Security analysis of the iMessage PQ3 protocol*

Douglas Stebila
University of Waterloo

February 27, 2024

Abstract

The iMessage PQ3 protocol is an end-to-end encrypted messaging protocol designed for exchanging data in long-lived sessions between two devices. It aims to provide classical and post-quantum confidentiality for forward secrecy and post-compromise secrecy, as well as classical authentication. Its initial authenticated key exchange is constructed from digital signatures plus elliptic curve Diffie–Hellman and post-quantum key exchanges; to derive per-message keys on an ongoing basis, it employs an adaptation of the Signal double ratchet that includes a post-quantum key encapsulation mechanism. This paper presents the cryptographic details of the PQ3 protocol and gives a reductionist security analysis by adapting the multi-stage key exchange security analysis of Signal by Cohn-Gordon et al. (J. Cryptology, 2020). The analysis shows that PQ3 provides confidentiality with forward secrecy and post-compromise security against both classical and quantum adversaries, in both the initial key exchange as well as the continuous rekeying phase of the protocol.

*This paper was supported by Apple Inc.

Contents

1 Introduction

Secure messaging protocols. The last decade has seen a rapid advance in the security of messaging protocols. Early instant messaging protocols often had little to no security: some systems employed encryption between a user's device and the service provider's server, but this still allowed the service provider to see plaintext of users' messages as they were relayed from one user to another. PGP-encrypted email [Zim95] was one possible way for users to obtain end-to-end encrypted communication, but the usability challenges of PGP meant it was only accessible to advanced users. There was a clear need for comprehensive security with end-to-end encryption to protect instant messaging from passive and active attackers, including a potentially malicious service provider.

The Off-the-Record Messaging Protocol (OTR) [BGB04] was one of the first protocols that provided end-to-end security for 2-party instant messaging, and was primarily used via a plug-in or add-on to existing instant messaging clients. OTR recognized that the security needs of chat sessions might be different from traditional key exchange protocols, since chat sessions may be long-lived, and user devices might be compromised at some point over the potentially long lifetime of the chat session. Consequently, the OTR protocol did a new Diffie–Hellman key exchange with each round-trip of chat messages to derive new message encryption keys; this technique, now called *asymmetric ratcheting*, made it hard for an adversary who compromised a device at a particular point in time to re-compute earlier message encryption keys (assuming they were deleted from memory) and also later message encryption keys (assuming the adversary's intrusion is only temporary).

While OTR did not see widespread adoption, it influenced the design of the Signal protocol [Sig16], by Marlinspike and Perrin, which combined an implicitly authenticated key exchange protocol with the ratcheting technique to achieve end-to-end encrypted security messaging with multiple security properties, including authenticity, deniability, and confidentiality with both forward secrecy and post-compromise secrecy. An important characteristic of the Signal protocol is that it achieves end-to-end encryption without sacrificing asynchronicity: messages can be sent from a sender to a recipient (via a relay server) at any point in time, without the recipient needing to be online to complete the exchange. Being end-to-end encrypted, the relay server cannot see the plaintext of the users' messages (though it may be able to see metadata). The Signal protocol has been widely adopted in the namesake Signal messenger, as well as many other applications and products, notably including WhatsApp.

The above protocols focused primarily on 2-party end-to-end encrypted messaging; building protocols for encrypted group messaging is challenging, in part due to the difficulties of efficiently managing and updating group state in an asynchronous manner. The Messaging Layer Security (MLS) protocol [BBR+23] is an initiative of the Internet Engineering Task Force to build an open standard for secure group messaging.

Post-quantum security. Another major trend in the past decade has been the initiation of the transition to quantum-resistant cryptography.

In 1994, Shor [Sho94] published a quantum algorithm that could efficiently solve the factoring and discrete logarithm problems using a sufficiently large quantum computer, which would then break the public key cryptography used in all major secure communication protocols. Although it is not yet possible to build a sufficiently large quantum computer to run Shor's algorithm on cryptographic challenges of the size used in communication protocols, the state of the art continues to improve in quantum computing research. Furthermore, some security goals are threatened even if quantum computers do not yet exist: an eavesdropper could record communications today, store it, and then break it when they have a quantum computer available in the future. This "store now, decrypt later" attack would undermine data intended to have long-term security.

A variety of cryptographic primitives have been proposed that base their hardness on mathematical problems not solved by Shor's factoring algorithm. The endeavour to build *quantum-resistant cryptography*, also called *post-quantum cryptography* (PQC), was jump-started in 2015 with the announcement by the United States National Institute of Standards and Technology (NIST) of their post-quantum cryptography standardization project, with the goal of standardizing quantum-resistant digital signature schemes and public key encryption or key encapsulation mechanisms (KEMs). After a multi-year process involving several rounds of review, in 2022 NIST announced the selection of 4 post-quantum algorithms for standardization: the

key encapsulation mechanism CRYSTALS-Kyber [SAB+22], and 3 digital signature schemes (CRYSTALS-Dilithium [LDK+22], Falcon [PFH+22], and SPHINCS+ [HBD+22]).

In parallel, academia and industry have begun updating and redesigning communication protocols to incorporate post-quantum cryptography. There are several factors that mean the transition to post-quantum cryptography is non-trivial, even after new algorithms are standardized. One factor is that the available post-quantum algorithms generally have larger communication sizes compared to traditional algorithms: for example, post-quantum algorithm CRYSTALS-Kyber requires the exchange of 2432 bytes to establish a shared secret, whereas elliptic curve Diffie–Hellman key exchange can do so with just 64 bytes of communication. Additionally, KEMs are not always a drop-in replacement for Diffie–Hellman key exchange as they have a slightly different communication pattern.

Consequently, new or updated designs will be necessary in many cases to upgrade existing communication protocols to have post-quantum security. Furthermore, many adopters are choosing to deploy post-quantum cryptography in a so-called *hybrid mode*, which uses both an existing classical algorithm (e.g., elliptic curve Diffie–Hellman) and a post-quantum algorithm (e.g., Kyber key exchange) together to achieve security as long as either algorithm remains unbroken. The hybrid approach can reduce risk in depending on newer cryptographic assumptions while still providing the potential of post-quantum security, with relatively small additional cost, and is being considered for a variety of protocols, including TLS [SFG23] and SSH [KSH23].

Goals of the iMessage PQ3 protocol. The iMessage PQ3 protocol [App24] is an end-to-end encrypted messaging protocol designed for exchanging data in long-lived sessions between two devices, which aims to provide hybrid classical and post-quantum security.

The main goals of the PQ3 protocol are as follows:

- *Confidentiality* should be provided for application data between the sender and the receiver, including against eventually-quantum attackers carrying out "store now, decrypt later" attacks.
- *Authentication* should allow a receiver to identify the sender of a message.
- *Forward secrecy* of session keys: If a party's state (including its long-term key) is compromised at one point in time, ciphertexts previously transmitted cannot be decrypted (assuming the corresponding messages were deleted from the compromised device).
- Forward secrecy should be available on a per-message basis.
- *Post-compromise security* of session keys, also known as *healing*: If a party's device is compromised at one point in time, but subsequently secured again (for example, because malware was detected and removed), ciphertexts transmitted after the device was healed cannot be decrypted. Post-compromise security should be available on a per-round-trip basis; post-quantum post-compromise security may be amortized across several round-trips if the bandwidth cost of larger post-quantum messages is too high to do with every round-trip.
- *Cryptographic replay protection* should be able to detect and discard replayed messages.
- *Asynchronous messaging*: messages can be sent and received at any time via a relay server without requiring online interaction between the sender and receiver.

Because of the intended application scenario for the PQ3 protocol, it does not aim to address group messaging, authentication against quantum adversaries, or cryptographic deniability.

The PQ3 protocol should be secure against passive or active classical adversaries, as well as passive quantum adversaries.

Some, but not all, of the goals for PQ3 are similar to other end-to-end encrypted messaging protocols such as the Signal protocol [Sig16], so some design elements and analysis techniques from Signal are helpful in understanding PQ3. We begin by reviewing the Signal protocol, which was designed by Marlinspike and Perrin, building on earlier end-to-end encrypted message protocols such as Off-the-Record Messaging (OTR) [BGB04].

The Signal protocol. The Signal protocol involves several cryptographic components which work together to achieve multiple security goals. The main phases of the Signal protocol are as follows:

- *Registration*: Each user generates a long-term identity key pair as well as several additional key pairs, and uploads the public keys to a key server; this is called a "pre-key bundle".
- *Initial key exchange* also known as the *root key establishment* or *session start*: The initiator of a session retrieves the recipient's pre-key bundle from the key server, and performs multiple Diffie–Hellman key exchanges depending on long-term, medium-term, and ephemeral keys to produce a shared secret "root key". In Signal this is called the X3DH handshake [MP16].
- *Asymmetric ratchet*: In each round trip between the two users, new ephemeral Diffie–Hellman public keys are exchanged to generate new DH shared secrets, which are hashed together with the previous root key to generate a new root key.
- *Symmetric ratchet*: From each root key, a key derivation function is used to generate a chain of keys which can be used for symmetric authenticated encryption.

In Signal, the asymmetric and symmetric ratchet together are called the double ratchet [PM16].

Security of Signal. The Signal protocol aims to achieve multiple security goals.

- *Entity authentication*: Parties are mutually authenticated using implicitly authenticated key exchange in the X3DH handshake. Note that the initial delivery of long-term public keys is done by a potentially adversarial key server, but the application does provide a mechanism for parties to compare the received keys by scanning QR codes in an out-of-band ceremony.
- *Deniability*: Authentication is derived from implicitly authenticated key exchange rather than digital signatures, so no artifacts are generated that can be verified by a judge, providing a form of offline deniability [DGK06, VGIK20].
- *Forward secrecy* of session keys.
- *Post-compromise security* of session keys.

The main building blocks in the Signal protocol are elliptic curve Diffie–Hellman key exchange and the HKDF key derivation function [Kra10].

The first security analysis of the Signal protocol was given by Cohn-Gordon et al. [CGCD$^+$17, CCD$^+$20]. They adapted the multi-stage key exchange security model of Fischlin and Günther [FG14] to handle the various stages and security properties of the Signal protocol, including forward secrecy and post-compromise security. They gave a reductionist proof of the security of Signal in this multi-stage key exchange model, assuming either the gap-Diffie–Hellman assumption in the random oracle model for HKDF, or the PRF-ODH assumption [BFGJ17] plus PRF security of HKDF.

The iMessage PQ3 protocol. The PQ3 protocol has four main phases: registration, initial key exchange, asymmetric ratchet, and symmetric ratchet. Compared with the Signal protocol, it shares some design elements, including the basic idea for the double ratchet, but uses different designs for the registration and initial key exchange, and adds post-quantum components to the asymmetric ratchet. The PQ3 protocol does not aim to provide deniability, so it uses digital signatures for authentication in the initial key exchange, rather than implicitly authenticated key exchange.

Additionally, the PQ3 protocol aims to provide post-quantum confidentiality, so it adds post-quantum key exchange in the initial key exchange and asymmetric ratchet using the ML-KEM [Nat23] key encapsulation mechanisms (KEM); the post-quantum KEMs are used alongside Diffie–Hellman key exchange in a hybrid mode, aiming to achieve security as long as either the Diffie–Hellman or post-quantum assumption holds. Because deniability is not a security goal, PQ3 is able to rely on digital signatures for authentication, rather than needing to use implicit authentication as in Signal's X3DH protocol. Furthermore, DH is a non-interactive key exchange whereas KEMs are not, which in practice means that KEM ciphertexts cannot

be used as KEM public keys. Hence, whereas the DH portion of the asymmetric ratchet can use a DH public key in two consecutive ratchet steps, the post-quantum KEM-based portion of the asymmetric ratchet must use independent KEM public keys and ciphertexts, so twice as many values need to be transmitted compared to the DH-based asymmetric ratchet. Finally, because the public keys and ciphertexts in ML-KEM are much larger than ECDH public keys, PQ3 implementations provide the option for only periodically doing a post-quantum key exchange, rather than on every round trip.

