS-OpenSSH [24], use hybrid key exchange (or more technically, KEMs) in the handshake phase which combine a post-quantum secure KEM with a classically secure counterpart by simply concatenating their respective shared keys and hashing the result to derive the final key. More formally, a hybrid KEM $\mathsf{KEM}_{\mathsf{hy}} = (\mathsf{Gen}_{\mathsf{hy}}, \mathsf{Encap}_{\mathsf{hy}}, \mathsf{Decap}_{\mathsf{hy}})$ used in SSH combines two constituent KEMs, i.e., a post-quantum secure $\mathsf{KEM}_0 = (\mathsf{Gen}_0, \mathsf{Encap}_0, \mathsf{Decap}_0)$ and a classically secure $\mathsf{KEM}_1 = (\mathsf{Gen}_1, \mathsf{Encap}_1, \mathsf{Decap}_1)$, with the hash function H_c[6] as shown in Figure 3.

As indicated in our above ACCE security analysis of the post-quantum SSH protocol, we would need an IND-CPA secure hybrid KEM. In this section, we will show that the hybrid KEM $\mathsf{KEM}_{\mathsf{hy}}$ used in SSH retains IND-CPA security of any of its constituent KEMs when H_c is modeled as a quantum random oracle. Now before diving into our proof, we briefly discuss how our analysis relates to prior work on hybrid KEM combiners.

The works [25] and [26] initiated a formal study of KEM combiners in a post-quantum setting. However they considered different and (slightly) more complicated hybrid KEMs than $\mathsf{KEM}_{\mathsf{hy}}$ above since their goal was to retain/achieve security stronger than IND-CPA (e.g., IND-CCA[7]); but as we saw in our ACCE security analysis of post-quantum SSH, IND-CPA security of hybrid KEMs is in fact sufficient. On the other hand, Petcher and Campagna [3] studied hybrid KEMs which concatenate shared keys as in the post-quantum versions of SSH (and TLS 1.3); however they considered a more "advanced" KEM abstraction wherein the final key is not only derived from the concatenated keys but also involves public keys and ciphertexts of the constituent KEMs along with some auxiliary context and label information. They were also able to prove IND-CCA security of such a hybrid KEM albeit in the *classical* random oracle model. But the authors did not discuss how their results formally relate to the wider protocol-level security of post-quantum SSH. In contrast, the subsequent analysis of $\mathsf{KEM}_{\mathsf{hy}}$ follows from our ACCE security theorem for post-quantum SSH above; this top-down approach also allows us to focus on a modular and much simpler $\mathsf{KEM}_{\mathsf{hy}}$ abstraction in the handshake phase.

Now our following IND-CPA security analysis of $\mathsf{KEM}_{\mathsf{hy}}$ in the QROM only considers two constituent KEMs for simplicity. However it is straightforward to extend the proof to multiple constituent KEMs. At the same time, one can also easily extend our analysis from the QROM to the standard model by replacing the hash function H_c with a so-called "*dual PRF*"[8].

6. The same hash function H_c is used in $\mathsf{PRF}_{\mathsf{SSH}}$ as described in Figure 5.

7. In contrast to IND-CPA, the notion of IND-CCA security allows adversaries to obtain decapsulations of ciphertexts of their choosing.

8. A function $\mathsf{F}(\cdot,\cdot)$ is said to be a secure *dual PRF* if it is a secure PRF (in the standard sense) when "keyed" by the first input, i.e., $\mathsf{F}(K,\cdot)$, and *also* when "keyed" by the second input, i.e., $\mathsf{F}(\cdot,K)$.

$\mathsf{Encap}_{\mathsf{hy}}(ek)$	$\mathsf{Decap}_{\mathsf{hy}}(dk, ct)$
1 : Parse $(ek_0, ek_1) \leftarrow ek$	1 : Parse $(dk_0, dk_1) \leftarrow dk$
2 : $(ct_0, K_0) \leftarrow \mathsf{Encap}_0(ek_0)$	2 : Parse $(ct_0, ct_1) \leftarrow ct$
3 : $(ct_1, K_1) \leftarrow \mathsf{Encap}_1(ek_1)$	3 : $K_0 \leftarrow \mathsf{Decap}_0(dk_0, ct_0)$
4 : $ct \coloneqq (ct_0, ct_1)$	4 : $K_1 \leftarrow \mathsf{Decap}_1(dk_1, ct_1)$
5 : $K \coloneqq \mathsf{H}_c(K_0 \,\|\, K_1)$	5 : **if** $K_0 = \perp \vee K_1 = \perp$
6 : **return** (ct, K)	6 : **return** $\perp$
	7 : $K \coloneqq \mathsf{H}_c(K_0 \,\|\, K_1)$
	8 : **return** K

Figure 3. Abstract KEM combiner used in post-quantum SSH. $\mathsf{Gen}_{\mathsf{hy}}(1^\lambda)$ simply generates $(ek_0, dk_0) \leftarrow \mathsf{Gen}_0(1^\lambda)$, $(ek_1, dk_1) \leftarrow \mathsf{Gen}_1(1^\lambda)$, and outputs the encapsulation key $ek \coloneqq (ek_0, ek_1)$ and decapsulation key $dk \coloneqq (dk_0, dk_1)$.

Theorem 4.2. *For any IND-CPA adversary $\mathcal{A}$ against $\mathsf{KEM}_{\mathsf{hy}}$ (described in Fig. 3) making q_{H_c} quantum queries to the underlying random oracle H_c, there exists an IND-CPA adversary $\mathcal{B}$ against the constituent KEM KEM_i — for any $i \in \{0,1\}$ — such that*

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}_{\mathsf{hy}}}(\mathcal{A}) \leq \mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}_i}(\mathcal{B}) + \frac{2q_{\mathsf{H}_c}}{\sqrt{|\mathcal{K}_i|}},$$

where $\mathcal{K}_i$ is the encapsulated key space of KEM_i, and the running time of $\mathcal{B}$ is about the same as that of $\mathcal{A}$.

Proof. We first consider the case of $i = 0$. The proof proceeds with a sequence of game hops ($\mathbf{G}_0 \rightarrow \mathbf{G}_1 \rightarrow \mathbf{G}_2$) with respect to adversary $\mathcal{A}$. The games $\mathbf{G}_0$ and $\mathbf{G}_2$ are essentially the IND-CPA security games for $\mathsf{KEM}_{\mathsf{hy}}$ where $\mathcal{A}$ gets the "real" encapsulated key $K^* \coloneqq \mathsf{H}_c(K_0^* \,\|\, K_1^*)$ as in Figure 3 and a uniformly "random" encapsulated key $K^* \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{hy}}$ respectively. From the advantage definition of IND-CPA adversaries, it is not hard to see that

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}_{\mathsf{hy}}}(\mathcal{A}) = |\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_2]|.$$

In the intermediate game $\mathbf{G}_1$, we use a uniformly random KEM_0 encapsulated key $K_0^* \leftarrow\!\!\$\ \mathcal{K}_0$ to derive the final key $K^* \coloneqq \mathsf{H}_c(K_0^* \,\|\, K_1^*)$ in contrast to Figure 3. Using a straightforward reduction to the IND-CPA security of KEM_0, it is not hard to obtain

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}_0}(\mathcal{B}) = |\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_1]|,$$

where $\mathcal{B}$ is an IND-CPA adversary against KEM_0 which runs in about the same time as $\mathcal{A}$; $\mathcal{B}$ can simulate games $\mathbf{G}_0$ and $\mathbf{G}_1$ towards $\mathcal{A}$ by simulating the QRO $|\mathsf{H}_c\rangle$ (e.g., using a $2q_{\mathsf{H}_c}$-wise independent function as in [27]) by itself, in addition to deriving the KEM_1-based values by generating $(ek_1, dk_1) \leftarrow \mathsf{Gen}_1(1^\lambda)$.

Now interpreting $K_0^* \leftarrow\!\!\$\ \mathcal{K}_0$ as a secret "PRF key" in $K^* \coloneqq \mathsf{H}_c(K_0^* \,\|\, K_1^*)$, we use the fact that the QRO $|\mathsf{H}_c\rangle$ is a secure PRF (Lemma 2.2) to replace K^* with a uniformly random key in $\mathbf{G}_2$, thereby obtaining

$$|\Pr[1 \leftarrow \mathbf{G}_1] - \Pr[1 \leftarrow \mathbf{G}_2]| \leq \frac{2q_{\mathsf{H}_c}}{\sqrt{|\mathcal{K}_0|}}.$$

Applying a triangle inequality on the above bounds finishes the proof — at least for the case $i = 0$.

The proof for $i = 1$ follows analogously wherein we rely on IND-CPA security of KEM_1 to first replace the "real" encapsulated key K_1^* derived from Encap_1 with a uniformly random $K_1^* \leftarrow\!\!\$\ \mathcal{K}_1$. Then to replace the final $\mathsf{KEM}_{\mathsf{hy}}$ key $K^* := \mathsf{H}_c(K_0^* \,\|\, K_1^*)$ with a uniformly random one, we observe that the proof of Lemma 2.2 in the literature — specifically, the proof of [21, Corollary 1] — can be extended in a straightforward manner to show $|\mathsf{H}_c\rangle$, and any QRO in general, is also a secure *dual PRF*. In fact, in Subsection 4.3 below, we extend the proof of [21, Corollary 1] to prove a stronger result — namely that $\mathsf{H}_c(K_1^* \,\|\, \cdot)$ and $\mathsf{H}_c(\cdot \,\|\, K_1^*)$ are *jointly* secure PRFs. □

4.2.1. Concrete Instantiations. When it comes to instantiating the above hybrid $\mathsf{KEM}_{\mathsf{hy}}$ for post-quantum versions of SSH, there have been a couple of proposals for standardization. Friedl, Mojzis, and Josefsson proposed the hybrid KEM `sntrup761x25519-sha512` in [28] which basically instantiates the post-quantum secure KEM_0 with the NIST submission *Streamlined NTRU Prime* [8] and the classically secure KEM_1 with the well-known *X25519* elliptic curve Diffie-Hellman key exchange (ECDH) [29]; also the hash function H_c is instantiated with SHA-512. Following the FIPS publication of the NIST standard *ML-KEM* [9], Kampanakis, Stebila, and Hansen proposed the hybrid KEMs `mlkem768nistp256-sha256`, `mlkem1024nistp384-sha384`, `mlkem768x25519-sha256` in [30] which use different parameter sets for ML-KEM in KEM_0 and different ECDH methods for KEM_1 [9], along with different instantiations of H_c.

Focusing on real-world implementations of SSH, OpenSSH v.8.0 (Apr. 2019) and TinySSH (Jan. 2019) incorporated `sntrup4591761x25519-sha512` which used an older version of Streamlined NTRU Prime for KEM_0. Later OpenSSH v.8.5 (Mar. 2021) and TinySSH (Mar. 2021) replaced the hybrid KEM with `sntrup761x25519-sha512` described above; OpenSSH v.9.0 (Apr. 2022) also set it as the *default* key exchange method. Recently, OpenSSH v.9.9 (Sep. 2024) added support for the hybrid KEM `mlkem768x25519-sha256`.[10] Since Streamlined NTRU Prime is quite prevalent in post-quantum implementations of SSH, and arguably has not received much scrutiny compared to ML-KEM, we will be analyzing this constituent KEM in what follows.

But let us first quickly consider the *classical* security of the above hybrid KEMs. Now X25519, when seen as a KEM, offers IND-CPA security based on the Decisional Diffie-Hellman (DDH) assumption in the nominal group. From Theorem 4.2, it then follows that the related hybrid KEMs in SSH retain IND-CPA security albeit against *classical* attackers. For post-quantum ACCE security of SSH, we would also need to show that Streamlined NTRU Prime does achieve IND-CPA security against *quantum* attackers.

Zooming in on Streamlined NTRU Prime, the KEM is constructed by starting with a deterministic OW-CPA secure (i.e., "one-way") PKE scheme called *Streamlined NTRU Prime Core* [8] and applying a variant of the so-called *Fujisaki-Okamoto (FO) transform* ([32], [33], [34]); the exact variant of the FO transform used in Streamlined NTRU Prime is described in Figure 4. Now the original FO transforms have been analyzed in the QROM (e.g., in [21], [34], [35]) to show that if we start with such a deterministic OW-CPA secure PKE scheme, then the resulting KEM achieves the stronger notion IND-CCA security. However as pointed out in [10], the specific transform used in Streamlined NTRU Prime deviates quite significantly from the original FO transforms which in turn makes the above QROM IND-CCA security results in the literature inapplicable. Fortunately, in the context of our ACCE security analysis of post-quantum SSH, we only need to prove the weaker IND-CPA security of Streamlined NTRU Prime — which we do so in the QROM as stated below. At the same time, we also sketch an IND-CCA security proof for Streamlined NTRU Prime in Appendix B by adapting some advanced QROM proof techniques, thereby addressing the related open problem in [10]. Such an IND-CCA security proof accounts for potential "*key-reuse*" attacks in misimplementations of post-quantum SSH wherein the hybrid KEM is used with a static public key, instead of an ephemeral one, in the handshake phase.

$\overline{\mathsf{Gen}}(1^\lambda)$

1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\lambda)$
2 : $s \leftarrow\!\!\$\ \{0,1\}^\ell$
3 : $h := \mathsf{H}_4(ek)$
4 : $\overline{dk} := (dk, ek, s, h)$
5 : **return** $(ek, \overline{dk})$

$\mathsf{Encap}(ek)$

1 : $m \leftarrow\!\!\$\ \mathcal{M}$
2 : $ct_0 := \mathsf{Enc}(ek, m)$
3 : $ct_1 := \mathsf{H}_2(\mathsf{H}_3(m), \mathsf{H}_4(ek))$
4 : $ct := (ct_0, ct_1)$
5 : $K := \mathsf{H}_1(\mathsf{H}_3(m), ct)$
6 : **return** (ct, K)

$\mathsf{Decap}(\overline{dk}, ct)$

1 : Parse $(dk, ek, s, h) \leftarrow \overline{dk}, (ct_0, ct_1) \leftarrow ct$
2 : $m' := \mathsf{Dec}(dk, ct_0)$
3 : $ct_0' := \mathsf{Enc}(ek, m')$
4 : $ct_1' := \mathsf{H}_2(\mathsf{H}_3(m'), h)$
5 : **if** $(ct_0, ct_1) \neq (ct_0', ct_1')$ **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), ct)$
6 : **else return** $K := \mathsf{H}_1(\mathsf{H}_3(m'), ct)$

Figure 4. The FO-type transform used in Streamlined NTRU Prime. Here $\mathsf{sntrup\text{-}core} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is the deterministic OW-CPA secure PKE scheme called *Streamlined NTRU Prime Core* [8] with message space $\mathcal{M}$; $\mathsf{sntrup} = (\overline{\mathsf{Gen}}, \mathsf{Encap}, \mathsf{Decap})$ is the Streamlined NTRU Prime KEM. Also for all $i \in \{0, 1, \ldots, 4\}$, we have $\mathsf{H}_i \colon \{0,1\}^* \to \{0,1\}^{256}$ to be hash functions. Streamlined NTRU Prime instantiates these different hash functions with appropriately domain separated SHA-512 where the outputs are truncated to the first 32 bytes.

Theorem 4.3. *For any IND-CPA adversary $\mathcal{A}$ against the* sntrup *KEM (described in Figure 4) making q_{H_i} queries to the underlying quantum random oracles H_i for $i \in \{0, 1, \ldots, 4\}$, there exist OW-CPA adversaries $\mathcal{B}$ and $\mathcal{B}'$*

9. Here `nistp256` and `nistp384` refer to the NIST-recommended elliptic curves *P-256* and *P-384* respectively in [31].

10. OQS-OpenSSH [24] also supports hybrid KEMs which instantiate KEM_0 with quite a few NIST submissions; however these implementations are for experimental purposes, and as such are not deployed in practice.

against the sntrup-core *PKE scheme such that*

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{sntrup}}(\mathcal{A}) \leq 2\Big(\sqrt{\mathsf{Adv}^{\textit{ow-cpa}}_{\mathsf{sntrup\text{-}core}}(\mathcal{B})} + \sqrt{\mathsf{Adv}^{\textit{ow-cpa}}_{\mathsf{sntrup\text{-}core}}(\mathcal{B}')}\Big) + \frac{2q_{\mathsf{H}_1} + 4q_{\mathsf{H}_2}}{2^{128}},$$

where the running times of $\mathcal{B}$ and $\mathcal{B}'$ are about the same as that of $\mathcal{A}$.

We provide the full proof of this theorem in Appendix A because of space limitations. At a high level, we leverage the fact that the underlying Streamlined NTRU Prime Core PKE scheme is *deterministic* in order to use the "tighter" *double-sided* OW2H lemma (Lemma 2.3), instead of the generalized OW2H lemma (Lemma 2.1); this in turn results in tight bounds on the IND-CPA security of Streamlined NTRU Prime KEM in the QROM. One can also extend our above IND-CPA security results to ML-KEM in a straightforward manner since the NIST standard uses a much simpler FO-type transform when compared to Streamlined NTRU Prime. However the resulting bounds would be relatively non-tight[11] since we cannot use the double-sided OW2H lemma in this case as the PKE scheme underlying ML-KEM is *randomized*.

4.3. SSH PRF in the QROM

$\mathsf{PRF}_{\mathsf{SSH}}(K, x)$

1 : $H \leftarrow \mathsf{H}_c(x \,\|\, K)$
2 : $\mathsf{label} \leftarrow [A, B, C, D, E, F]$
3 : **for** $i \in \{1, 2, \ldots, 6\}$
4 : $\quad k_i \leftarrow \mathsf{H}_c(K \,\|\, H \,\|\, \mathsf{label}[i] \,\|\, H)$
5 : **return** $(H, k_1, \ldots, k_6)$

Figure 5. Description of $\mathsf{PRF}_{\mathsf{SSH}}$ used in the SSH protocol to compute session ID H and session keys $(k_i)_{i\in[6]}$ using the hash function H_c.

Note that in our above analysis of post-quantum SSH in the ACCE security model, we also needed to assume that the $\mathsf{PRF}_{\mathsf{SSH}}$ function used in the protocol is a secure PRF. Prior works on analysing SSH (namely, [4], [6]) go into detail on why we need to make such a "monolithic" PRF assumption on $\mathsf{PRF}_{\mathsf{SSH}}$, and why we cannot rely on a weaker assumption that the underlying hash function H_c is collision-resistant. At a high level, this is because the hash H_c is applied on an input which includes the shared key K (see Figures 2 and 5), and collision resistance of H_c does not guarantee the secrecy of K.

The authors of [6] also mention that if H_c is modeled as a (classical) random oracle, then $\mathsf{PRF}_{\mathsf{SSH}}$ can be proven to be a secure PRF. At the same time, they leave proving the same in the standard model as an open question. Now in the post-quantum setting, since the QROM uses a weaker heuristic than the ROM (but a stronger heuristic than the standard model) in terms of providing attackers with a more realistic quantum access to public hash functions, we take a step towards answering this question by concretely establishing the security of $\mathsf{PRF}_{\mathsf{SSH}}$ when H_c is modeled as a quantum random oracle.[12] But before diving into our QROM security analysis of $\mathsf{PRF}_{\mathsf{SSH}}$, we first give some intuition behind it.

It is folklore in the QROM literature that a keyed quantum random oracle is a secure PRF (e.g., see [35, Lemma 2.2], [21, Corollary 1]). This means that the functions $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$ in the description of $\mathsf{PRF}_{\mathsf{SSH}}$ (see Figure 5) are secure PRFs. So towards proving the security of $\mathsf{PRF}_{\mathsf{SSH}}$ in the QROM, one can first replace $\mathsf{H}_c(\cdot \,\|\, K)$ with a uniformly random function R_1 (with the same domain and co-domain as that of $\mathsf{H}_c(\cdot \,\|\, K)$), and then replace $\mathsf{H}_c(K \,\|\, \cdot)$ with another independent random function R_2. But there is a problem with this approach. Note that the functions $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$ are highly correlated in the sense that they share the same key K. So we cannot just replace these functions with their uniformly random counterparts *individually*. For example, if we first replace $\mathsf{H}_c(\cdot \,\|\, K)$ with R_1, then the reduction needs to simulate the function $\mathsf{H}_c(K \,\|\, \cdot)$ towards the $\mathsf{PRF}_{\mathsf{SSH}}$-attacker; however the reduction cannot do so because it does not know the secret key K. To overcome this problem, we would need some kind of a *joint* PRF assumption on $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$. However the authors of [4] mention that this is an "unusual" assumption, and hence resort to the monolithic PRF assumption on $\mathsf{PRF}_{\mathsf{SSH}}$ (similar to [6]) in their analysis of post-quantum SSH.