Security of PQ3. In this paper, we show that PQ3 provides authenticated key exchange and establishes secure session keys, satisfying forward secrecy and post-compromise security. The security definition is an adaptation of the multi-stage authenticated key exchange security model used by Cohn-Gordon et al. [CGCD$^+$17] to prove security of the Signal protocol. In particular, the model covers the authentication and confidentiality (with forward secrecy and post-compromise security) goals as stated above, against passive or active classical adversaries, as well as passive quantum adversaries.

Review of secure messaging literature. Two essential properties of modern secure messaging protocols are forward secrecy and post-compromise security. Forward secrecy has long been known in the authenticated key exchange literature [Gün90], but post-compromise security, also called healing, emerged more recently with secure messaging protocols that use ratcheting, starting with OTR [BGB04] and Signal [Sig16]. Post-compromise security was first formalized by Cohn-Gordon et al. [CCG16] and incorporated into the first provable security analysis of Signal [CGCD$^+$17], and has emerged as an important goal for many protocols [BBL$^+$23]. Ratcheted key exchange—characterized by both forward secrecy and post-compromise security—has continued to be studied sometimes under the name continuous key agreement [MCYR17, PR18, ACD19, DV19, JMM19, ACJM20, DHRR22, DH23]. Secure messaging in general and the double ratchet in particular have also been analyzed in the universal composability paradigm [CJSV22, BFG$^+$22a].

There have been several works in the literature aiming to provide post-quantum secure messaging protocols. Alwen et al. [ACD19] showed how to generalize the double ratchet construction to use KEMs, which could then provide post-quantum security with an appropriate KEM. Several papers have focused on adapting the Signal X3DH handshake to provide post-quantum security [BFG$^+$20, HKKP21, BFG$^+$22b, DG22] while trying to preserve the offline deniability feature of the Signal handshake, but this has been challenging since most post-quantum KEMs do not fit the message flow of non-interactive key exchange (NIKE) [FHKP13]. In 2023, Signal [KS23] released the PQXDH handshake (supported by a formal analysis using both CryptoVerif and ProVerif tools [BJK23]) to replace the X3DH handshake in the Signal protocol, which hybridizes the X3DH handshake with an additional post-quantum component (using the Kyber KEM), but using post-quantum only for ephemeral key exchange in the handshake, not identity key exchange, which sidesteps the challenges encountered in the aforementioned academic attempts at a post-quantum Signal-like handshake.

Outline. In Section 2, we present the cryptographic building blocks used in the iMessage PQ3 protocol. Section 3 gives the details of the PQ3 protocol. In Section 4, we define the security model used for the cryptographic analysis of the PQ3 protocol, and Section 5 gives the proof of security of PQ3 in this model.

2 Preliminaries and notation

In this section, we review the cryptographic building blocks used in the PQ3 protocol and their security definitions, as well as introduce some notation for key exchange protocols.

Notation. If A is a deterministic algorithm, then $y \leftarrow A(x)$ denotes running A with input x and assigning the output to variable y. If A is a probabilistic algorithm, then $y \leftarrow\!\!\$\ A(x)$ denotes running A with input x and fresh random coins, and assigning the output to variable y. $A^O(x)$ denotes running algorithm A oracle access to oracle O.

2.1 Cryptographic building blocks

The iMessage PQ3 protocol uses several cryptographic building blocks, including a digital signature scheme, key encapsulation mechanisms, pseudorandom functions, and Diffie–Hellman key exchange.

Definition 1 (Digital signature scheme)**.** A *digital signature scheme* Σ is a tuple of algorithms:

- $\Sigma.\mathsf{KGen}() \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$: A probabilistic key generation algorithm that produces a public key pk and a private key sk.
- $\Sigma.\mathsf{Sig}(\mathsf{sk}, m) \xrightarrow{\$} \sigma$: A possibility probabilistic signature generation algorithm that takes as input a private key sk and message m, and produces a signature σ.
- $\Sigma.\mathsf{Vf}(\mathsf{pk}, m, \sigma) \to \{0, 1\}$: A deterministic signature verification algorithm that takes as input a public key pk, message m, and signature σ, and produces as output a bit representing accept (1) or reject (0).

In PQ3, Σ is the ECDSA signature scheme using the NIST P-256 elliptic curve.

Definition 2 (Existential unforgeability under chosen message attack)**.** A digital signature scheme Σ is said to be *existentially unforgeable under chosen message attack* if it is computationally infeasible for an adversary to produce a new message-signature pair that verifies, even given access to a signing oracle. More precisely, define

$$\mathsf{Adv}^{\mathrm{euf-cma}}_{\Sigma}(\mathcal{A}) = \Pr\left[\text{EUF-CMA}^{\mathcal{A}}_{\Sigma} \Rightarrow 1\right]$$

where $\text{EUF-CMA}^{\mathcal{A}}_{\Sigma}$ is the security experiment shown in Figure 1.

$\text{EUF-CMA}^{\mathcal{A}}_{\Sigma}$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \Sigma.\mathsf{KGen}()$
2 : $M \leftarrow \emptyset$
3 : $(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{OSig}}(\mathsf{pk})$
4 : **if** $\Sigma.\mathsf{Vf}(\mathsf{pk}, m^*, \sigma^*) \wedge m^* \notin M$ **then return** 1
5 : **else return** 0

$\mathsf{OSig}(m)$

1 : $M \leftarrow M \cup \{m\}$
2 : **return** $\Sigma.\mathsf{Sig}(\mathsf{sk}, m)$

Figure 1: Security experiment for existential unforgeability under chosen message attack for a signature scheme Σ.

Definition 3 (Key encapsulation mechanism)**.** A *key encapsulation mechanism* Γ is a tuple of the following algorithms and a shared secret space $\mathcal{K}$:

- $\Gamma.\mathsf{KGen}() \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$: A probabilistic key generation algorithm that produces a public key pk and a private key sk.
- $\Gamma.\mathsf{Enc}(\mathsf{pk}) \xrightarrow{\$} (\mathsf{ct}, \mathsf{ss})$: A probabilistic encapsulation algorithm that takes as input a public key pk and produces as output a ciphertext (also called an encapsulation) ct and a shared secret ss. Let $\Gamma.\mathcal{R}$ denote the space of randomness used by probabilistic algorithm $\Gamma.\mathsf{Enc}$. Note that the probabilistic formulation $(\mathsf{ct}, \mathsf{ss}) \leftarrow\!\!\$\ \Gamma.\mathsf{Enc}(\mathsf{pk})$ is equivalent to separately selecting the randomness and then calling the derandomized version: $r \leftarrow\!\!\$\ \Gamma.\mathcal{R}; (\mathsf{ct}, \mathsf{ss}) \leftarrow \Gamma.\mathsf{Enc}(\mathsf{pk}; r)$.
- $\Gamma.\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{ss}$: A deterministic decapsulation algorithm that takes as input a secret key sk and a ciphertext ct and produces as output a shared secret ss.

In PQ3, two key encapsulation mechanisms are used: Π_1, which is ML-KEM-1024-ipd ('ipd' standards for 'Initial Public Draft' [Nat23]), is used in the initial key exchange; and Π_2, which is ML-KEM-768-ipd, is used in the asymmetric ratchet.

Definition 4 (Indistinguishability under chosen ciphertext attack)**.** A KEM Γ is said to be *indistinguishable under chosen ciphertext attack* if it is computationally infeasible for an adversary to distinguish the shared secret of a challenge ciphertext from a uniformly random shared secret, even given access to a decapsulation oracle. More precisely, define

$$\mathsf{Adv}^{\mathrm{ind-cca}}_{\Gamma}(\mathcal{A}) = \left| \Pr\left[\text{IND-CCA}^{\mathcal{A}}_{\Gamma} \Rightarrow 1\right] - \frac{1}{2} \right|$$

where $\text{IND-CCA}^{\mathcal{A}}_{\Gamma}$ is the security experiment shown in Figure 2.

$\text{IND-CCA}^{\mathcal{A}}_{\Gamma}$	$\mathsf{ODec}(c)$
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \Gamma.\mathsf{KGen}()$	1 : **if** $c = c^*$ **then return** $\perp$
2 : $(c^*, ss_0^*) \leftarrow\!\!\$\ \Gamma.\mathsf{Enc}(\mathsf{pk})$	2 : **else return** $\Gamma.\mathsf{Dec}(\mathsf{sk}, c)$
3 : $ss_1^* \leftarrow\!\!\$\ \Gamma.\mathcal{K}$	
4 : $b \leftarrow\!\!\$\ \{0,1\}$	
5 : $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ODec}}(\mathsf{pk})$	
6 : **if** $b = b'$ **then return** 1	
7 : **else return** 0	

Figure 2: Security experiment for indistinguishability under chosen ciphertext attack for a key encapsulation mechanism Γ.

Definition 5 (Pseudorandom function)**.** A *pseudorandom function* $F : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^\lambda$ takes as input a key $k \in \mathcal{K}$ and a string $label \in \{0,1\}^*$, and produces as output a key $F(k, label) \in \{0,1\}^\lambda$.

In PQ3, several pseudorandom functions are used based on HKDF [Kra10]. The $\mathsf{KDFRKCK}$ construction based on HKDF that is used in the PQ3 protocol is defined in Section 3.5.2.

Definition 6 (HKDF)**.** HKDF [Kra10] consists of two algorithms:

- $\mathsf{HKDF.Extract}(ikm, salt) \to ext$: A deterministic extraction algorithm that takes as input initial keying material ikm and salt $salt$, and produces as output an extract key ext.
- $\mathsf{HKDF.Expand}(k, label, \ell) \to k'$: A deterministic expansion algorithm that takes as input a key k (typically the output of $\mathsf{HKDF.Extract}$), a label $label \in \{0,1\}^*$, and a length $\ell > 0$, and produces a binary string k' of length ℓ.

$\mathsf{HKDF.Extract}$ and $\mathsf{HKDF.Expand}$ are both based on HMAC, and are instantiated using a cryptographic hash function. In PQ3, the hash function used is SHA-384.

Definition 7 (Secure pseudorandom function)**.** A pseudorandom function F is said to be *secure* if it is computationally infeasible for an adversary to distinguish the output of a pseudorandom function (under an unknown key) from the output of a random function. More precisely, let $F : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^\lambda$ and define

$$\mathsf{Adv}^{\mathrm{prf}}_{F}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{F(k,\cdot)}() \Rightarrow 1 | k \leftarrow\!\!\$\ \mathcal{K}\right] - \Pr\left[\mathcal{A}^{R(\cdot)}() \Rightarrow 1 | R \leftarrow\!\!\$\ \{\text{all functions } : \{0,1\}^* \to \{0,1\}^\lambda\}\right] \right|$$

In PQ3, we sometimes require that a multi-argument function be a secure pseudorandom function in a specific input. For example, suppose f is an n-input function; we may require that f is a pseudorandom function in its ith input. Formally, this can be done by defining

$$f'(x_j, (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)) = f(x_1, \ldots, x_n)$$

where $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$ is unambiguously encoded in $\{0,1\}^*$, and requiring that f' is a secure pseudorandom function. We use the notation

$$\mathsf{Adv}^{\mathrm{prf_j}}_{f}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{prf}}_{f'}(\mathcal{A})$$

Definition 8 (Diffie–Hellman key exchange)**.** Let g be the generator of a group G of prime order q. *Diffie–Hellman key exchange* is a two party protocol in which Alice generates a private key $a \leftarrow\!\!\$\ \mathbb{Z}_q$ and corresponding public key $A \leftarrow g^a$, and Bob generates a private key $b \leftarrow\!\!\$\ \mathbb{Z}_q$ and corresponding public key $B \leftarrow g^b$. They exchange public keys A and B, and compute shared secret $g^{ab} = A^b = B^a$.