In our following analysis, we justify this unusual assumption in the QROM by revisiting the proof of the folklore "keyed QRO is a PRF" result (specifically, that of [21, Corollary 1]) and extending it to our above joint PRF setting. This allows us to replace the functions $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$ with R_1 and R_2 *simultaneously*, which later makes it easy to formally argue security of $\mathsf{PRF}_{\mathsf{SSH}}$.

Theorem 4.4. *Let ℓ_K and ℓ_H denote the bit-lengths of shared hybrid key K and session ID H respectively in the post-quantum SSH protocol. Then for any adversary $\mathcal{A}$ against $\mathsf{PRF}_{\mathsf{SSH}}$ (see Fig. 5) making q_{PRF} classical PRF queries and q_{H_c} quantum queries to the underlying random oracle H_c, we have*

$$\mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{A}) \leq q_{\mathsf{H}_c} \cdot 2^{-\left(\frac{\ell_K - 3}{2}\right)} + q^2_{\mathsf{PRF}} \cdot 2^{-\ell_H}.$$

Proof. Let ℓ_{tr} denote the bit-length of transcripts on which $\mathsf{PRF}_{\mathsf{SSH}}$ is applied in the post-quantum SSH protocol (see Figure 2). Note that we have

$$\mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{A}) = |\Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle, \mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)}] - \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle, \mathsf{R}}]|,$$

where we have the secret key $K \leftarrow\!\$\, \{0,1\}^{\ell_K}$, and R to be an independent and uniformly random function with domain $\{0,1\}^{\ell_{\mathsf{tr}}}$ and co-domain $\{0,1\}^{7\ell_H}$ $\big($i.e., the same as that of $\mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)\big)$. To establish the above upper bound on $\mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{A})$, we first provide $\mathcal{A}$ classical access to an

11. More specifically, we would obtain a multiplicative factor based on the number of quantum random oracle queries made by the IND-CPA adversary.

12. In fact, our security proof of $\mathsf{PRF}_{\mathsf{SSH}}$ also holds in the so-called *Non-Observable QROM (NO QROM)* [36] which uses even weaker heuristics than the plain QROM wherein we do not observe an attacker's queries to the hash oracle (but we still rely on "programming" such an oracle).

"intermediate" function F described as follows: F is basically the same as $\mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)$ but where instead of computing $H \leftarrow \mathsf{H}_c(x \,\|\, K)$ and $k_i \leftarrow \mathsf{H}_c(K \,\|\, H \,\|\, \mathsf{label}[i] \,\|\, H)$ (for $i \in \{1,\ldots,6\}$) as in Figure 5, we compute $H \leftarrow \mathsf{R}_1(x)$ and $k_i \leftarrow \mathsf{R}_2(H \,\|\, \mathsf{label}[i] \,\|\, H)$ for two independent and uniformly random functions $\mathsf{R}_1 : \{0,1\}^{\ell_{\mathsf{tr}}} \rightarrow \{0,1\}^{\ell_H}$ and $\mathsf{R}_2 : \{0,1\}^{2\ell_H+\ell_{\mathsf{label}}} \rightarrow \{0,1\}^{\ell_H}$, where ℓ_{label} denotes the bit-length of labels in the $\mathsf{PRF}_{\mathsf{SSH}}$ computations.

Now we prove the following lemma.

Lemma 4.3.

$$|\Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)}] - \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{F}}]| \leq q_{\mathsf{H}_c} \cdot 2^{-\left(\frac{\ell_K-3}{2}\right)}.$$

Proof. Let $\mathcal{B}$ be an algorithm that has quantum access to random oracle H_c, and classical access either to the oracles $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$ or oracles R_1 and R_2 used in function F.[13] It is not hard to see that $\mathcal{B}^{|\mathsf{H}_c\rangle,\mathsf{H}_c(\cdot \,\|\, K),\mathsf{H}_c(K \,\|\, \cdot)}$ (resp. $\mathcal{B}^{|\mathsf{H}_c\rangle,\mathsf{R}_1,\mathsf{R}_2}$) can perfectly simulate the PRF security experiment $\mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)}$ (resp. $\mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{F}}$); $\mathcal{B}$ can respond to $\mathcal{A}$'s classical PRF queries by performing the corresponding computations (either that of $\mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)$ or F) locally using its classical oracles. By having $\mathcal{B}$ output the same bit as that of $\mathcal{A}$ at the end of the experiment, we therefore have that $\Pr[1 \leftarrow \mathcal{B}^{|\mathsf{H}_c\rangle,\mathsf{H}_c(\cdot \,\|\, K),\mathsf{H}_c(K \,\|\, \cdot)}] = \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{PRF}_{\mathsf{SSH}}(K,\cdot)}]$, $\Pr[1 \leftarrow \mathcal{B}^{|\mathsf{H}_c\rangle,\mathsf{R}_1,\mathsf{R}_2}] = \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{R}}]$.

We now use Lemma 2.1 (i.e., the generalized "One-Way To Hiding" lemma) for bounding the probability of $\mathcal{B}$ distinguishing the classical oracles $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$ from the oracles R_1 and R_2 — while having quantum access to oracle H_c — to complete the proof. Note that the task of distinguishing between the oracle tuples $(|\mathsf{H}_c\rangle, \mathsf{H}_c(\cdot \,\|\, K), \mathsf{H}_c(K \,\|\, \cdot))$ and $(|\mathsf{H}_c\rangle, \mathsf{R}_1, \mathsf{R}_2)$ is the same as distinguishing the tuples $(|\mathsf{G}\rangle, \mathsf{R}_1, \mathsf{R}_2)$ and $(|\mathsf{H}_c\rangle, \mathsf{R}_1, \mathsf{R}_2)$ where the quantum oracle G is obtained by reprogramming H_c as follows:

$$\mathsf{G}(y) = \begin{cases} \mathsf{R}_1(x) & \text{if } y \text{ is of the form } (x \,\|\, K) \text{ where} \\ & x \in \{0,1\}^{\ell_{\mathsf{tr}}}, \\ \mathsf{R}_2(x) & \text{if } y \text{ is of the form } (K \,\|\, x) \text{ where} \\ & x \in \{0,1\}^{2\ell_H+\ell_{\mathsf{label}}}, \\ \mathsf{H}_c(y) & \text{otherwise.} \end{cases}$$

In other words, the quantum oracles H_c and G behave the same on all inputs except those belonging to the set $S = \{(x \,\|\, K) \mid x \in \{0,1\}^{\ell_{\mathsf{tr}}}\} \cup \{(K \,\|\, x) \mid x \in \{0,1\}^{2\ell_H+\ell_{\mathsf{label}}}\}$. Hence from Lemma 2.1, the distinguishing advantage of any oracle algorithm making at-most q_{H_c} quantum queries either to $|\mathsf{H}_c\rangle$ or $|\mathsf{G}\rangle$ in the above tuples would be at most $2q_{\mathsf{H}_c}\sqrt{P_{\text{guess}}}$; in our setting, P_{guess} is essentially the probability of the event where measurement of a random i-th quantum query ($i \leftarrow\!\$\, [q_{\mathsf{H}_c}]$) of $\mathcal{B}^{|\mathsf{H}_c\rangle,\mathsf{R}_1,\mathsf{R}_2}$ falls in S. Since the random oracles H_c, R_1 and R_2 carry no information on key $K \leftarrow\!\$\, \{0,1\}^{\ell_K}$, the view of $\mathcal{B}^{|\mathsf{H}_c\rangle,\mathsf{R}_1,\mathsf{R}_2}$ — and hence, the measurement of its i-th quantum query — is independent of set S. Therefore, we have $P_{\text{guess}} \leq 2 \cdot 2^{-\ell_K}$ which results in the above distinguishing probability to be bounded by $q_{\mathsf{H}_c} \cdot 2^{-\left(\frac{\ell_K-3}{2}\right)}$. This finishes the proof. □

To obtain the final bound on $\mathsf{Adv}^{prf}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{A})$, all that remains is to prove the following lemma.

Lemma 4.4.

$$|\Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{F}}] - \Pr[1 \leftarrow \mathcal{A}^{|\mathsf{H}_c\rangle,\mathsf{R}}]| \leq q^2_{\mathsf{PRF}} \cdot 2^{-\ell_H}.$$

Proof. Note that the functions F and R are independent of the quantum random oracle H_c, and $\mathcal{A}$ is only allowed to access these functions *classically*. To argue about the above distinguishing probability, it helps to consider each component of the outputs of F and R, i.e., $(H, k_1, \ldots, k_6)$, separately.

Starting with the session ID component H, for each unique input x to F, we have $H \leftarrow \mathsf{R}_1(x)$ to be an independent and uniformly random output (since R_1 is a random function). This is also the case for R as it is a random function by definition. Hence the distributions of session ID outputs match in both F and R.