In PQ3, G is the NIST P-256 elliptic curve.

Definition 9 (PRF-ODH)**.** Let g be the generator of a group G of prime order q, and let $F : G \times \{0,1\}^* \to \{0,1\}^\lambda$ be a pseudorandom function. The *PRF-ODH assumption* for F and g holds if it is computationally infeasible for an adversary to distinguish from random the output of F on a Diffie–Hellman shared secret and an adversary-chosen label, even given the ability to get F evaluated on values of the adversary's choice. More precisely, define

$$\mathsf{Adv}^{\mathrm{prf-odh}}_{g,F}(\mathcal{A}) = \left| \Pr\left[\mathrm{PRFODH}^{\mathcal{A}}_{g,F} \Rightarrow 1\right] - \frac{1}{2} \right|$$

where $\mathrm{PRFODH}^{\mathcal{A}}_{g,f}$ is the security experiment shown in Figure 3.

Note that we use the $\mathsf{mmPRF-ODH}$ variant from [BFGJ17].

$\mathrm{PRFODH}^{\mathcal{A}}_{g,F}$

1 : $u, v \leftarrow\!\!\$\ \mathbb{Z}_q$
2 : $L \leftarrow \emptyset$
3 : $(x^*, st) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{U},\mathsf{V}}(g^u, g^v)$
4 : $y_0 \leftarrow F(g^{uv}, x^*)$
5 : $y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$
6 : $b \leftarrow\!\!\$\ \{0,1\}$
7 : $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{U},\mathsf{V}}(st, y_b)$
8 : **if** $(g^{uv}, x^*) \in L$ **then**
9 : $\quad z \leftarrow\!\!\$\ \{0,1\}$
10 : $\quad$ **return** z
11 : **if** $b = b'$ **then return** 1
12 : **else return** 0

$\mathsf{U}(W, x)$

1 : $L \leftarrow L \cup \{(W^u, x)\}$
2 : **return** $F(W^u, x)$

$\mathsf{V}(W, x)$

1 : $L \leftarrow L \cup \{(W^v, x)\}$
2 : **return** $F(W^v, x)$

Figure 3: Security experiment for the PRF-ODH assumption for group with generator g and pseudorandom function F.

2.2 Key exchange protocol notation

Definition 10 (Multi-stage key exchange protocol)**.** A *multi-stage key exchange protocol* Π is a tuple of algorithms along with a keyspace $\mathcal{K}$. The algorithms are:

1. $\mathrm{IDKeyGen}() \xrightarrow{\$} (\mathsf{idecvk}_P, \mathsf{idecsk}_P)$: A probabilistic *long-term identity key generation algorithm* that outputs a long-term public key / secret key pair.

2. $\mathrm{PKBGen}(\mathsf{idecsk}_P) \xrightarrow{\$} (\mathsf{preecpk}_{P,i}, \mathsf{preecsk}_{P,i}, \mathsf{prepqpk}_{P,i}, \mathsf{prepqsk}_{P,i}, \sigma)$: A probabilistic *pre-key bundle generation algorithm* that takes as input a long-term identity secret key and outputs an ephemeral elliptic curve pre-key key pair, an ephemeral post-quantum KEM pre-key key pair, and a signature.

3. $\mathrm{SessionStartS}(\pi, \mathsf{idecvk}_R, \mathsf{PKB}_{R,i}) \xrightarrow{\$} (\pi', msg')$: A probabilistic *sender session start algorithm* that takes as input a session state π, the receiver's identity public key idecvk_R, and the public parts of pre-key bundle $\mathsf{PKB}_{R,i}$, and updates the session state π' as well as producing an outgoing protocol message, which in PQ3 contains an EC ratchet public key $\mathsf{rchecpk}$, PQ pre-key ciphertext $\mathsf{prepqct}$, and PQ ratchet public key $\mathsf{rchpqpk}$, a hash value, and a signature σ.

4. SessionStartR($\pi, \mathsf{idecvk}_S, \mathsf{PKB}_{R,i}, msg$) $\xrightarrow{\$}$ (π', msg'): A probabilistic *receiver session start algorithm* that takes as input a session state π, the sender's identity public key idecvk_S, the public and private parts of pre-key bundle $\mathsf{PKB}_{R,i}$, and a protocol message msg which in PQ3 contains asymmetric ratchet public keys and ciphertexts and check values, and updates the session state π' as well as producing a protocol message which in PQ3 contain asymmetric ratchet public keys and ciphertexts, namely an outgoing EC ratchet public key $\mathsf{rchecpk}'$, PQ ratchet ciphertext $\mathsf{rchpqct}'$, and PQ ratchet public key $\mathsf{rchpqpk}'$, as well as a signature.

5. AsymRatchet(π, msg) $\xrightarrow{\$}$ (π', msg'): A probabilistic *asymmetric ratchet algorithm* that takes as input a session state π and a protocol message msg which in PQ3 contains asymmetric ratchet public keys and ciphertexts and a signature, and updates the session state π' as well as produces a protocol message which in PQ3 contains asymmetric ratchet public keys and ciphertexts and a signature.

6. SymRatchet($\pi, chain, j$) $\rightarrow \pi$ where $chain = \mathsf{in}$ or out: A *symmetric ratchet algorithm* that takes as input a session state π and the index of a symmetric chain $chain[j]$, and updates the chain state.

Let P be a party. We denote P's pth session by $\pi_{P,p}$. A *session* is an execution of the protocol at a party, and represents one party's execution in a single long-lived chat. In a long-lived chat between two parties Alice and Bob, there will be two sessions: one at Alice, one at Bob.

Within each session, there will be many *stages*, each of which may establish a shared secret. In Signal and PQ3, the stages within a session can be viewed as being organized into a "tree" of stages. The main 'trunk' of the tree consists of the initial session start / handshake stage, and those created by the asymmetric ratchet. Stages on the trunk are denoted by $[\mathtt{asym}_0], [\mathtt{asym}_1], [\mathtt{asym}_2], \ldots$. Off of each trunk stage $[\mathtt{asym}_j]$ there are two branches: the incoming chain $\mathsf{in}[j]$ and the outgoing chain $\mathsf{out}[j]$. These two chains are advanced using the symmetric ratchet, and the stages on these chains are $[\mathtt{in}_{j,0}], [\mathtt{in}_{j,1}], \ldots$, and similarly for $[\mathtt{out}_{j,\ell}]$. (In PQ3, off of each trunk stage, only one of the incoming or outgoing chains will be populated, and the other will be null, as the PQ3 asymmetric ratchet is stepped forward to create alternating incoming and outgoing chains.) Implicitly each chain stores its current length, which is denoted $|[chain]|$.

Definition 11 (Session state)**.** Each party, for each session, maintains a *session state* π which is a collection of the following variables:

- $\pi.\mathsf{role} \in \{\mathtt{init}, \mathtt{resp}\}$: the instance's role
- $\pi.\mathsf{peerid}$: the identifier of the alleged peer
- $\pi.\mathsf{peeripk}$: the peer's long-term identity public key
- $\pi.\mathsf{peerprepk}$: the peer's EC and PQ pre-key public keys
- $\pi.\mathsf{status}[s] \in \{\mathtt{active}, \mathtt{accept}, \mathtt{reject}\}$: execution status for stage s, set to `active` upon start of a new stage, and set to `accept` or `reject` to indicate acceptance or rejection of a particular stage and its key
- $\pi.k[s] \in \mathcal{K}$: the session key for stage s; can be a root key rk, a chain key ck, or a message key mk
- $\pi.\mathsf{sid}[s]$: the identifier of stage s of session π
- $\pi.st[s]$: protocol-specific state for stage s of session π

3 iMessage PQ3 Protocol description

In this section we present the iMessage PQ3 protocol. As previously explained, the main components of the protocol are: *user registration* (Section 3.1) in which each party generates a long-term key pair as well as many pre-key bundles, and uploads them to the server; *session start* (Section 3.2), which establishes the initial shared secret between the initiator and the responder; the *asymmetric ratchet* (Section 3.3), which establishes new shared secrets after each round trip; and the *symmetric ratchet* (Section 3.4), which derives keys for each message.

Table 1 shows the various public key material used by parties in the session start and asymmetric ratchet, and the symmetric keys produced at the various stages.

Asymmetric key pairs		
idecvk_P	idecsk_P	Party P long-term (identity) EC key pair for signature scheme Σ
$\mathsf{preecpk}_{P,i}$	$\mathsf{preecsk}_{P,i}$	Party P prekey bundle i prekey EC key pair
$\mathsf{prepqpk}_{P,i}$	$\mathsf{prepqsk}_{P,i}$	Party P prekey bundle i prekey PQ key pair
$\mathsf{prepqct}_{P,p}$		Party P session p prekey PQ ciphertext
$\mathsf{rchecpk}_{P,p,i}$	$\mathsf{rchecsk}_{P,p,i}$	Party P session p asymmetric stage i ratchet EC key pair
$\mathsf{rchpqpk}_{P,p,i}$	$\mathsf{rchpqsk}_{P,p,i}$	Party P session p asymmetric stage i ratchet PQ key pair
$\mathsf{rchpqct}_{P,p,i}$		Party P session p asymmetric stage i ratchet PQ ciphertext
Symmetric keys		
$\pi_{P,p}.rk[\mathtt{asym}_i]$		Party P session p asymmetric stage i root key
$\pi_{P,p}.ck[\mathtt{in}_{i,j}]$		Party P session p incoming chain i symmetric stage j chain key
$\pi_{P,p}.mk[\mathtt{in}_{i,j}]$		Party P session p incoming chain i symmetric stage j message key
$\pi_{P,p}.ck[\mathtt{out}_{i,j}]$		Party P session p outgoing chain i symmetric stage j chain key
$\pi_{P,p}.mk[\mathtt{out}_{i,j}]$		Party P session p outgoing chain i symmetric stage j message key

Table 1: Keys used in the PQ3 protocol.

3.1 User registration

The user registration phase is run by each party P when they first set up their device. Associated to each party P is an identity string id_P.

Identity keypair. Each party P prepares a long-term identity key pair consisting of (the public components of):

- $(\mathsf{idecvk}_P, \mathsf{idecsk}_P) \leftarrow\!\!\$\ \Sigma.\mathsf{KGen}()$

Pre-key bundles. Each party P also prepares many pre-key bundles $\mathsf{PKB}_{P,i}$ for $i = 1, \ldots,$ each consisting of (the public components of):

- $\mathsf{preecsk}_{P,i} \leftarrow\!\!\$\ \mathbb{Z}_q, \mathsf{preecpk}_{P,i} \leftarrow g^{\mathsf{preecsk}_{P,i}}$
- $(\mathsf{prepqpk}_{P,i}, \mathsf{prepqsk}_{P,i}) \leftarrow\!\!\$\ \Pi_1.\mathsf{KGen}()$
- $\sigma \leftarrow\!\!\$\ \Sigma.\mathsf{Sig}(\mathsf{idecsk}_P, \mathtt{label}_{\text{registration}} \| \mathsf{prepqpk}_{P,i} \| \mathsf{preecpk}_{P,i} \| \mathtt{label}_{\text{version}})$

At registration time, party P uploads its first pre-key bundle to the server, and then uploads subsequent pre-key bundles periodically at later times. The party saves the corresponding secret keys locally. A pre-key bundle is meant to be used for a particular period of time, and as such includes a timestamp to prevent the server from replaying a pre-key bundle at a later period of time outside the validity period. As such, we model pre-keys as "medium-term" keys which can be reused in many sessions.