Moving to the session keys k_i ($i \in [6]$), first note that in F, any two components $k_i \leftarrow \mathsf{R}_2(\ldots \,\|\, \mathsf{label}[i] \,\|\, \ldots)$ and $k_j \leftarrow \mathsf{R}_2(\ldots \,\|\, \mathsf{label}[j] \,\|\, \ldots)$ where $i \neq j$ are independent of each other because of the label separation (i.e., $\mathsf{label}[i] \neq \mathsf{label}[j]$) in the random function R_2. This is also true for R. Hence we only have to focus on the individual distributions of k_i's. Now with respect to any fixed index $i \in [6]$, for each unique input x to R, the resulting k_i is independent and uniformly random. In the case of F, this is true for $k_i \leftarrow \mathsf{R}_2(\mathsf{R}_1(x) \,\|\, \mathsf{label}[i] \,\|\, \mathsf{R}_1(x))$ as well *unless* there are collisions in R_1. Since $\mathcal{A}$ can make at-most q_{PRF} classical queries to F — and hence, at-most q_{PRF} indirect classical queries to the random function R_1 with co-domain $\{0,1\}^{\ell_H}$ — the probability that $\mathcal{A}$ induces a collision in R_1 is upper-bounded by $\frac{q^2_{\mathsf{PRF}}}{2^{\ell_H}}$. This completes the proof. □

Hence we arrive at the above upper bound on $\mathsf{Adv}^{prf}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{A})$ by applying a triangle inequality on the bounds shown in Lemmas 4.3 and 4.4. □

5. IND-CPA v/s IND-CCA Secure KEMs in Post-quantum SSH

Our security analysis in the previous section essentially states that SSH is a post-quantum secure ACCE protocol when the hybrid KEM used in the handshake phase is IND-CPA secure (along with the other primitives, such as signatures, being secure according to our theorems in Subsection 4.1). And post-quantum IND-CPA security of the hybrid KEM follows tightly from IND-CPA security of the quantum-secure constituent KEM, as seen in Theorem 4.2. However as discussed in Subsection 4.2.1 above, such quantum-secure *ephemeral* KEMs used in real-world implementations of SSH, namely, Streamlined NTRU Prime and ML-KEM, employ variants of the FO transforms to achieve the stronger notion of IND-CCA security — which is more than what is required for the security of SSH in a formal sense. Also such FO-like

13. Note that the oracles $\mathsf{H}_c(\cdot \,\|\, K)$ and $\mathsf{H}_c(K \,\|\, \cdot)$ are appropriately domain separated because in the post-quantum SSH protocol, ℓ_{tr} is strictly greater than $2\ell_H + \ell_{\mathsf{label}}$.

transforms not only complicate cryptographic analyses of the resulting KEMs in post-quantum models such as the QROM, but they result in performance overheads too.

Hence in the context of post-quantum SSH, we suggest a much simpler transform for the quantum-secure constituent KEMs of the hybrid key exchange which achieves IND-CPA security in the QROM, and also avoids the aforementioned costs. We describe our transform in Figure 6.

$\mathsf{Encap}^\star(ek)$	$\mathsf{Decap}^\star(dk, ct)$
1 : $m \leftarrow\!\!\$\ \mathcal{M}$	1 : $m' \coloneqq \mathsf{Dec}(dk, ct)$
2 : $ct \coloneqq \mathsf{Enc}(ek, m)$	2 : **if** $m' = \perp$ **then return** $K \coloneqq \perp$
3 : $K \coloneqq \mathsf{H}(m)$	3 : **else return** $K \coloneqq \mathsf{H}(m')$
4 : **return** (ct, K)	

Figure 6. Our suggested transform to construct quantum-secure KEMs $(\mathsf{Gen}, \mathsf{Encap}^\star, \mathsf{Decap}^\star)$ for SSH's hybrid key exchange. Here $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is the starting PKE scheme, with message space $\mathcal{M}$, which is assumed to be OW-CPA secure ("Streamlined NTRU Prime Core" [8] is an example of such a PKE scheme); note that the constructed KEM uses the same Gen as the starting PKE scheme. We also have $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{256}$ to be a hash function. More importantly, note the lack of a "re-encryption check" in $\mathsf{Decap}^\star$, in contrast to FO-like transforms as seen in Figure 4.

In Appendix A, we formally prove IND-CPA security of a variant of Streamlined NTRU Prime called sntrup* which is obtained by applying our suggested transform to the sntrup-core PKE scheme. Our analysis in the QROM is much simpler compared to the one for the original Streamlined NTRU Prime, as stated in Theorem 4.3.

Theorem 5.1. *For any IND-CPA adversary $\mathcal{A}$ against the* sntrup* *KEM (described in Figure 6), there exists a OW-CPA adversary $\mathcal{B}$ against the* sntrup-core *PKE scheme such that*

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{sntrup}^\star}(\mathcal{A}) \leq 2\sqrt{\mathsf{Adv}^{\textit{ow-cpa}}_{\mathsf{sntrup\text{-}core}}(\mathcal{B})},$$

where $\mathcal{B}$ runs in about the same time as that of $\mathcal{A}$.

Our result can also be extended to ML-KEM in a straightforward way. But again as discussed w.r.t. Theorem 4.3 above, the resulting bounds would be relatively non-tight as the starting PKE scheme used in ML-KEM is *randomized*.

5.1. Experiments

We quantify our modification, that is removing the FO transform, by applying it to two post-quantum hybrid key exchange methods used in OpenSSH version 9.9p1, namely `sntrup761x25519-sha512` and `mlkem768x25519-sha256`.[14] We modified the code ourselves and provide the patches and experiment notes along with the three experiments we performed, available at [39]. All of our experiments were performed on a Linux 6.11.6 machine using a single CPU core (i.e., 12th Gen Intel Core i7-1265U@3.9GHz) and 16 GB RAM. Our findings are summarized in Tables 1 and 2.

14. Because the C code for the `mlkem768` constituent KEM of `mlkem768x25519-sha256` is machine-generated from `libcrux` [37], we instead link against `liboqs` [38] (commit `2ee908df`).

TABLE 1. CPU TIMINGS OF KEM AND HYBRID KEX WHICH MEASURE KeyGen, Encap, AND Decap EXECUTED ONE AFTER THE OTHER. NUMBER REPORTED IS THE MEAN OF 1000 SAMPLES, WITH THE SAMPLE VARIANCE BEING ABOUT 10^{-5} TIMES THE CORRESPONDING MEAN.

Scheme	IND-CCA [s]	IND-CPA [s]
`sntrup761`	$2.8164 \cdot 10^{-2}$	$2.7056 \cdot 10^{-2}$
`mlkem768`	$2.7853 \cdot 10^{-5}$	$1.3242 \cdot 10^{-5}$
`sntrup761x25519-sha512`	$3.0290 \cdot 10^{-2}$	$2.9105 \cdot 10^{-2}$
`mlkem768x25519-sha256`	$3.1412 \cdot 10^{-3}$	$3.1015 \cdot 10^{-3}$

TABLE 2. NETWORK TIMINGS WHICH MEASURE A SINGLE COMPLETE SSH CONNECTION. SAMPLE VARIANCE IS ABOUT 0.0001X THIS TIME.

Scheme	IND-CCA [s]	IND-CPA [s]
`sntrup761x25519-sha512`	0.1565	0.1534
`mlkem768x25519-sha256`	0.1325	0.1316

We find that, as expected, the difference in the CPU timings of the KEMs themselves is noticeable — around a 52% improvement in `mlkem768` where Encap is most costly, but around a 4% improvement in `sntrup761` where KeyGen is most costly. On the protocol level, however, the difference is barely noticeable (with respect to a single SSH connection) — below 2% for `sntrup761x25519-sha512` and below 1% for `mlkem768x25519-sha256`.[15]

On the other hand, it is worth pointing out that removing the FO transform from the hybrid KEX saves one from performing a significant number of hash function computations. Namely, in our modification of the KEMs we were able to remove four out of five hash function computations performed by the SSH server in Encap — that is, we only compute $\mathsf{H}(m)$ — and four out of five hash computations performed by the SSH client in KeyGen and Decap — that is, the key ek is no longer hashed, and we do not re-encrypt and only compute $\mathsf{H}(m)$. These benefits add up considering the large number of SSH connections that happen per day worldwide. For example, as pointed out in Meta's post-quantum transition update [15], "extra hashing steps" such as the ones used in the above KEMs because of the FO transform contribute to "non-negligible capacity cost" associated with deploying post-quantum key exchange that "*can cost hundreds of thousands or even millions of dollars a year*". Using our suggested IND-CPA secure KEMs thus reduces the required number of hash function computations from ten to two, helping to alleviate ecological and financial costs of transitioning to post-quantum cryptography on a larger scale.

References

[1] "OpenSSH," https://www.openssh.com/, accessed: 2024-11-06.

[2] M. Campagna and A. Petcher, "Security of hybrid key encapsulation," Cryptology ePrint Archive, Report 2020/1364, 2020. [Online]. Available: https://eprint.iacr.org/2020/1364

15. In fact, one can see from the experiment data provided with this paper that the improvement in the CPU timings of `mlkem768` is about the size of sample variation in the network timings of `mlkem768x25519-sha256`.

[3] A. Petcher and M. Campagna, "Security of hybrid key establishment using concatenation," Cryptology ePrint Archive, Report 2023/972, 2023. [Online]. Available: https://eprint.iacr.org/2023/972

[4] B. Blanchet and C. Jacomme, "Post-quantum sound cryptoverif and verification of hybrid tls and ssh key-exchanges," in *37th IEEE Computer Security Foundations Symposium, CSF 2024, Enschede, The Netherlands, July 8-12, 2024*. IEEE, 2024.

[5] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *CRYPTO'93*, ser. LNCS, D. R. Stinson, Ed., vol. 773. Springer, Berlin, Heidelberg, Aug. 1994, pp. 232–249.

[6] F. Bergsma, B. Dowling, F. Kohlar, J. Schwenk, and D. Stebila, "Multi-ciphersuite security of the Secure Shell (SSH) protocol," in *ACM CCS 2014*, G.-J. Ahn, M. Yung, and N. Li, Eds. ACM Press, Nov. 2014, pp. 369–381.

[7] "TinySSH," https://tinyssh.org/, accessed: 2024-11-06.

[8] D. J. Bernstein, B. B. Brumley, M.-S. Chen, C. Chuengsatiansup, T. Lange, A. Marotzke, B.-Y. Peng, N. Tuveri, C. van Vredendaal, and B.-Y. Yang, "NTRU Prime," National Institute of Standards and Technology, Tech. Rep., 2020, available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[9] National Institute of Standards and Technology (NIST), "Module-lattice-based key-encapsulation mechanism standard," U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS PUBS) 203, Published August 13, 2024, 2024.

[10] K. Xagawa, "Anonymity of NIST PQC round 3 KEMs," in *EUROCRYPT 2022, Part III*, ser. LNCS, O. Dunkelman and S. Dziembowski, Eds., vol. 13277. Springer, Cham, May / Jun. 2022, pp. 551–581.

[11] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, "Random oracles in a quantum world," in *ASIACRYPT 2011*, ser. LNCS, D. H. Lee and X. Wang, Eds., vol. 7073. Springer, Berlin, Heidelberg, Dec. 2011, pp. 41–69.

[12] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS 93*, D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, Eds. ACM Press, Nov. 1993, pp. 62–73.

[13] L. Huguenin-Dumittan and S. Vaudenay, "On IND-qCCA security in the ROM and its applications - CPA security is sufficient for TLS 1.3," in *EUROCRYPT 2022, Part III*, ser. LNCS, O. Dunkelman and S. Dziembowski, Eds., vol. 13277. Springer, Cham, May / Jun. 2022, pp. 613–642.

[14] B. Zhou, H. Jiang, and Y. Zhao, "CPA-secure KEMs are also sufficient for post-quantum TLS 1.3," 2024, to appear at ASIACRYPT 2024. [Online]. Available: https://eprint.iacr.org/2024/1360

[15] B. Westerbaan, [X (Twitter)], https://x.com/bwesterb/status/1771958142147973390, 2024.

[16] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, "On the security of TLS-DHE in the standard model," in *CRYPTO 2012*, ser. LNCS, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, Berlin, Heidelberg, Aug. 2012, pp. 273–293.

[17] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the TLS protocol: A systematic analysis," in *CRYPTO 2013, Part I*, ser. LNCS, R. Canetti and J. A. Garay, Eds., vol. 8042. Springer, Berlin, Heidelberg, Aug. 2013, pp. 429–448.

[18] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[19] D. Unruh, "Revocable quantum timed-release encryption," in *EUROCRYPT 2014*, ser. LNCS, P. Q. Nguyen and E. Oswald, Eds., vol. 8441. Springer, Berlin, Heidelberg, May 2014, pp. 129–146.

[20] A. Ambainis, M. Hamburg, and D. Unruh, "Quantum security proofs using semi-classical oracles," in *CRYPTO 2019, Part II*, ser. LNCS, A. Boldyreva and D. Micciancio, Eds., vol. 11693. Springer, Cham, Aug. 2019, pp. 269–295.

[21] N. Bindel, M. Hamburg, K. Hövelmanns, A. Hülsing, and E. Persichetti, "Tighter proofs of CCA security in the quantum random oracle model," in *TCC 2019, Part II*, ser. LNCS, D. Hofheinz and A. Rosen, Eds., vol. 11892. Springer, Cham, Dec. 2019, pp. 61–90.

[22] C. M. Lonvick and T. Ylonen, "RFC 4253: The secure shell (SSH) transport layer protocol," Jan. 2006. [Online]. Available: https://datatracker.ietf.org/doc/rfc4253/

[23] ——, "RFC 4252: The secure shell (SSH) authentication protocol," Jan. 2006. [Online]. Available: https://datatracker.ietf.org/doc/rfc4252/

[24] "OQS-OpenSSH," https://github.com/open-quantum-safe/openssh, accessed: 2024-11-06.

[25] F. Giacon, F. Heuer, and B. Poettering, "KEM combiners," in *PKC 2018, Part I*, ser. LNCS, M. Abdalla and R. Dahab, Eds., vol. 10769. Springer, Cham, Mar. 2018, pp. 190–218.

[26] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Hybrid key encapsulation mechanisms and authenticated key exchange," in *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, J. Ding and R. Steinwandt, Eds. Springer, Cham, 2019, pp. 206–226.

[27] M. Zhandry, "Secure identity-based encryption in the quantum random oracle model," in *CRYPTO 2012*, ser. LNCS, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, Berlin, Heidelberg, Aug. 2012, pp. 758–775.

[28] M. Friedl, J. Mojzis, and S. Josefsson, "Secure shell (SSH) key exchange method using hybrid Streamlined NTRU Prime sntrup761 and X25519 with SHA-512: sntrup761x25519-sha512, ver. 3," Aug. 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-josefsson-ntruprime-ssh/

[29] D. J. Bernstein, "Curve25519: New Diffie-Hellman speed records," in *PKC 2006*, ser. LNCS, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds., vol. 3958. Springer, Berlin, Heidelberg, Apr. 2006, pp. 207–228.

[30] P. Kampanakis, D. Stebila, and T. Hansen, "PQ/T hybrid key exchange in SSH, ver. 4," Aug. 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-kampanakis-curdle-ssh-pq-ke/

[31] L. Chen, D. Moody, A. Regenscheid, A. Robinson, and K. Randall, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST Special Publication (SP) 800-186, 2023.

[32] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *CRYPTO'99*, ser. LNCS, M. J. Wiener, Ed., vol. 1666. Springer, Berlin, Heidelberg, Aug. 1999, pp. 537–554.

[33] A. W. Dent, "A designer's guide to KEMs," in *9th IMA International Conference on Cryptography and Coding*, ser. LNCS, K. G. Paterson, Ed., vol. 2898. Springer, Berlin, Heidelberg, Dec. 2003, pp. 133–151.

[34] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki-Okamoto transformation," in *TCC 2017, Part I*, ser. LNCS, Y. Kalai and L. Reyzin, Eds., vol. 10677. Springer, Cham, Nov. 2017, pp. 341–371.

[35] T. Saito, K. Xagawa, and T. Yamakawa, "Tightly-secure key-encapsulation mechanism in the quantum random oracle model," in *EUROCRYPT 2018, Part III*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10822. Springer, Cham, Apr. / May 2018, pp. 520–551.

[36] N. Alamati, V. Maram, and D. Masny, "Non-observable quantum random oracle model," in *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023*, T. Johansson and D. Smith-Tone, Eds. Springer, Cham, Aug. 2023, pp. 417–444.

[37] "libcrux," https://cryspen.com/libcrux/, accessed: 2024-11-12.

[38] "liboqs," https://github.com/open-quantum-safe/liboqs, accessed: 2024-11-12.

[39] pq-ssh bench, "SnP25-experiments." [Online]. Available: https://github.com/pq-ssh-bench/SnP25-experiments

[40] E. E. Targhi and D. Unruh, "Post-quantum security of the Fujisaki-Okamoto and OAEP transforms," in *TCC 2016-B, Part II*, ser. LNCS, M. Hirt and A. D. Smith, Eds., vol. 9986. Springer, Berlin, Heidelberg, Oct. / Nov. 2016, pp. 192–216.

[41] D. J. Bernstein, "Re: [pqc-forum] anonymity of KEMs in the QROM," 2021. [Online]. Available: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/8k3MhD_5stk/m/TWGKtuL4BgAJ

[42] H. Jiang, Z. Zhang, and Z. Ma, "Key encapsulation mechanism with explicit rejection in the quantum random oracle model," in *PKC 2019, Part II*, ser. LNCS, D. Lin and K. Sako, Eds., vol. 11443. Springer, Cham, Apr. 2019, pp. 618–645.

[43] V. Kuchta, A. Sakzad, D. Stehlé, R. Steinfeld, and S. Sun, "Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security," in *EUROCRYPT 2020, Part III*, ser. LNCS, A. Canteaut and Y. Ishai, Eds., vol. 12107. Springer, Cham, May 2020, pp. 703–728.

[44] M. Zhandry, "How to record quantum queries, and applications to quantum indifferentiability," in *CRYPTO 2019, Part II*, ser. LNCS, A. Boldyreva and D. Micciancio, Eds., vol. 11693. Springer, Cham, Aug. 2019, pp. 239–268.

Appendix A. Omitted Proofs

A.1. Proof of Lemma 4.1

Proof. **Game 0.** The game equals the ACCE security experiment described in Section 2.4. Thus, $\mathsf{Adv}^{\textit{acce-so-auth}}_{\mathsf{SSH}}(\mathcal{A}) = \Pr(\mathsf{break}_0^{(0)})$.

Game 1. In this game we abort if any two nonces r_i, r_j collide. Specifically the challenger collects a list L of all nonces r_i sampled by the challenger during the simulation. If any nonce appears twice, we abort the simulation. Thus $\Pr(\mathsf{break}_0^{(0)}) \leq \Pr(\mathsf{break}_1^{(0)}) + \frac{(n_P n_S)^2}{2^\mu}$.

Game 2. In this game we abort if, during the experiment, any two hash inputs collide to the same output. We do so by initialising a collision-resistance challenger at the beginning of the experiment. Our reduction $\mathcal{B}_1$ maintains a list Coll, where all the input/output pairs of all executions of the hash function H are recorded. $\mathcal{B}_1$ aborts if at any time a pair $(in, \mathsf{H}(in))$ is added to Coll such that there already exists an entry $(in', \mathsf{H}(in'))$ in Coll with $\mathsf{H}(in) = \mathsf{H}(in')$ but $in \neq in'$. Whenever $\mathcal{A}$ causes a collision, $\mathcal{B}_1$ inspects Coll and outputs this collision to the hash collision-resistance challenger. Since $\mathcal{B}_1$ finds a collision, we have that $\Pr(\mathsf{break}_1^{(0)}) \leq \Pr(\mathsf{break}_2^{(0)}) + \mathsf{Adv}^{\textit{coll-res}}_{\mathsf{H}_c}(\mathcal{B}_1^{\mathcal{A}})$.

Game 3. In this game, we guess the session the first session that accepts maliciously. Since we are considering server-only authentication we note that only client sessions can accept maliciously. Thus we guess two indices $(i, s) \in [n_P] \times [n_S]$ and abort if the adversary causes some other session π_j^t to accept maliciously before π_i^s. This happens with probability $\frac{1}{n_P n_S}$. Thus we have: $\Pr(\mathsf{break}_2^{(0)}) \leq n_P n_S \cdot \Pr(\mathsf{break}_3^{(0)})$. Note that in what follows π_i^s accepts maliciously, and thus by Definition 2.4 $\mathcal{A}$ cannot issue $\mathsf{OCorrupt}(\pi_i^s.\mathsf{pid})$.

Game 4. In this game we exclude signature forgeries. We abort the simulation if some session π_i^s accepts after it receives a signature which was never output of a session with a matching session identifier. Note that we have excluded nonce and hash collisions, so from now on all values to be signed are different. Thus any abort event is related to a signature forgery.