3.2 Session start (initial key establishment)

Suppose a sender S wants to start a session with a recipient R. The s'th session at S will be denoted by $\pi_{S,s}$.

We assume the sender has a copy of the receiver's identity public key idecvk_R obtained authentically through some out-of-band mechanism, and similarly the responder R has an authentic copy of the sender's identity public key idecvk_S. (If these are obtained from the untrusted server, then the security properties shown in this paper apply only if the server does not replace / alter user's identity public keys, or if the users authenticate each others' identity public keys using some out-of-band mechanism, such as comparing a hash of the identity public keys [App23].)

The sender S obtains a fresh pre-key bundle $\mathsf{PKB}_{R,i}$ for recipient R from the server. The sender then proceeds as in Figure 4. By the end of SessionStartS, the initiator has set the root key for stage $[\mathtt{asym}_0]$ and the chain key for outgoing chain $[\mathtt{out}_{0,0}]$, and produced an outgoing message for the responder.

SessionStartS($\pi_{S,s}, \mathsf{idecvk}_R, \mathsf{PKB}_{R,i}$)

// Verify signature on prekey bundle
$\Sigma.\mathsf{Vf}(\mathsf{idecvk}_R, \mathtt{label}_{\text{registration}} \| \mathsf{prepqpk}_{R,i} \| \mathsf{preecpk}_{R,i} \| \mathtt{label}_{\text{version}}, \mathsf{PKB}_{R,i}.\sigma)$
// Generate sender ratchet ECDH key pair
$\mathsf{rchecsk}_{S,s,1} \leftarrow\!\!\$\ \mathbb{Z}_q, \mathsf{rchecpk}_{S,s,1} \leftarrow g^{\mathsf{rchecsk}_{S,s,1}}$
// Compute ratchet/prekey ECDH shared secret
$\mathsf{ecss} \leftarrow \mathsf{preecpk}_{R,i}^{\mathsf{rchecsk}_{S,s,1}}$
// Encapsulate against receiver PQ prekey
$\mathsf{prepqrnd}_{S,s} \leftarrow\!\!\$\ \Pi_1.\mathcal{R}$
$(\mathsf{prepqct}_{S,s}, \mathsf{pqss}) \leftarrow \Pi_1.\mathsf{Enc}(\mathsf{prepqpk}_{R,i}; \mathsf{prepqrnd}_{S,s})$
// Derive session identifier and root/chain keys
$\pi_{S,s}.\mathsf{sid}[\mathtt{asym}_0] \leftarrow \mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{start}} \| \mathsf{preecpk}_{R,i} \| \mathsf{rchecpk}_{S,s,1} \| \mathsf{prepqct}_{S,s} \| \mathsf{prepqpk}_{R,i}$
$(rk, ck) \leftarrow \mathsf{KDFRKCK}(0, \mathsf{ecss}, \mathsf{pqss}, \pi_{S,s}.\mathsf{sid}[\mathtt{asym}_0])$
// Set first root key
$\pi_{S,s}.rk[\mathtt{asym}_0] \leftarrow rk$
// No incoming chain 0
// Initialize outgoing chain 0
$\pi_{S,s}.ck[\mathtt{out}_{0,0}] \leftarrow ck$
// Generate outgoing PQ ratchet key
$(\mathsf{rchpqpk}_{S,s,1}, \mathsf{rchpqsk}_{S,s,1}) \leftarrow\!\!\$\ \Pi_2.\mathsf{KGen}()$
// Save state
$\pi_{S,s}.st[\mathtt{asym}_1] \leftarrow (\mathsf{rchecsk}_{S,s,1}, \mathsf{rchpqsk}_{S,s,1})$
// Outgoing message
generate signature σ using idecsk_S as per Section 3.5.1 with values $\mathsf{rchecpk}_{S,s,1}, \mathsf{prepqct}_{S,s}, \mathsf{rchpqpk}_{S,s,1}$
    and pre-key hash $H(\mathsf{preecpk}_{R,i} \| \mathsf{prepqpk}_{R,i})$
return $(\mathsf{rchecpk}_{S,s,1}, \mathsf{prepqct}_{S,s}, \mathsf{rchpqpk}_{S,s,1}, H(\mathsf{preecpk}_{R,i} \| \mathsf{prepqpk}_{R,i}), \sigma)$

Figure 4: The operations of the PQ3 initial key exchange for initiator S in session $\pi_{S,s}$ with responder R using pre-key bundle $\mathsf{PKB}_{R,i}$.

The receiver R, to start a new session number r with sender S, after receiving an incoming session start message, proceeds as in Figure 5. By the end of SessionStartR, the responder has set the root keys for stages $[\mathtt{asym}_0]$ and $[\mathtt{asym}_1]$ and the chain keys for incoming chain $[\mathtt{in}_{0,0}]$ and outgoing chain $[\mathtt{out}_{1,0}]$, and produced an outgoing message for the initiator.

3.3 Asymmetric ratchet

Upon the receipt of a message from its peer containing ephemeral public keys allowing the creation of a new asymmetric ratchet stage, a party P in session $\pi_{P,p}$ that has set $\pi_{P,p}.rk[\mathtt{asym}_{j-1}]$ advances the asymmetric ratchet two steps (step j for the incoming chain, step $j+1$ for the outgoing chain) by proceeding as in Figure 6. By the end of AsymRatchet, the party has set the root keys for stages $[\mathtt{asym}_j]$ and $[\mathtt{asym}_{j+1}]$ and the chain keys for incoming chain $[\mathtt{in}_{j,0}]$ and outgoing chain $[\mathtt{out}_{j+1,0}]$, and produced an outgoing message for its peer.

3.4 Symmetric ratchet

In order to derive message keys in a chain $chain[j]$ where $chain[j] = \mathsf{in}[j]$ or $\mathsf{out}[j]$, party P in session $\pi_{P,p}$ proceeds as in Figure 7. By the end of SymRatchet, the party has derived the ℓth message key

SessionStartR($\pi_{R,r}, \mathsf{idecvk}_S, \mathsf{PKB}_{R,i}, (\mathsf{rchecpk}_{S,s,1}, \mathsf{prepqct}_{S,s}, \mathsf{rchpqpk}_{S,s,1}, h = H(\mathsf{preecpk}_{R,i} \| \mathsf{prepqpk}_{R,i}), \sigma)$)

Verify signature σ on incoming message using idecvk_S as per Section 3.5.1
Look up $\mathsf{preecsk}_{R,i}$ from $\mathsf{PKB}_{R,i}$ based on $h = H(\mathsf{preecpk}_{R,i} \| \mathsf{prepqpk}_{R,i})$
// Compute ratchet/prekey ECDH shared secret
$\mathsf{ecss} \leftarrow (\mathsf{rchecpk}_{S,s,1})^{\mathsf{preecsk}_{R,i}}$
// Compute prekey PQ shared secret
$\mathsf{pqss} \leftarrow \Pi_1.\mathsf{Dec}(\mathsf{prepqsk}_{R,i}, \mathsf{prepqct}_{S,s})$
// Derive session identifier and root/chain keys
$\pi_{R,r}.\mathsf{sid}[\mathtt{asym}_0] \leftarrow \mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{start}} \| \mathsf{preecpk}_{R,i} \| \mathsf{rchecpk}_{S,s,1} \| \mathsf{prepqct}_{S,s} \| \mathsf{prepqpk}_{R,i}$
$(rk, ck) \leftarrow \mathsf{KDFRKCK}(0, \mathsf{ecss}, \mathsf{pqss}, \pi_{R,r}.sid[0])$
// Set first root key
$\pi_{R,r}.rk[\mathtt{asym}_0] \leftarrow rk$
// No outgoing chain 0
// Initialize incoming chain 0
$\pi_{R,r}.ck[\mathtt{in}_{0,0}] \leftarrow ck$
// Generate outgoing ECDH ratchet key
$\mathsf{rchecsk}_{R,r,2} \leftarrow\!\!\$\ \mathbb{Z}_q, \mathsf{rchecpk}_{R,r,2} \leftarrow g^{\mathsf{rchecsk}_{R,r,2}}$
// Compute ratchet ECDH & PQ shared secret
$\mathsf{ecss}' \leftarrow (\mathsf{rchecpk}_{S,s,1})^{\mathsf{rchecsk}_{R,r,2}}$
$\mathsf{rchpqrnd}_{S,s,1} \leftarrow\!\!\$\ \Pi_2.\mathcal{R}$
$(\mathsf{rchpqct}_{R,r,1}, \mathsf{pqss}') \leftarrow \Pi_2.\mathsf{Enc}(\mathsf{rchpqpk}_{S,s,1}; \mathsf{rchpqrnd}_{S,s,1})$
// Derive session identifier and root/chain keys
$\pi_{R,r}.\mathsf{sid}[\mathtt{asym}_1] \leftarrow \mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{ratchet}} \| \mathsf{rchecpk}_{S,s,1} \| \mathsf{rchecpk}_{R,r,2} \| \mathsf{rchpqct}_{R,r,1} \| \mathsf{rchpqpk}_{S,s,1}$
$(rk', ck') \leftarrow \mathsf{KDFRKCK}(\pi_{R,r}.rk[\mathtt{asym}_0], \mathsf{ecss}', \mathsf{pqss}', \pi_{R,r}.\mathsf{sid}[\mathtt{asym}_1])$
// Set next root key
$\pi_{R,r}.rk[\mathtt{asym}_1] \leftarrow rk'$
// Initialize outgoing chain 1
$\pi_{R,r}.ck[\mathtt{out}_{1,0}] \leftarrow ck'$
// Generate outgoing PQ ratchet key
$(\mathsf{rchpqpk}_{R,r,2}, \mathsf{rchpqsk}_{R,r,2}) \leftarrow \Pi_2.\mathsf{KGen}()$
// Save state
$\pi_{R,r}.st[\mathtt{asym}_2] \leftarrow (\mathsf{rchecsk}_{R,r,2}, \mathsf{rchpqsk}_{R,r,2})$
// Outgoing message
generate signature σ using idecsk_R as per Section 3.5.1 with values $\mathsf{rchecpk}_{R,r,2}, \mathsf{rchpqct}_{R,r,1}, \mathsf{rchpqpk}_{R,r,2}$
return $(\mathsf{rchecpk}_{R,r,2}, \mathsf{rchpqct}_{R,r,1}, \mathsf{rchpqpk}_{R,r,2}, \sigma)$

Figure 5: The operations of the PQ3 initial key exchange for responder R in session $\pi_{R,r}$ with initiator S using pre-key bundle $\mathsf{PKB}_{R,i}$.