Technically, we construct an algorithm $\mathcal{B}_2^{\mathcal{A}}$ which simulates the SSH protocol as in Game 3. $\mathcal{B}_2$ interacts with $\mathcal{A}$. $\mathcal{B}_2$ receives a public key vk from an EUF-CMA signature challenger for DSS, guesses which public key vk_{j^*} the session will use to verify the signature (which costs us a factor n_P in the reduction) and sets $vk_{j^*} = vk$. Since the signing key has to be uncorrupted it is no problem for the reduction that the secret signing key is unknown. If $\mathcal{B}_2$ needs to sign a message on behalf of party P_{j^*}, it makes a signing query to the EUF-CMA challenger. If the session π_i^s maliciously accepts in the sense of definition 2.4 in Game 4, we know from the discussion above that the maliciously accepting session has verified a signature σ' over a session ID H where there is no session $\pi_{j^*}^t$ with the same session ID, thus this signature was not generated with a call to the signature challenger. Thus $\mathcal{B}_2$ has found (H, σ') as a signature forgery, so $\Pr(\mathsf{break}_3^{(0)}) \leq \Pr(\mathsf{break}_4^{(0)}) + n_P \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{DSS}}(\mathcal{B}_2^{\mathcal{A}})$. We note that at this point, for a client session π_i^s, all messages sent up to and including `KEX_KEM_REPLY` are authenticated via the server signature. Now we ensure all other messages in the handshake are authenticated via the BSAE scheme.

Game 5. In this game we introduce an abort event that triggers if the test session π_i^s is a client and accepts a KEM ciphertext ct but ct was not generated by an honest session owned by $\pi_i^s.\mathsf{pid}$. Note that here we are in server-only authentication mode, so if the test session π_i^s is a server and the adversary injects a KEM public key ek that is not from an honest session then it will lose the game. Note that at the point the test session π_i^s accepts the KEM ciphertext, neither its communicating partner nor itself has accepted the session and thus the adversary cannot issue $\mathsf{OCorrupt}(\pi_i^s.\mathsf{pid})$ without losing the game. Since π_i^s will only accept ct after verifying a signature σ_S over $\mathsf{H}_c(x \| K)$ where $ct \subset x$, and by Game 4 we exclude signature forgeries and by Game 2 we exclude hash collisions we find: $\Pr(\mathsf{break}_4^{(0)}) = \Pr(\mathsf{break}_5^{(0)})$.

Game 6. In this game we replace the value $K, ct = \mathsf{KEM.Encap}(ek)$, $K = \mathsf{KEM.Decap}(dk, ct)$ computed by π_i^s and π_j^t with a uniformly random value K^*. Specifically, we introduce a reduction $\mathcal{B}_3$ that interacts with $\mathcal{A}$ and embeds an IND-CPA challenge into the test session's transcript. Note that in what follows, we assume WLOG that the test session π_i^s is the client. The alternative case (where π_i^s is the server) follows identically up to a change in notation.

At the beginning of the experiment, $\mathcal{B}_3$ initialises an IND-CPA KEM challenger. When π_i^s needs to compute a KEM public key pair, $\mathcal{B}_3$ replaces the honest computation of (ek, dk) with (ek^*, dk^*) output by the IND-CPA challenger. Similarly, when the server session π_j^t needs to encapsulate a shared secret, $\mathcal{B}_3$ replaces K, ct with the key output K^* and the ciphertext output ct^* from the IND-CPA challenger. Finally, when π_i^s needs to compute K, $\mathcal{B}_3$ replaces K with

K^*. Since we have excluded maliciously accepting sessions, and π_i^s and π_j^t communicate without any modification from the adversary.

Any adversary $\mathcal{A}$ that can distinguish this game from the previous game can directly be used to construct an adversary $\mathcal{B}_3^{\mathcal{A}}$ that can break the IND-CPA assumption: If the challenge bit b sampled by the IND-CPA KEM challenger is 0, then $K^* = \mathsf{Decap}(dk, ct)$ and we are in Game 5. If the challenge bit b sampled by the IND-CPA KEM challenger is 1, then K^* is instead uniformly random and independent of the protocol flow and we are in Game 6. Thus $\Pr(\mathsf{break}_5^{(0)}) \leq \Pr(\mathsf{break}_6^{(0)}) + \mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}}(\mathcal{B}_3^{\mathcal{A}})$.

Game 7. In this game we replace the values $H, k_1, ..., k_6$ computed by π_i^s and π_j^t as $\mathsf{PRF}_{\mathsf{SSH}}(K^*, sid)$ with random values $H^*, k_1^*, ..., k_6^*$. Any adversary $\mathcal{A}$ that can distinguish this game from the previous game can directly be used to construct an adversary $\mathcal{B}_4^{\mathcal{A}}$ that can break the PRF assumption: let $S = H\|k_1\|...\|k_6$ be the output of $\mathsf{PRF}_{\mathsf{SSH}}$, and let $S^* = H^*\|k_1^*\|...\|k_6^*$ be a random string of the same length. For S we are in Game 6, and for S^* in Game 7. Thus $\Pr(\mathsf{break}_6^{(0)}) \leq \Pr(\mathsf{break}_7^{(0)}) + \mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{B}_4^{\mathcal{A}})$.

Game 8. In this game, we introduce an abort event that triggers if π_i^s decrypts `AUTHOK` or `AUTH_SUCCESS` ciphertexts (keyed by $k_1^* \dots k_6^*$ but the ciphertext was not output by an honest session owned by $\pi_i^s.\mathsf{pid}$.

Specifically, we specify a reduction $\mathcal{B}_5$ that proceeds identically as in Game 7, but at the beginning of the experiment initialises an *bsae* challenger, which $\mathcal{B}_5$ queries when π_i^s needs to encrypt or decrypt using $k_1^* \dots k_6^*$. The abort event only triggers if $\mathcal{A}$ can produce a valid ciphertext that decrypts under $k_1^* \dots k_6^*$, and we can submit the ciphertext to the *bsae* challenger, causing $\mathsf{phase} \leftarrow 1$ in the *bsae* security, allowing a trivial win by asking for an encryption of m_0, m_1, and submitting the challenge ciphertext to the Dec oracle, recovering m_b and allowing $\mathcal{B}_5$ to win the game. By Game 7 $k_1^* \dots k_6^*$ are already uniformly random and independent and this replacement is sound.

Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the *bsae* security of the BSAE scheme. Thus: $\Pr(\mathsf{break}_7^{(0)}) \leq \Pr(\mathsf{break}_8^{(0)}) + \mathsf{Adv}^{\textit{bsae}}_{\mathsf{BSAE}}(\mathcal{B}_5^{\mathcal{A}})$.

Final analysis. Now all signatures are computed by legitimate parties only, and are all computed for different session IDs, which allows π_i^s to verify that `KEXREPLY`, `KEX_KEM_REPLY` come from an honest server party. Additionally, the game aborts when `AUTHOK` or `AUTH_SUCCESS` are forged to π_i^s. Thus there is no way for π_i^s to accept maliciously, and we have $\Pr(\mathsf{break}_8^{(0)}) = 0$. □

A.2. Proof of Lemma 4.2

Proof. Let $\mathsf{break}_\delta^{(1)}$ be the event that occurs when $\mathcal{A}$ answers the encryption challenge correctly in Game δ in the sense of Definition 2.5.

Game 0. This game equals the ACCE channel security experiment described in Section 2.4.

Game 1. In this game we abort if the test session π_i^s accepts maliciously, and thus proceeds identically to the proof of Lemma 4.1. Thus, in this game any session that accepts non-maliciously in the sense of Definition 2.4 has a unique uncorrupted partner session. From the previous proof, we have $\Pr(\mathsf{break}_0^{(1)}) \leq \Pr(\mathsf{break}_1^{(1)}) + \mathsf{Adv}^{\textit{acce-so-aenc}}_{\mathsf{SSH}}(\mathcal{A})$. From now on, we always have a matching session for the session π_i^s where the adversary tries to guess the random bit: for server sessions through Definition 2.4, and for client sessions through this game.

Game 2. In this game, we guess the session for which the adversary outputs the bit b'. We guess two indices $(i, s), (j, t) \in [n_P] \times [n_S]$ and abort if the adversary outputs (i^*, s^*, b') with $(i^*, s^*) \neq (i, s)$, or has a unique matching partner with session $\pi_{j^*}^{t^*}$, but $(j^*, t^*) \neq (j, t)$. This happens with probability $\frac{1}{n_P^2 n_S^2}$. By the definition of the channel security experiment we have that there exists a unique partner session π_j^t which can be easily determined by the simulator. Thus we have: $\Pr(\mathsf{break}_1^{(1)}) \leq n_P^2 n_S^2 \cdot \Pr(\mathsf{break}_2^{(1)})$.

Game 3. In this game we replace the value $K, ct = \mathsf{KEM.Encap}(ek)$, $K = \mathsf{KEM.Decap}(dk, ct)$ computed by π_i^s and π_j^t with a uniformly random value K^*. Specifically, we introduce a reduction $\mathcal{B}_6$ that interacts with $\mathcal{A}$ and embeds an IND-CPA challenge into the test session's transcript. Note that in what follows, we assume WLOG that the test session π_i^s is the client. The alternative case (where π_i^s is the server) follows identically up to a change in notation.

At the beginning of the experiment, $\mathcal{B}_6$ initialises an IND-CPA KEM challenger. When π_i^s needs to compute a KEM public key pair, $\mathcal{B}_6$ replaces the honest computation of (ek, dk) with (ek^*, dk^*) output by the IND-CPA challenger. Similarly, when the server session π_j^t needs to encapsulate a shared secret, $\mathcal{B}_6$ replaces K, ct with the key output K^* and the ciphertext output ct^* from the IND-CPA challenger. Finally, when π_i^s needs to compute K, $\mathcal{B}_6$ replaces K with K^*. Since we have excluded maliciously accepting sessions, and π_i^s and π_j^t communicate without any modification from the adversary.

Any adversary $\mathcal{A}$ that can distinguish this game from the previous game can directly be used to construct an adversary $\mathcal{B}_6^{\mathcal{A}}$ that can break the IND-CPA assumption: If the challenge bit b sampled by the IND-CPA KEM challenger is 0, then $K^* = \mathsf{Decap}(dk, ct)$ and we are in Game 2. If the challenge bit b sampled by the IND-CPA KEM challenger is 1, then K^* is instead uniformly random and independent of the protocol flow and we are in Game 3. Thus $\Pr(\mathsf{break}_2^{(1)}) \leq \Pr(\mathsf{break}_3^{(1)}) + \mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}}(\mathcal{B}_6^{\mathcal{A}})$.