$\text{AsymRatchet}(\pi_{P,p}, (\mathsf{rchecpk}_{P',p',j+1}, \mathsf{rchpqct}_{P',p',j}, \mathsf{rchpqpk}_{P',p',j+1}), \sigma)$

1 : $j \leftarrow |\pi_{P,p}.[\mathtt{asym}]|$ // j is the current length of the asym chain in $\pi_{p,p}$, which will be odd for initiator and even for responder sessions
2 : Verify signature σ on incoming message using $\mathsf{idecvk}_{\pi_{P,p}.\mathsf{peerid}}$ as per Section 3.5.1
3 : // Compute ratchet ECDH shared secret
4 : $\mathsf{ecss} \leftarrow (\mathsf{rchecpk}_{P',p',j+1})^{\mathsf{rchecsk}_{P,p,j}}$
5 : // Compute ratchet PQ shared secret
6 : $\mathsf{pqss} \leftarrow \Pi_2.\mathsf{Dec}(\mathsf{rchpqsk}_{P,p,j}, \mathsf{rchpqct}_{P',p',j})$
7 : // Derive session identifier and root/chain keys
8 : $\pi_{P,p}.\mathsf{sid}[\mathtt{asym}_j] \leftarrow \mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{ratchet}} \| \mathsf{rchecpk}_{P,p,j} \| \mathsf{rchecpk}_{P',p',j+1} \| \mathsf{rchpqct}_{P',p',j} \| \mathsf{rchpqpk}_{P,p,j}$
9 : $(rk, ck) \leftarrow \mathsf{KDFRKCK}(\pi_{P,p}.rk[\mathtt{asym}_{j-1}], \mathsf{ecss}, \mathsf{pqss}, \pi_{P,p}.\mathsf{sid}[\mathtt{asym}_j])$
10 : // Set next root key
11 : $\pi_{P,p}.rk[\mathtt{asym}_j] \leftarrow rk$
12 : // Initialize incoming chain j
13 : $\pi_{P,p}.ck[\mathtt{in}_{j,0}] \leftarrow ck$
14 : // Generate outgoing ratchet ECDH key
15 : $\mathsf{rchecsk}_{P,p,j+2} \leftarrow\!\!\$\ \mathbb{Z}_q, \mathsf{rchecpk}_{P,p,j+2} \leftarrow g^{\mathsf{rchecsk}_{P,p,j+2}}$
16 : // Compute ratchet ECDH & PQ shared secret
17 : $\mathsf{ecss}' \leftarrow (\mathsf{rchecpk}_{P',p',j+1})^{\mathsf{rchecsk}_{P,p,j+2}}$
18 : $\mathsf{rchpqrnd}_{P,p,j+1} \leftarrow\!\!\$\ \Pi_2.\mathcal{R}$
19 : $(\mathsf{rchpqct}_{P,p,j+1}, \mathsf{pqss}') \leftarrow \Pi_2.\mathsf{Enc}(\mathsf{rchpqpk}_{P',p',j+1}; \mathsf{rchpqrnd}_{P,p,j+1})$
20 : // Derive session identifier and root/chain keys
21 : $\pi_{P,p}.\mathsf{sid}[\mathtt{asym}_{j+1}] \leftarrow \mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{ratchet}} \| \mathsf{rchecpk}_{P',p',j+1} \| \mathsf{rchecpk}_{P,p,j+2} \| \mathsf{rchpqct}_{P,p,j+1} \| \mathsf{rchpqpk}_{P',p',j+1}$
22 : $(rk', ck') \leftarrow \mathsf{KDFRKCK}(\pi_{P,p}.rk[\mathtt{asym}_j], \mathsf{ecss}', \mathsf{pqss}', \pi_{P,p}.\mathsf{sid}[\mathtt{asym}_{j+1}])$
23 : // Set next root key
24 : $\pi_{P,p}.rk[\mathtt{asym}_{j+1}] \leftarrow rk'$
25 : // Initialize outgoing chain $j+1$
26 : $\pi_{P,p}.ck[\mathtt{out}_{j+1,0}] \leftarrow ck'$ // Generate outgoing PQ ratchet key
27 : $(\mathsf{rchpqpk}_{P,p,j+2}, \mathsf{rchpqsk}_{P,p,j+2}) \leftarrow\!\!\$\ \Pi_2.\mathsf{KGen}()$
28 : // Save state
29 : $\pi_{P,p}.st[\mathtt{asym}_{j+2}] \leftarrow (\mathsf{rchecsk}_{P,p,j+2}, \mathsf{rchpqsk}_{P,p,j+2})$
30 : // Outgoing message
31 : generate signature σ using idecsk_P as per Section 3.5.1 with values $\mathsf{rchecpk}_{P,p,j+2}, \mathsf{rchpqct}_{P,p,j+1}, \mathsf{rchpqpk}_{P,p,j+2}$
32 : **return** $(\mathsf{rchecpk}_{P,p,j+2}, \mathsf{rchpqct}_{P,p,j+1}, \mathsf{rchpqpk}_{P,p,j+2}, \sigma)$

Figure 6: The operations of the PQ3 asymmetric ratchet for party P in session $\pi_{P,p}$.

$\pi_{P,p}.mk[chain_{j,\ell}]$ in this chain, and the next chain key $\pi_{P,p}.ck[chain_{j,\ell+1}]$.

SymRatchet($\pi_{P,p}, chain, j$) where $chain = \mathsf{in}$ or out

1 : $\ell \leftarrow |\pi_{P,p}.chain[j]|$ // Index of current last stage of $chain[j]$
2 : // Derive message key for this index
3 : $\pi_{P,p}.mk[chain_{j,\ell}] \leftarrow \mathsf{HKDF.Expand}(\pi_{P,p}.ck[chain_{j,\ell}], \mathtt{label}_{\text{mkderivation}}, 256)$
4 : $\pi_{P,p}.\mathsf{status}[chain_{j,\ell}] \leftarrow \mathtt{accept}$
5 : $\pi_{P,p}.k[chain_{j,\ell}] \leftarrow \pi_{P,p}.mk[chain_{j,\ell}]$
6 : // Derive chain key for next index
7 : $\pi_{P,p}.ck[chain_{j,\ell+1}] \leftarrow \mathsf{HKDF.Expand}(\pi_{P,p}.ck[chain_{j,\ell}], \mathtt{label}_{\text{ckderivation}}, 256)$

Figure 7: The operations of the PQ3 symmetric ratchet for party P in session $\pi_{P,p}$ on chain $chain[j]$.

3.5 Additional PQ3 components

3.5.1 Message authentication

All protocol messages transmitted in PQ3 are authenticated using signature scheme Σ. Each time party P sends a message, the following information (including explicit length fields) is signed using idecsk_P. The recipient correspondingly verifies each received message and signature using idecvk_P.

- $\mathtt{label}_{\text{messagesignature}}$: a fixed label
- a fixed-size version number
- *appct*: application ciphertext with length encoding
- *appad*: application authenticated data with length encoding
- *mkind*: a value derived from the message key indicating the location in the ratchet
- $\mathsf{rchecpk}$: the outgoing ratchet EC public key
- *msgind*: the message index
- $dst = \mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R$
- $\mathsf{rchpqpk}$: the outgoing ratchet PQ public key (or a length-0 string if the PQ portion is omitted)
- $\mathsf{rchpqct}$: the outgoing ratchet PQ ciphertext (or a length-0 string if the PQ portion is omitted)
- $H(\mathsf{preecpk}_{R,i} \| \mathsf{prepqpk}_{R,i})$: the hash of the recipient's pre-keys from the pre-key bundle (only included on the session-start message from the initiator to the responder, otherwise a length-0 string)

3.5.2 The KDFRKCK function

The initial key establishment and asymmetric ratchets make use of key derivation function KDFRKCK as shown in Figure 8, which combines up to three pieces of keying material (a root key rk, an ECDH shared secret ss_1, and a post-quantum shared secret ss_2) with a session identifier *sid*. The function produces a pair of keys (rk, ck) which will be used as root and chain keys.

$\mathsf{KDFRKCK}(rk, ss_1, ss_2, sid)$

1 : $ext_1 \leftarrow \mathsf{HKDF.Extract}(ikm = ss_1, salt = rk)$
2 : **if** $ss_2 = \perp$ **then** $ext_2 \leftarrow \mathsf{HKDF.Extract}(ikm = ext_1, salt = 0)$
3 : **else** $ext_2 \leftarrow \mathsf{HKDF.Extract}(ikm = ext_1, salt = ss_2)$
4 : $z \leftarrow \mathsf{HKDF.Expand}(ext_2, \mathtt{label}_{\mathrm{rootkeyderivation}} \| sid, 512)$
5 : $rk \leftarrow z[0 \ldots 255], ck \leftarrow z[256 \ldots 511]$
6 : **return** (rk, ck)

Figure 8: The key derivation function KDFRKCK used in PQ3 initial key establishment and asymmetric.

4 Security model

In this section, we present the multi-stage authenticated key exchange security model used to analyze the security of the PQ3 protocol. It is based on an adaptation of the security model developed by Cohn-Gordon et al. [CGCD+17] used to analyze the Signal protocol, which was based on the multi-stage AKE security model of Fischlin and Günther [FG14], which is a Bellare–Rogaway-style AKE security model [BR94].

The basic idea of the Bellare–Rogaway (BR) security model for authenticated key exchange is as follows; see [BMS19] for a more detailed introduction to AKE security models. The adversary interacts with a challenger who simulates all honest parties to the adversary. The adversary directs all interactions between the honest parties, and all communications flow via the adversary, modelling an active adversary. Additionally, the adversary is allowed to compromise various secrets of the honest parties, including long-term identity private keys, ephemeral private keys, intermediate session state, and established session keys. The adversary is then challenged to learn information about the session key in a target session run by an honest party, modelled as whether the adversary is able distinguish the session key from a uniformly random bitstring of the same length; if the adversary cannot distinguish this with probability substantially different from $\frac{1}{2}$, then the session keys are effectively random and unknown from the adversary's perspective. Given that the adversary can compromise so many secret values, security is defined only for session keys for which the adversary has not compromised sufficiently many inputs to make the key derivation trivial; this restriction is captured by the notion of *freshness*, and the specific details of freshness vary depending on the exact security characteristics the protocol aims to achieve. (Roughly speaking, allowing the adversary to compromise the long-term identity key and intermediate session state allows the model to capture the notion of forward secrecy, and allowing the adversary to compromise intermediate session state allows the model to capture the notion of post-compromise security.)

The original Bellare–Rogaway AKE security model was defined for key exchange protocols that establish a single session key, but modern protocols including TLS [JKSS12, DFGS15], QUIC [FG14], Signal [CGCD+17], and iMessage PQ3 establish multiple keys in a single session, which can be capture using Fischlin and Günther's multi-stage AKE security model [FG14] which extends the BR model.

In this section, we customize the multi-stage AKE security model that was used by Cohn-Gorden et al. [CGCD+17] to capture the security characteristics intended by the PQ3 protocol. Section 4.1 gives the main security experiment for the security model, and Section 4.2 details the freshness condition.

4.1 Security experiment

Definition 12 (Multi-stage key indistinguishability)**.** Let Π be a multi-stage key exchange protocol. Let $\mathcal{A}$ be a probabilistic algorithm. Define

$$\mathsf{Adv}^{\mathrm{IND}}_{\Pi, n_\mathsf{P}, n_\mathsf{B}, n_\mathsf{S}, n_\mathsf{s}}(\mathcal{A}) = \left| 2 \cdot \Pr\left[\mathrm{IND}^{\mathcal{A}}_{\Pi, n_\mathsf{P}, n_\mathsf{B}, n_\mathsf{S}, n_\mathsf{s}} \Rightarrow 1\right] - 1 \right|$$

where the experiment $\mathrm{IND}^{\mathcal{A}}_{\Pi}$ is as defined in Figure 9 and $n_\mathsf{P}, n_\mathsf{B}, n_\mathsf{S}, n_\mathsf{s}$ are positive integers representing the maximum number of parties, pre-key bundles per party, sessions per party, and stages per session, respectively.

The experiment $\mathrm{IND}^{\mathcal{A}}_{\Pi}$ includes the following global variables:

- b: a challenge bit

Role π.role	Stage s	Input state $\pi.st[s]$	Input intermediate keys	Output state	Output intermediate keys
init	$[\mathtt{asym}_0]$	$\perp$	$\perp$	$\mathsf{rchecsk}_{P,p,1}, \mathsf{rchpqsk}_{P,p,1}$	$\pi_{P,p}.rk[\mathtt{asym}_0], \pi_{P,p}.ck[\mathtt{in}_{0,0}]$
resp	$[\mathtt{asym}_0]$ & $[\mathtt{asym}_1]$	$\perp$	$\perp$	$\mathsf{rchecsk}_{P,p,1}, \mathsf{rchpqsk}_{P,p,1}$	$\pi_{P,p}.rk[\mathtt{asym}_0], \pi_{P,p}.ck[\mathtt{in}_{0,0}], \pi_{P,p}.rk[\mathtt{asym}_1], \pi_{P,p}.ck[\mathtt{out}_{1,0}]$
init	$[\mathtt{asym}_j]$ & $[\mathtt{asym}_{j+1}]$, $j \geq 1$ odd	$\mathsf{rchecsk}_{P,p,j}, \mathsf{rchpqsk}_{P,p,j}$	$\pi_{P,p}.rk[\mathtt{asym}_{j-1}]$	$\mathsf{rchecsk}_{P,p,j+2}, \mathsf{rchpqsk}_{P,p,j+2}$	$\pi_{P,p}.rk[\mathtt{asym}_j], \pi_{P,p}.ck[\mathtt{in}_{j,0}], \pi_{P,p}.rk[\mathtt{asym}_{j+1}], \pi_{P,p}.ck[\mathtt{out}_{j+1,0}]$
resp	$[\mathtt{asym}_j]$ & $[\mathtt{asym}_{j+1}]$, $j \geq 2$ even	$\mathsf{rchecsk}_{P,p,j}, \mathsf{rchpqsk}_{P,p,j}$	$\pi_{P,p}.rk[\mathtt{asym}_{j-1}]$	$\mathsf{rchecsk}_{P,p,j+2}, \mathsf{rchpqsk}_{P,p,j+2}$	$\pi_{P,p}.rk[\mathtt{asym}_j], \pi_{P,p}.ck[\mathtt{in}_{j,0}], \pi_{P,p}.rk[\mathtt{asym}_{j+1}], \pi_{P,p}.ck[\mathtt{out}_{j+1,0}]$
*	$[\mathtt{in}_{j,\ell}]$	$\perp$	$\pi_{P,p}.ck[\mathtt{in}_{j,\ell}]$	$\perp$	$\pi_{P,p}.ck[\mathtt{in}_{j,\ell+1}]$
*	$[\mathtt{out}_{j,\ell}]$	$\perp$	$\pi_{P,p}.ck[\mathtt{out}_{j,\ell}]$	$\perp$	$\pi_{P,p}.ck[\mathtt{out}_{j,\ell+1}]$

Table 2: Per-stage variables for session $\pi_{P,p}$ in the security model for the PQ3 protocol

Stage s	Session identifier $\pi.\mathsf{sid}[s]$	Session key $\pi.k[s]$
$[\mathtt{asym}_0]$	$\mathsf{id}_P \| \mathsf{idecvk}_P \| \mathsf{id}_{P'} \| \mathsf{idecvk}_{P'} \| \mathtt{label}_{\text{start}} \| \mathsf{preecpk}_{P',i} \| \mathsf{rchecpk}_{P,p,0} \| \mathsf{prepqct}_{P,p} \| \mathsf{prepqpk}_{P',i}$	$\perp$
$[\mathtt{asym}_j]$	$\mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{ratchet}} \| \mathsf{rchecpk}_{P,p,j} \| \mathsf{rchecpk}_{P',p',j+1} \| \mathsf{rchpqct}_{P',p',j} \| \mathsf{rchpqpk}_{P,p,j}$	$\perp$
$[\mathtt{in}_{j,\ell}]$	$\pi_{P,p}.\mathsf{sid}[[\mathtt{asym}_j]] \| \mathtt{label}_{\text{in}} \| \ell$	$\pi_{P,p}.mk[\mathtt{in}_{j,\ell}]$
$[\mathtt{out}_{j,\ell}]$	$\pi_{P,p}.\mathsf{sid}[[\mathtt{asym}_j]] \| \mathtt{label}_{\text{out}} \| \ell$	$\pi_{P,p}.mk[\mathtt{out}_{j,\ell}]$

Table 3: Per-stage session identifiers and keys for session $\pi_{P,p}$ in the security model for the PQ3 protocol

- $\mathsf{tested} = (P, p, s)$ or $\perp$: the record of the input to the query $\mathsf{Test}(P, p, s)$ or $\perp$ if no Test query happened

Additional experiment $\text{IND}^{\mathcal{A}}_{\Pi}$ keeps additional variables as follows:

- $\mathsf{rev\_session}[P, p, s] \in \{\mathsf{true}, \mathsf{false}\}$: whether $\mathsf{RevSessKey}(P, p, s)$ was called or not
- $\mathsf{rev\_state}[P, p, s] \in \{\mathsf{true}, \mathsf{false}\}$: whether $\mathsf{RevState}(P, p, s)$ was called or not
- $\mathsf{rev\_rchkey}[P, p, s, t] \in \{\mathsf{true}, \mathsf{false}\}$: whether $\mathsf{RevRchKey}(P, p, s, t)$ was called or not

In order to instantiate PQ3 within the model of Figure 9, we define the per-stage variables in Table 2 and the stage session identifiers in Table 3.

4.2 Freshness

In the PQ3 protocol, there are many different types of stages: the initial key establishment, the stages along the asymmetric ratchet, and the stages along the symmetric ratchet. These different types of stages have subtly different security properties: the secret values allowed to be compromised by the adversary without trivially undermining security differ across the different types of stages. The freshness condition models these conditions for each type of stage.

The overall freshness condition is given in Definition 17, and is built up from several intermediate definitions. Roughly speaking, a session key is fresh if:

- It is valid (Definition 13): the stage must have accepted, and the adversary cannot have revealed the peer's long-term key before the stage accepted (which would otherwise allow for trivial impersonation).
- Not all of the inputs to the stage have been revealed (Definition 16): at least one input to the derivation of the session key in this stage cannot have been revealed (which would otherwise allow for trivial session key computation). As each stage type (initial key exchange, asymmetric ratchet, symmetric ratchet) has different inputs, this condition is customized to each stage type as shown in Table 4. It makes use of intermediate definitions tracking whether the intermediate session state has been revealed (Definition 15) or whether the peer's inputs to the stage have been revealed (Definition 14).
- The session key has not been revealed, either at the session itself or at the peer of the session, if it exists.

Definition 13 (Validity)**.** *Validity* captures some bare minimum conditions of a stage for the session key to be plausibly fresh: the stage must have accepted and the adversary has not revealed the peer's long-term key before the stage accepted (which captures forward secrecy). Define the predicate $\mathsf{valid}(P, p, s)$ as:

$$\begin{aligned}\mathsf{valid}(P, p, s) = (&\pi_{P,p}.\mathsf{status}[s] = \mathtt{accept}) \\ &\wedge \mathsf{time}(\pi_{P,p}.\mathsf{status}[s] \leftarrow \mathtt{accept}) < \mathsf{time}(\mathsf{RevLongTermKey}(\pi_{P,p}.\mathsf{peerid}))\end{aligned}$$

$\mathrm{IND}^{\mathcal{A}}_{\Pi,n_\mathsf{P},n_\mathsf{B},n_\mathsf{S},n_\mathsf{s}}$

$\mathsf{b} \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{tested} \leftarrow \perp$
for $P = 1 \ldots n_\mathsf{P}$ **do**
  $(\mathsf{idecvk}_P, \mathsf{idecsk}_P) \leftarrow\!\!\$\ \mathrm{IDKeyGen}()$
  for $i = 1 \ldots n_\mathsf{B}$ **do**
    $(\mathsf{preecpk}_{P,i}, \mathsf{preecsk}_{P,i}, \mathsf{prepqpk}_{P,i}, \mathsf{prepqsk}_{P,i}, \sigma) \leftarrow\!\!\$\ \mathrm{PKBGen}(\mathsf{idecsk}_P)$
$\mathit{pubinfo} \leftarrow (\mathsf{idecvk}_*, \mathsf{preecpk}_*, \mathsf{prepqpk}_*, \sigma_*)$
$\mathsf{b}' \leftarrow \mathcal{A}^{\mathsf{Send},\mathsf{Rev}*,\mathsf{Test}}(\mathit{pubinfo})$
if $(\mathsf{tested} = \perp) \vee \neg\mathsf{fresh}(\mathsf{tested})$ **then**
  $r \leftarrow\!\!\$\ \{0,1\}$
  return r
if $\mathsf{b} = \mathsf{b}'$ **then return** 1
else return 0

$\mathsf{Send}(P,p,m)$

if $\pi_{P,p} = \perp$ **then**
  parse $(P', i, \mathsf{role}, m') \leftarrow m$
  if $\mathsf{role} = \mathtt{init}$ **then return** $\mathrm{SessionStartS}(\pi_{P,p}, \mathsf{idecvk}_{P'}, \mathsf{PKB}_{P',i})$
  else $\mathsf{role} = \mathtt{resp}$ **then return** $\mathrm{SessionStartR}(\pi_{P,p}, \mathsf{idecvk}_{P'}, \mathsf{PKB}_{P,i}, m')$
else
  parse $(\mathit{type}, m') \leftarrow m$
  if $\mathit{type} = \mathit{asym}$ **then return** $\mathrm{AsymRatchet}(\pi_{P,p}, m')$
  else $\mathit{type} = \mathit{sym}$
    parse $(\mathit{chain}, j) \leftarrow m'$
    return $\mathrm{SymRatchet}(\pi_{P,p}.\mathit{chain}[j])$

$\mathsf{RevLongTermKey}(P)$

$\mathsf{rev\_ltk}[P] \leftarrow \mathsf{true}$
return idecsk_P

$\mathsf{RevPreKey}(P,i,t)$

$\mathsf{rev\_prekey}[P,i,t] \leftarrow \mathsf{true}$
if $t = \mathsf{ec}$ **then return** $\mathsf{preecsk}_{P,i}$
elseif $t = \mathsf{pqsk}$ **then return** $\mathsf{prepqsk}_{P,i}$)
elseif $t = \mathsf{pqrnd}$ **then return** $\mathsf{prepqrnd}_{P,i}$)

$\mathsf{RevRchKey}(P,p,s,t)$

$\mathsf{rev\_rchkey}[P,p,s,t] \leftarrow \mathsf{true}$
if $t = \mathsf{ec}$ **then return** $\mathsf{rchecsk}_{P,p,s}$
elseif $t = \mathsf{pqsk}$ **then return** $\mathsf{rchpqsk}_{P,p,s}$)
elseif $t = \mathsf{pqrnd}$ **then return** $\mathsf{rchpqrnd}_{P,p,s}$)

$\mathsf{RevSessKey}(P,p,s)$

$\mathsf{rev\_session}[P,p,s] \leftarrow \mathsf{true}$
return $\pi_{P,p}.k[s]$ as per Table 3

$\mathsf{RevState}(P,p,s)$

$\mathsf{rev\_state}[P,p,s] \leftarrow \mathsf{true}$
return $\pi_{P,p}.st[s]$
  and input intermediate keys in Table 2

$\mathsf{Test}(P,p,s)$

if $\mathsf{tested} \neq \perp$ **then return** $\perp$
if $\pi_{P,p}.\mathsf{status}[s] \neq \mathtt{accept}$ **then return** $\perp$
if $\pi_{P,p}.k[s] = \perp$ **then return** $\perp$
$\mathsf{tested} \leftarrow (P,p,s)$
if $\mathsf{b} = 0$ **then** $k \leftarrow \pi_{P,p}.k[s]$
else $k \leftarrow\!\!\$\ \mathcal{K}$
return k

Figure 9: Security experiment for adversary $\mathcal{A}$ against multi-stage key indistinguishability of protocol Π

Note that within a session $\pi_{P,p}$, if $\mathsf{valid}(P, p, s)$ holds and s' is a stage of $\pi_{P,p}$ that accepted prior to stage s, then $\mathsf{valid}(P, p, s')$ also holds.