Game 4. In this game we replace the values $H, k_1, ..., k_6$ computed by π_i^s and π_j^t as $\mathsf{PRF}_{\mathsf{SSH}}(K^*, sid)$ with random values $H^*, k_1^*, ..., k_6^*$. Any adversary $\mathcal{A}$ that can distinguish this game from the previous game can directly be used to construct an adversary $\mathcal{B}_7^{\mathcal{A}}$ that can break the PRF assumption: let $S = H\|k_1\|...\|k_6$ be the output of $\mathsf{PRF}_{\mathsf{SSH}}$, and let $S^* = H^*\|k_1^*\|...\|k_6^*$ be a random string of the same length. For S we are in Game 3, and for S^* in Game 4. Thus $\Pr(\mathsf{break}_3^{(1)}) \leq \Pr(\mathsf{break}_4^{(1)}) + \mathsf{Adv}^{\textit{prf}}_{\mathsf{PRF}_{\mathsf{SSH}}}(\mathcal{B}_7^{\mathcal{A}})$.

Final analysis. We now have that the keys $k_1^*, ..., k_6^*$ are uniformly random and independent from the protocol transcript. Thus any adversary $\mathcal{A}$ that can guess (i^*, s^*, b')

correctly can directly be used to construct an adversary $\mathcal{B}_8^{\mathcal{A}}$ that breaks the BSAE scheme. Technically we exploit the fact that all keys for the encryption scheme are now independent from the handshake and embed a BSAE challenger to answer $\mathcal{A}$'s queries for Enc, Dec. Now $\mathcal{B}_8$ simply forwards $\mathcal{Q}$'s output to the challenger and thus we have $\Pr(\mathsf{break}_4^{(1)}) \leq \mathsf{Adv}^{bsae}_{\mathsf{BSAE}}(\mathcal{B}_8^{\mathcal{A}})$. □

A.3. Proof of Theorem 4.3

Proof. The proof proceeds with a sequence of game hops ($\mathbf{G}_0 \rightarrow \ldots \rightarrow \mathbf{G}_4$) with respect to adversary $\mathcal{A}$. The games $\mathbf{G}_0$ and $\mathbf{G}_4$ are basically the IND-CPA security games for sntrup KEM where $\mathcal{A}$ gets the real encapsulated key $K^* := \mathsf{H}_1(r, ct^*)$ with $r := \mathsf{H}_3(m)$ as in Figure 4 and a random encapsulated key $K^* \leftarrow\$ \{0,1\}^{256}$ respectively. We hence have

$$\mathsf{Adv}^{ind\text{-}cpa}_{\mathsf{sntrup}}(\mathcal{A}) = |\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_4]|.$$

In game $\mathbf{G}_1$, we program the random oracle H_3 on a single input, i.e., m, by replacing $r := \mathsf{H}_3(m)$ with a uniformly random value $r \leftarrow\$ \{0,1\}^{256}$ which is independent of H_3. We will use the double-sided OW2H lemma (see Lemma 2.3 above) to bound $|\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_1]|$. In the context of Lemma 2.3, let G_3 be the quantum oracle that is obtained by reprogramming H_3 as above: i.e., $\mathsf{G}_3(x) = \mathsf{H}_3(x)$ for all $x \neq m$, and $\mathsf{G}_3(m) := r$ for a uniformly random $r \leftarrow\$ \{0,1\}^{256}$ which is independent of H_3. Also let $\mathcal{D}_{\mathcal{A}}$ be a quantum oracle algorithm which has access to either H_3 or G_3, and takes as input $z = (ek, ct_0^*, \mathsf{H}_3(m))$ where $(ek, dk) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \leftarrow\$ \mathcal{M}$ and $ct_0^* := \mathsf{Enc}(ek, m)$.

It is not hard to see that $\mathcal{D}_{\mathcal{A}}^{|\mathsf{H}_3\rangle}(z)$ (resp. $\mathcal{D}_{\mathcal{A}}^{|\mathsf{G}_3\rangle}(z)$) can perfectly simulate the game $\mathbf{G}_0$ (resp. $\mathbf{G}_1$) towards $\mathcal{A}$. Specifically, note that $\mathcal{D}_{\mathcal{A}}^{|\mathsf{G}_3\rangle}(z)$ uses the $\mathsf{H}_3(m)$-part of its input z as "r" to derive ct_1^* and K^* as in Figure 4, and uses the oracle G_3 to respond to $\mathcal{A}$'s H_3-queries; but from $\mathcal{A}$'s perspective, this is virtually the same as in $\mathbf{G}_1$. By having $\mathcal{D}_{\mathcal{A}}$ output the same bit as $\mathcal{A}$ at the end of the game, from Lemma 2.3 we have

$$|\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_1]| \leq 2\sqrt{\Pr[m \leftarrow \mathcal{D}_{\mathcal{B}}^{|\mathsf{H}_3\rangle,|\mathsf{G}_3\rangle}(z)]}$$

for a quantum oracle algorithm $\mathcal{D}_{\mathcal{B}}$ — as described in Lemma 2.3 — which runs in about the same time as that of $\mathcal{D}_{\mathcal{A}}$, and hence, also $\mathcal{A}$. Using $\mathcal{D}_{\mathcal{B}}$, we will now construct a OW-CPA adversary $\mathcal{B}$ against the sntrup-core PKE scheme such that $\mathsf{Adv}^{ow\text{-}cpa}_{\mathsf{sntrup\text{-}core}}(\mathcal{B}) = \Pr[m \leftarrow \mathcal{D}_{\mathcal{B}}^{|\mathsf{H}_3\rangle,|\mathsf{G}_3\rangle}(z)]$.

Upon receiving the pair (ek, ct_0^*) from its OW-CPA challenger — where $(ek, dk) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \leftarrow\$ \mathcal{M}$ and $ct_0^* := \mathsf{Enc}(ek, m)$ — $\mathcal{B}$ samples $r \leftarrow\$ \{0,1\}^{256}$ uniformly at random and forwards the input $z = (ek, ct_0^*, r)$ to $\mathcal{D}_{\mathcal{B}}$. To simulate *both* the QROs H_3 and G_3 towards $\mathcal{D}_{\mathcal{B}}$, $\mathcal{B}$ first uses a $2q$-wise independent function to simulate G_3 (as in [27]) where q is the number of quantum queries made by $\mathcal{D}_{\mathcal{B}}$ to its oracles. Now here is the important part. Since sntrup-core is a perfectly-correct *deterministic* PKE scheme, $\mathcal{B}$ simulates H_3 by first checking if $\mathcal{D}_{\mathcal{B}}$'s H_3-query x satisfies $\mathsf{Enc}(ek, x) = ct_0^*$: if it does, it means that $x = m$ and $\mathcal{B}$ returns r as the response of H_3; otherwise, $x \neq m$ and $\mathcal{B}$ forwards $\mathsf{G}_3(x)$ ($\mathcal{B}$ can do this simulation in quantum superposition using an appropriate unitary mapping in a straightforward way). After $\mathcal{D}_{\mathcal{B}}$ completes its execution and returns a value, $\mathcal{B}$ forwards it to its OW-CPA challenger. Since $\mathcal{B}$ perfectly simulates $\mathcal{D}_{\mathcal{B}}^{|\mathsf{H}_3\rangle,|\mathsf{G}_3\rangle}(z)$, we have $\mathsf{Adv}^{ow\text{-}cpa}_{\mathsf{sntrup\text{-}core}}(\mathcal{B}) = \Pr[m \leftarrow \mathcal{D}_{\mathcal{B}}^{|\mathsf{H}_3\rangle,|\mathsf{G}_3\rangle}(z)]$; also note that the running time of $\mathcal{B}$ is about the same as that of $\mathcal{D}_{\mathcal{B}}$, and hence, also that of $\mathcal{A}$.

In game $\mathbf{G}_2$, we replace the values $ct_1^* := \mathsf{H}_2(r, \mathsf{H}_4(ek))$ and $K^* := \mathsf{H}_1(r, (ct_0^*, ct_1^*)))$ with two independent and uniformly random values in $\{0,1\}^{256}$. Consider the QRO $(\mathsf{H}_1 \times \mathsf{H}_2)(\cdot) = (\mathsf{H}_1(\cdot), \mathsf{H}_2(\cdot))$. For the above replacement, we are effectively relying on the fact that $|(\mathsf{H}_1 \times \mathsf{H}_2)(r, \cdot)\rangle$ is a secure PRF (see Lemma 2.2) where we have the secret "PRF key" to be $r \leftarrow\$ \{0,1\}^{256}$ following game $\mathbf{G}_1$. To argue this more formally via a reduction, we note that a PRF-adversary with access to $|\mathsf{H}_1 \times \mathsf{H}_2\rangle$ can respond to $\mathcal{A}$'s *individual* quantum queries to $|\mathsf{H}_1\rangle$ (resp. to $|\mathsf{H}_2\rangle$) by initially preparing a 256-qubit uniform superposition in the output register corresponding to $|\mathsf{H}_2\rangle$ (resp. to $|\mathsf{H}_1\rangle$); this would make it "unentangled" with the $|\mathsf{H}_1\rangle$-register (resp. $|\mathsf{H}_2\rangle$-register), thereby allowing the PRF-adversary to ignore the unentangled output following $\mathcal{A}$'s individual query (also see [40, Footnote 1]). With this observation, we can use a straightforward reduction to the PRF security of $|(\mathsf{H}_1 \times \mathsf{H}_2)(r, \cdot)\rangle$ as described in Lemma 2.2 to obtain

$$|\Pr[1 \leftarrow \mathbf{G}_1] - \Pr[1 \leftarrow \mathbf{G}_2]| \leq \frac{2(q_{\mathsf{H}_1} + q_{\mathsf{H}_2})}{2^{128}}.$$

(In such a reduction, the PRF-adversary makes a query to $|\mathsf{H}_1 \times \mathsf{H}_2\rangle$ each time $\mathcal{A}$ makes an individual H_1-query or H_2-query; hence the PRF-adversary effectively makes at-most $(q_{\mathsf{H}_1} + q_{\mathsf{H}_2})$ queries.)