Definition 14 (Peer ratchet freshness)**.** *Peer ratchet freshness* captures the idea that the ratchet secret key of the peer of a stage has not been revealed. Define the predicate $\mathsf{unrev}_{\mathsf{peer}}(P, p, j, t)$ as:

$$\mathsf{unrev}_{\mathsf{peer}}(P, p, j, t) = \forall p' : \pi_{P,p}.\mathsf{sid}[[\mathtt{asym}_j]] = \pi_{\pi_{P,p}.\mathsf{peerid},p'}.\mathsf{sid}[[\mathtt{asym}_j]] \implies \neg\mathsf{rev\_rchkey}[\pi_{P,p}.\mathsf{peerid}, p', j, t]$$

Definition 15 (State freshness)**.** *State freshness* captures the idea that the state (intermediate keys) of a stage has not been revealed, nor has the state (intermediate keys) of the stage at a peer session with a matching session identifier. Define the predicate $\mathsf{unrev}_{\mathsf{state}}(P, p, s)$ as:

$$\begin{aligned}\mathsf{unrev}_{\mathsf{state}}(P, p, s) = {} & \neg\mathsf{rev\_state}[P, p, s] \\ & \wedge \big(\forall p' : \pi_{P,p}.\mathsf{sid}[s] = \pi_{\pi_{P,p}.\mathsf{peerid},p'}.\mathsf{sid}[s] \implies \neg\mathsf{rev\_state}[\pi_{P,p}.\mathsf{peerid}, p', s]\big)\end{aligned}$$

Definition 16 (Stage input unrevealedness)**.** *Stage input unrevealedness* captures the idea that at least one input to the derivation of the session key of the stage has not been revealed. Because session keys of different stages have different inputs, each stage type has a different stage unrevealedness predicate. Define the predicate $\mathsf{unrev}(P, p, s)$ as shown in Table 4.

Stage s	**Role** $\pi_{P,p}.\mathsf{role}$	$\mathsf{unrev}(P, p, s)$
$[\mathtt{asym}_0]$	`init`	$\neg\mathsf{rev\_prekey}[R, i, \mathsf{ec}] \wedge \neg\mathsf{rev\_rchkey}[P, p, 1, \mathsf{ec}] \wedge \neg\mathsf{rev\_prekey}[R, i, \mathsf{pqsk}] \wedge \neg\mathsf{rev\_prekey}[P, p, \mathsf{pqrnd}]$
$[\mathtt{asym}_0]$	`resp`	$\neg\mathsf{rev\_prekey}[P, i, \mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, 1, \mathsf{ec}) \wedge \neg\mathsf{rev\_prekey}[R, i, \mathsf{pqsk}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, 1, \mathsf{pqrnd})$
$[\mathtt{asym}_j] : j > 0, j$ odd	`init`	$\big(\mathsf{unrev}_{\mathsf{state}}(P, p, [\mathtt{asym}_j] \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_{j-1}])\big)$ $\vee\big(\neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j+1, \mathsf{ec}) \wedge \neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{pqsk}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j, \mathsf{pqrnd})\big)$
$[\mathtt{asym}_{j+1}] : j > 0, j$ odd	`init`	$\big(\mathsf{unrev}_{\mathsf{state}}(P, p, [\mathtt{asym}_{j+1}]) \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_j])\big)$ $\vee\big(\neg\mathsf{rev\_rchkey}[P, p, j+2, \mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j+1, \mathsf{ec}) \wedge \neg\mathsf{rev\_rchkey}[P, p, j+1, \mathsf{pqrnd}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j+1, \mathsf{pqsk})\big)$
$[\mathtt{asym}_j] : j > 0, j$ odd	`resp`	same as for $[\mathtt{asym}_j] : j > 0, j$ odd with $\mathsf{role} = \mathtt{init}$
$[\mathtt{asym}_{j+1}] : j > 0, j$ odd	`resp`	same as for $[\mathtt{asym}_{j+1}] : j > 0, j$ odd with $\mathsf{role} = \mathtt{init}$
$[chain_{j,0}] : chain \in \{\mathsf{in}, \mathsf{out}\}$	$*$	$\mathsf{unrev}_{\mathsf{state}}(P, p, [chain_{j,0}]) \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_j])$
$[chain_{j,\ell}] : chain \in \{\mathsf{in}, \mathsf{out}\}, \ell > 0$	$*$	$\mathsf{unrev}_{\mathsf{state}}(P, p, [chain_{j,\ell}]) \wedge \mathsf{unrev}(P, p, [chain_{j,\ell-1}])$

Table 4: Stage unrevealedness predicates $\mathsf{unrev}(P, p, s)$

Definition 17 (Freshness)**.** *Freshness* captures everything that is necessary for a session key to be plausibly secure: the stage is valid and relevant input values are unrevealed according to the previous definitions, and the adversary has not revealed the session key of the stage or of that stage at any matching session. Let s be a stage in the pth session at party P. Define the predicate $\mathsf{fresh}(P, p, s)$ as:

$$\begin{aligned}\mathsf{fresh}(P, p, s) = {} & \mathsf{valid}(P, p, s) \wedge \mathsf{unrev}(P, p, s) \\ & \wedge \neg\mathsf{rev\_session}[P, p, s] \\ & \wedge \big(\forall p' : \pi_{P,p}.\mathsf{sid}[s] = \pi_{\pi_{P,p}.\mathsf{peerid},p'}.\mathsf{sid}[s] \implies \neg\mathsf{rev\_session}[\pi_{P,p}.\mathsf{peerid}, p', s]\big)\end{aligned}$$

Interpreting the security model: forward secrecy and post-compromise security. It can be seen that the security model implies the main desired security characteristics as follows.

- Forward secrecy: According to Definition 13, the adversary is only prohibited from revealing the long-term key of the peer of the target session before the target session has accepted, but can do so after it has accepted. Furthermore, according to Definition 17, a stage remains fresh even if the session key or state of a later session is compromised.
- Post-compromise secrecy: According to Definition 16, a stage remains fresh if some, but not all of the inputs to the state have been compromised. In other words, if the chaining state from an earlier asymmetric stage is compromised, but the ephemeral key exchange in this asymmetric stage is uncompromised, then the stage should still be considered fresh, which is post-compromise security / healing.

5 Security proof

Although the PQ3 protocol derives many root and chain keys, it is the message keys that are considered the session keys that are the "output" of the multi-stage key exchange protocol and which are required to be indistinguishable from random. In order to prove that a particular message key is secure, we need to show that the root and chain keys involved in the derivation of that message key are secure. The proof proceeds modularly in multiple lemmas, each considering the root and chain keys derived in different types of stages in the protocol.

We begin with an overview of the proof approach, then proceed with the lemmas for various stages of the PQ3 protocol.

5.1 Overview of proof and main theorem

The lemmas for the various stages are as follows:

- Lemma 3 shows that the root key $\pi_{S,s}.rk[\mathtt{asym}_0]$ and chain key $\pi_{S,s}.ck[\mathtt{out}_{0,0}]$ established by the initiator in the initial key establishment are secure.
- Lemma 4 shows that the root keys $\pi_{R,r}.rk[\mathtt{asym}_0]$ and $\pi_{R,r}.rk[\mathtt{asym}_1]$, and chain keys $\pi_{R,r}.ck[\mathtt{in}_{0,0}]$ and $\pi_{R,r}.ck[\mathtt{out}_{1,0}]$ established by the responder in the initial key establishment are secure.
- Lemmas 5 and 6 show that the root and chain keys established in the first half and the second half of the asymmetric ratchet are secure.
- Lemmas 7 and 8 show that the chain keys and message keys established in the symmetric ratchet are secure.

It is the last two lemmas, Lemmas 7 and 8, that, when combined with the previous lemmas, yield the overall result, collected in Theorem 1: that message keys output by PQ3 are secure in the multi-stage authenticated key exchange security model.

Each lemma assumes that the stage in question is sufficiently unrevealed based on the relevant freshness condition, and as described above shows that the root and chain keys established in that stage are secure. For later stages, clauses of the freshness condition could imply that earlier stages are also sufficiently unrevealed, and hence the lemma for the earlier stage would yield that the root and chain keys established in that earlier stage are secure.

The proofs of the lemmas are shown using a sequence of games, starting from the original multi-stage authenticated key exchange indistinguishability experiment (Definition 12) for the PQ3 protocol. Although the specifics of the sequence of games varies depending on the particular lemma, we provide a little insight here into the overall approach.

- We usually make a hop enforcing that honestly generated Diffie–Hellman public keys are unique; since the protocol does not have separate nonces, unique DH public keys leads to unique session identifiers (Lemma 1 and Lemma 2) and helps with uniqueness during authentication.
- As necessary, we guess the index of the target session, the peer session, the owner of the target session, the peer of the target session, or the index of the responder's pre-key bundle. This guessing results in a non-tight proof.
- We reject any signatures received not generated by an honest party, giving a reduction to unforgeability of the signature scheme.
- For elliptic curve Diffie–Hellman-based key exchanges, in the target stage, we replace the ECDH shared secret with a random value. Note that technically we replace the intermediate key *ext* derived from the ECDH shared secret using HKDF.Extract inside KDFRKCK with a random value by using the PRF-ODH assumption; this is necessary because the reduction may be required to derive ECDH shared secrets with some of the challenge values in other sessions, and hence the reduction requires the assistance of one of the PRF-ODH oracles to do so.

- For post-quantum KEM-based key exchanges, in the target stage, we replace the KEM shared secret with a random value, giving a reduction to the IND-CCA security of the KEM. (CPA security does not suffice here, as the reduction may need to compute shared secrets involving the same KEM public key used or replayed in other sessions, and hence the reduction requires the assistance of a decapsulation oracle to do so.)
- After the ECDH or PQ shared secrets have been replaced with random values, subsequent keys derived from those are replaced with random values, giving a reduction to the secure of the relevant pseudorandom function, either HKDF.Extract, HKDF.Expand, or the combined KDFRKCK, in one or more of their arguments.

There are also two helper lemmas (Lemmas 1 and 2) that prove that session identifiers established by honest parties are unique.

Main theorem. The following theorem summarizes that the iMessage PQ3 protocol achieves security in the multi-stage security model of Section 4, under appropriate assumptions on the building blocks.

Theorem 1. *Assume that* Σ *is* EUF-CMA*, that* HKDF.Expand *and* HKDF.*Extract are both PRFs in their first argument, and that* KDFRKCK *is a PRF in its first and third arguments. Assume that either the PRF-ODH assumption holds for the elliptic curve group with* HKDF.Extract *in its first argument, or that* Π_1 *and* Π_2 IND-CCA*-secure KEMs. In the iMessage PQ3 protocol, consider the* ℓ*th message key in the symmetric ratchet stemming from the* j*th stage of the asymmetric ratchet in session* $\pi_{P,p}$*; assume that this stage is fresh according to Definition 17. Then, in the multi-stage security model of Section 4, this message key is indistinguishable from a random bitstring of the same length.*

Proof. Let t be an upper-bound on the runtime of the adversary. For a security notion x and a scheme y, let

$$\epsilon_y^{\mathrm{x}} = \max_{\mathcal{A}:\mathrm{time}(\mathcal{A})\leq t} \mathsf{Adv}_y^{\mathrm{x}}(\mathcal{A})$$

where the maximum is taken over all algorithms $\mathcal{A}$ with running time at most t.

Let the target session key be $\pi_{P,p}.mk[chain_{j,\ell}]$, which is assumed to be fresh. We will derive a bound on the adversary's advantage in distinguishing $\pi_{P,p}.mk[chain_{j,\ell}]$ from random by combining the various lemmas listed above.