In game $\mathbf{G}_3$, we revert the change introduced in $\mathbf{G}_2$ w.r.t. ct_1^*, while still keeping the change w.r.t. K^* intact— i.e., we now have $ct_1^* := \mathsf{H}_2(r, \mathsf{H}_4(ek))$ once again instead of $ct_1^* \leftarrow\$ \{0,1\}^{256}$. By having $r \leftarrow\$ \{0,1\}^{256}$ to be our secret PRF key in "$ct_1^* := \mathsf{H}_2(r, \mathsf{H}_4(ek))$", we can use Lemma 2.2 for the PRF security of $|\mathsf{H}_2\rangle$ to get

$$|\Pr[1 \leftarrow \mathbf{G}_2] - \Pr[1 \leftarrow \mathbf{G}_3]| \leq \frac{2q_{\mathsf{H}_2}}{2^{128}}.$$

Finally in game $\mathbf{G}_4$, we *re*program the random oracle H_3 on the single input m by replacing back $r \leftarrow\$ \{0,1\}^{256}$ with $r := \mathsf{H}_3(m)$, thereby essentially reverting the change introduced in $\mathbf{G}_1$. By invoking the double-sided OW2H lemma (Lemma 2.3) in a similar way as in the above "$\mathbf{G}_0 \rightarrow \mathbf{G}_1$" hop, it is not hard to obtain another OW-CPA adversary $\mathcal{B}'$ against the underlying sntrup-core, with about the same running time as that of $\mathcal{A}$, such that

$$|\Pr[1 \leftarrow \mathbf{G}_3] - \Pr[1 \leftarrow \mathbf{G}_4]| \leq 2\sqrt{\mathsf{Adv}^{ow\text{-}cpa}_{\mathsf{sntrup\text{-}core}}(\mathcal{B}')}.$$

Collecting the above bounds finishes the proof. □

$\mathsf{Gen}'(1^\lambda)$

1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\lambda)$
2 : $\bar{s} \leftarrow\$ \{0,1\}^\ell$
3 : $\overline{dk} \coloneqq (dk, ek, \bar{s})$
4 : **return** $(ek, \overline{dk})$

$\mathsf{Encap}'(ek)$

1 : $m \leftarrow\$ \mathcal{M}$
2 : $ct_0 \coloneqq \mathsf{Enc}(ek, m)$
3 : $ct_1 \coloneqq \bar{\mathsf{H}}_2(m, \mathsf{H}_4(ek))$
4 : $ct \coloneqq (ct_0, ct_1)$
5 : $K \coloneqq \bar{\mathsf{H}}_1(m, ct)$
6 : **return** (ct, K)

$\mathsf{Decap}'(\overline{dk}, ct)$

1 : Parse $(dk, ek, \bar{s}) \leftarrow \overline{dk}, (ct_0, ct_1) \leftarrow ct$
2 : $m' \coloneqq \mathsf{Dec}(dk, ct_0)$
3 : **if** $m' = \perp$ **then return** $K \coloneqq \mathsf{H}_0(\bar{s}, ct)$
4 : $ct_0' \coloneqq \mathsf{Enc}(ek, m')$
5 : $ct_1' \coloneqq \bar{\mathsf{H}}_2(m', \mathsf{H}_4(ek))$
6 : **if** $(ct_0, ct_1) \neq (ct_0', ct_1')$ **then return** $K \coloneqq \mathsf{H}_0(\bar{s}, ct)$
7 : **else return** $K \coloneqq \bar{\mathsf{H}}_1(m', ct)$

Figure 7. $\mathsf{sntrup}' = (\mathsf{Gen}', \mathsf{Encap}', \mathsf{Decap}')$, the simplified FO-type transform used in Streamlined NTRU Prime with random oracles $\mathsf{H}_0, \mathsf{H}_4, \bar{\mathsf{H}}_1, \bar{\mathsf{H}}_2 \colon \{0,1\}^* \to \{0,1\}^{256}$.

A.4. Proof of Theorem 5.1

Proof. The proof simply consists of a single game hop ($\mathbf{G}_0 \to \mathbf{G}_1$). The games $\mathbf{G}_0$ and $\mathbf{G}_1$ are basically the IND-CPA security games for $\mathsf{sntrup}^\star$ where $\mathcal{A}$ gets the real encapsulated key $K^* \coloneqq \mathsf{H}(m)$ as in Figure 6 and a random encapsulated key $K^* \leftarrow\$ \{0,1\}^{256}$ respectively. We therefore have

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\mathsf{sntrup}^\star}(\mathcal{A}) = |\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_1]|.$$

Note that in $\mathbf{G}_1$, we are essentially programming the QRO H on a single input, i.e., m, by replacing $K \coloneqq \mathsf{H}(m)$ with a uniformly random key $K \leftarrow\$ \{0,1\}^{256}$. Similar to the "$\mathbf{G}_0 \to \mathbf{G}_1$" hop in our IND-CPA security proof of sntrup above (Subsection A.3), we can apply the double-sided OW2H lemma (Lemma 2.3) to construct a OW-CPA adversary $\mathcal{B}$ against $\mathsf{sntrup\text{-}core}$, whose running time is about the same as that of $\mathcal{A}$, such that

$$|\Pr[1 \leftarrow \mathbf{G}_0] - \Pr[1 \leftarrow \mathbf{G}_1]| \leq 2\sqrt{\mathsf{Adv}^{\textit{ow-cpa}}_{\mathsf{sntrup\text{-}core}}(\mathcal{B})}.$$

As in the proof of Theorem 4.3 above, we again crucially use the fact that $\mathsf{sntrup\text{-}core}$ is a perfectly-correct *deterministic* PKE scheme. □

Appendix B. IND-CCA Security of Streamlined NTRU Prime in the QROM

At a high level, the nested use of random oracles in Streamlined NTRU Prime, i.e., sntrup, is the main hurdle towards proving its IND-CCA security in the QROM. Here we provide a proof sketch by using so-called *quantum indifferentiability* to overcome this hurdle, following the idea proposed by Bernstein [41].

Theorem B.1 (informal)**.** *If* sntrup' *in Figure 7 is IND-CCA-secure in the QROM, then* sntrup *is also IND-CCA-secure in the QROM. If* $\mathsf{sntrup\text{-}core}$ *is OW-CPA-secure, then* sntrup *is also IND-CCA-secure in the QROM.*

Sketch. Our proof sketch follows a sequence of game hops. **Game 0**: This is the original IND-CCA game against sntrup. **Game 1**: We replace $K' \coloneqq \mathsf{H}_0(\mathsf{H}_3(s), ct)$ for invalid ciphertext with $K' \coloneqq \mathsf{H}_0(\bar{s}, ct)$, where $\bar{s} \leftarrow \{0,1\}^{256}$. This change is justified by the generalized OW2H lemma (Lemma 2.1).
Game 2: Next, we replace the computation of ct_1 and K by using two new random oracles $\bar{\mathsf{H}}_1, \bar{\mathsf{H}}_2 \colon \{0,1\}^* \to \{0,1\}^{256}$; we set $ct_1 = \bar{\mathsf{H}}_2(m, \mathsf{H}_4(ek))$ and $K' = \bar{\mathsf{H}}_1(m, ct)$ in the generation of the challenge and the decapsulation oracle. Now, the scheme is reduced to the simplified scheme sntrup' in Figure 7. This modification is justified by quantum indifferentiability as will be discussed later.

Finally, we note that this simplified variant employs one of the FO-like transforms $\mathsf{HU}^{\not\perp}$ described in [42] and can be considered as $\mathsf{HU}^{\not\perp}[\mathsf{sntrup\text{-}core}, \bar{\mathsf{H}}_1, \bar{\mathsf{H}}_2]$. Since the security of $\mathsf{HU}^{\not\perp}$ inherits that of the explicit-rejection variant $\mathsf{HU}^{\perp}$ [21], we only need to focus on security of $\mathsf{HU}^{\perp}[\mathsf{sntrup\text{-}core}, \bar{\mathsf{H}}_1, \bar{\mathsf{H}}_2]$ in the QROM. However the security of the latter is proven in [21], [43] following the OW-CPA security of the underlying PKE scheme, i.e., $\mathsf{sntrup\text{-}core}$. □

B.1. Quantum Indifferentiability

We now briefly discuss the concept of *quantum indifferentiability*. To do so, we generalize the corresponding results of Zhandry [44].

Namely in [44], Zhandry has shown the quantum indifferentiability of *domain extension* in hash functions as follows. Let $\mathsf{H}_0 \colon \mathcal{X}_0 \to \mathcal{Y}_0$ and $\mathsf{H}_1 \colon \mathcal{Y}_0 \times \mathcal{X} \to \mathcal{Y}$ be two random oracles. Define the construction $C^{\mathsf{H}_0, \mathsf{H}_1}(x_0, x) = \mathsf{H}_1(\mathsf{H}_0(x_0), x)$. Zhandry showed that this construction C is (quantum) indifferentiable from an independent "monolithic" random oracle.

But in our case, we would need to consider three random oracles H_0, H_1, and H_2 and two constructions $C_1^{\mathsf{H}_0, \mathsf{H}_1}(x_0, x) = \mathsf{H}_1(\mathsf{H}_0(x_0), x)$ and $C_2^{\mathsf{H}_0, \mathsf{H}_2}(x_0, x) = \mathsf{H}_2(\mathsf{H}_0(x_0), x)$ *simultaneously*.

It turns out that we can extend the simulator in Zhandry's original quantum indifferentiability proof for domain extension to our case in a straightforward manner. It is also easy to show that the simulator is *indistinguishable* [44, Lemma 8] and *consistent* [44, Lemma 13], and this leads to our constructions C_1 and C_2 being indifferentiable from the independent random oracles $\bar{\mathsf{H}}_1, \bar{\mathsf{H}}_2$ in our above proof sketch. That is, there exists a QPT simulator S such that for any QPT adversary D,

$$\left| \begin{array}{l} \Pr[D^{|\vec{\mathsf{H}}\rangle, |C_1^{\mathsf{H}_0, \mathsf{H}_1}\rangle, |C_2^{\mathsf{H}_0, \mathsf{H}_2}\rangle}(1^\lambda) = 1] \\ \quad - \Pr[D^{|S^{\bar{\mathsf{H}}_1, \bar{\mathsf{H}}_2}\rangle, |\bar{\mathsf{H}}_1\rangle, |\bar{\mathsf{H}}_2\rangle}(1^\lambda) = 1] \end{array} \right| = \mathsf{negl}(\lambda).$$

Thus, this simulator justifies our game hop from Game 1 to Game 2 in the above sketch.