- If $\pi_{P,p}$ is an initiator session, then by Lemma 3, the adversary's advantage in distinguishing $\pi_{P,p}.rk[\mathtt{asym}_0]$ and $\pi_{P,p}.ck[\mathtt{out}_{0,0}]$ from random is at most

$$\epsilon_{hs-init} = \frac{1}{q}\binom{n_{\mathsf{P}}n_{\mathsf{B}} + n_{\mathsf{P}}n_{\mathsf{S}}n_{\mathsf{s}}}{2} + n_{\mathsf{P}}^2 n_{\mathsf{S}} \left(\begin{array}{l} \epsilon_{\Sigma}^{\text{EUF-CMA}} \\ \quad + n_{\mathsf{B}} \min \left\{ \begin{array}{l} \frac{1}{q} + \epsilon_{g,\mathsf{HKDF.Extract}}^{\mathrm{prf}_1-\mathrm{odh}} + \epsilon_{\mathsf{HKDF.Extract}}^{\mathrm{prf}_1} + \epsilon_{\mathsf{HKDF.Expand}}^{\mathrm{prf}_1}, \\ \epsilon_{\Pi_1}^{\text{IND-CCA}} + \epsilon_{\mathsf{KDFRKCK}}^{\mathrm{prf}_3} \end{array} \right\} \end{array} \right)$$

- If $\pi_{P,p}$ is a responder session, then by Lemma 4, the adversary's advantage in distinguishing $\pi_{P,p}.rk[\mathtt{asym}_0]$ and $\pi_{P,p}.ck[\mathtt{in}_{0,0}]$ from random is at most

$$\epsilon_{hs-resp} = \frac{1}{q}\binom{n_{\mathsf{P}}n_{\mathsf{B}} + n_{\mathsf{P}}n_{\mathsf{S}}n_{\mathsf{s}}}{2} + n_{\mathsf{P}}^2 n_{\mathsf{S}}^2 n_{\mathsf{B}} \left(\begin{array}{l} \epsilon_{\Sigma}^{\text{EUF-CMA}} \\ \quad + \min \left\{ \begin{array}{l} \frac{1}{q} + \epsilon_{g,\mathsf{HKDF.Extract}}^{\mathrm{prf}_1-\mathrm{odh}} + \epsilon_{\mathsf{HKDF.Extract}}^{\mathrm{prf}_1} + \epsilon_{\mathsf{HKDF.Expand}}^{\mathrm{prf}_1}, \\ \epsilon_{\Pi_1}^{\text{IND-CCA}} + \epsilon_{\mathsf{KDFRKCK}}^{\mathrm{prf}_3} \end{array} \right\} \end{array} \right)$$

- Let

$$\epsilon_{hs} = \begin{cases} \epsilon_{hs-init}, & \text{if } \pi_{P,p} \text{ is an initiator session,} \\ \epsilon_{hs-resp}, & \text{if } \pi_{P,p} \text{ is a responder session} \end{cases}$$

- By repeated application of Lemma 5 and Lemma 6, the adversary's advantage in distinguishing $\pi_{P,p}.rk[\mathtt{asym}_j]$ and $\pi_{P,p}.ck[chain_{j,0}]$ from random is at most

$$\epsilon_{asym-j} = \epsilon_{hs} + j(\epsilon_{asym-first} + \epsilon_{asym-second})$$

where

$$\epsilon_{asym-first} = \left(\begin{array}{l} \epsilon_{\Sigma}^{\text{EUF-CMA}} \\ + \max\left\{ \epsilon_{\mathsf{KDFRKCK}}^{\text{prf}_1}, \min\left\{ \begin{array}{l} \frac{1}{q} + \epsilon_{g,\mathsf{HKDF.Extract}}^{\text{prf}_1-\text{odh}} + \epsilon_{\mathsf{HKDF.Extract}}^{\text{prf}_1} + \epsilon_{\mathsf{HKDF.Expand}}^{\text{prf}_1}, \\ \epsilon_{\Pi_2}^{\text{IND-CCA}} + \epsilon_{\mathsf{KDFRKCK}}^{\text{prf}_3} \end{array} \right\} \right\} \end{array} \right)$$

- By Lemma 7 and repeated application of Lemma 8, the adversary's advantage in distinguishing $\pi_{P,p}.ck[chain_{j,\ell}]$ and $\pi_{P,p}.mk[chain_{j,\ell}]$ from random is at most

$$\epsilon_{sym-j-\ell} = \epsilon_{hs} + j(\epsilon_{asym-first} + \epsilon_{asym-second}) + 2\ell \cdot \epsilon_{\mathsf{HKDF.Expand}}^{\text{prf}_1}$$

□

Hybrid security of ECDH and post-quantum. Theorem 1 shows that PQ3 message keys are secure in the hybrid traditional/post-quantum setting, meaning that they are secure if either the elliptic curve Diffie–Hellman problem remains hard (i.e., the PRF-ODH assumption holds in the relevant elliptic curve group when combined with HKDF), or the post-quantum scheme remains secure (i.e., IND-CCA security holds for the two post-quantum KEMs used). This can be seen by observing that the relevant advantage bounds involve a term of the form

$$\min\left\{ \epsilon_{g,\mathsf{HKDF.Extract}}^{\text{prf}_1-\text{odh}} + \dots,\ \epsilon_{\Pi}^{\text{ind}-\text{cca}} + \dots \right\}$$

which implies that security holds even if one of the two assumptions is broken (i.e., has high advantage), provided the other is unbroken (i.e., has low advantage).

5.2 Lemmas 1 and 2: Uniqueness of session identifiers

It will be helpful in later proofs to know that session identifiers are sufficiently unique. We prove some preliminary lemmas to that effect.

In all these lemmas, we assume that no two honest parties ever generate the same Diffie–Hellman public key. Our later proofs have a game hopping step to that effect, so that will be true at any of the times in later proofs where we rely on these lemmas about unique session identifiers.

Lemma 1 (Uniqueness of $[\mathtt{asym}_0]$ session identifiers)**.** *Suppose no two honest parties ever generate the same Diffie–Hellman public key. If $sid = \pi_{P,p}.\mathsf{sid}[\mathtt{asym}_0]$ for some session $\pi_{P,p}$, then there is no other session at party P with the same session identifier. Furthermore, there is at most one other session with the same identifier, which if it exists is at $R = \pi_{P,p}.\mathsf{peerid}$ and also in stage $[\mathtt{asym}_0]$.*

Proof. Recall that the $[\mathtt{asym}_0]$ session identifier for a session with initiator S, responder R, and responder prekey bundle indexed by i is

$$\mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{start}} \| \mathsf{preecpk}_{R,i} \| \mathsf{rchecpk}_{S,s,1} \| \mathsf{prepqct}_{S,s} \| \mathsf{prepqpk}_{R,i}$$

Initiator S will set this to be the session identifier for at most 1 stage across all its sessions. That stage if it exists will be an $[\mathtt{asym}_0]$ stage due to the presence of $\mathtt{label}_{\text{start}}$. That stage if it exists will be unique because the value $\mathsf{rchecpk}_{S,s,1}$ is generated by S and is assumed to be unique across all DH keys generated by S.

Responder R will set this to be the session identifier for at most 1 stage across all its sessions. That stage if it exists will be an $[\mathtt{asym}_0]$ stage due to the presence of $\mathtt{label}_{\text{start}}$. That stage if it exists will be unique because R will use prekey bundle i in at most one session, and the value $\mathsf{preecpk}_{R,i}$ is generated by R and is assumed to be unique across all DH keys generated by R.

No other honest party will set this to be a session identifier because of uniqueness of identity public keys. □

Lemma 2 (Uniqueness of $[\mathtt{asym}_j]$ session identifiers)**.** *Suppose no two honest parties ever generate the same Diffie–Hellman public key. If $sid = \pi_{P,p}.\mathsf{sid}[\mathtt{asym}_j]$ for some session $\pi_{P,p}$, then with probability at least $1 - 1/q$ there is no other session at party P with the same session identifier.*

Proof. Recall that the $[\mathtt{asym}_j]$ session identifier for a session with initiator S, responder R is

$$\mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{ratchet}} \| \mathsf{rchecpk}_{P,p,j} \| \mathsf{rchecpk}_{P',p',j+1} \| \mathsf{rchpqct}_{P',p',j} \| \mathsf{rchpqpk}_{P,p,j}$$

and that the $[\mathtt{asym}_{j+1}]$ session identifier is

$$\mathsf{id}_S \| \mathsf{idecvk}_S \| \mathsf{id}_R \| \mathsf{idecvk}_R \| \mathtt{label}_{\text{ratchet}} \| \mathsf{rchecpk}_{P',p',j+1} \| \mathsf{rchecpk}_{P,p,j+2} \| \mathsf{rchpqct}_{P,p,j+1} \| \mathsf{rchpqpk}_{P',p',j+1}$$

We ensure uniqueness by looking at the 6th and 7th components of the above two session identifiers.

Suppose either $\pi_{P,p}.\mathsf{role} = \mathtt{init}$ and j is odd, or $\pi_{P,p}.\mathsf{role} = \mathtt{resp}$ and j is even; in other words, we are considering when $[\mathtt{asym}_j]$ is the first of the two stages set in AsymRatchet. Party P will set this to be the session identifier for at most one $[\mathtt{asym}_j]$ stage with j of this parity, because the 6th component of the session identifier for `asym` stages of this parity is controlled by P and P picks a new $\mathsf{rchecpk}_{P,p,j}$ value for this in each such stage; these are assumed to be unique across all DH keys generated by P.

Consider now the session identifier for a $[\mathtt{asym}_{j'}]$ stage for j' of the opposite parity. Let c_6 and c_7 denote the 6th and 7th components of the $[\mathtt{asym}_j]$ session identifier. Let c'_6 and c'_7 denote the 6th and 7th components of the $[\mathtt{asym}_{j'}]$ session identifier.

In order for the session identifier for a $[\mathtt{asym}_{j'}]$ stage for j' even to be equal to the session identifier for $[\mathtt{asym}_j]$ stage for j odd, it would need to be the case that $c'_6 = c_7$, which is possible because the adversary could controls c'_6. But it would also need to be the case that $c_7 = c'_7$. Yet c'_7 is under the control of honest party P, is picked after c_7 is provided, and is picked uniformly at random from a set of size q, so the chance that $c_7 = c'_7$ is at most $1/q$.

Now consider the case when either $\pi_{P,p}.\mathsf{role} = \mathtt{init}$ and j is even, or $\pi_{P,p}.\mathsf{role} = \mathtt{resp}$ and j is odd; in other words, we are considering when $[\mathtt{asym}_j]$ is the second of the two stages set in AsymRatchet. An analogous argument works for this case, focusing on the 7th component of the session identifier, which is controlled by party P and picked freshly for this stage. □

5.3 Lemma 3: Initial key establishment for the initiator

Lemma 3. *Consider session $\pi_{S,s}$ such that $\pi_{S,s}.\mathsf{role} = \mathtt{init}$. Assume that Σ is* EUF-CMA*, that* HKDF.Expand *and* HKDF.Extract *are both PRFs in their first argument, and that* KDFRKCK *is a PRF in its third argument. Assume that either the PRF-ODH assumption holds for the elliptic curve group with* HKDF.Extract *in its first argument, or that Π_1 is an* IND-CCA*-secure KEM. If $\mathsf{valid}(S, s, [\mathtt{asym}_0]) \wedge \mathsf{unrev}(S, s, [\mathtt{asym}_0])$ holds, then $\pi_{S,s}.rk[\mathtt{asym}_0]$ and $\pi_{S,s}.ck[\mathtt{out}_{0,0}]$ are indistinguishable from random.*

Proof.

Game 0 (Original game). This game is the original multi-stage key indistinguishability security experiment for the PQ3 protocol. Thus

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\text{IND}}_{\Pi, n_\mathsf{P}, n_\mathsf{B}, n_\mathsf{S}, n_\mathsf{s}}(\mathcal{A})$$

Game 1 (Require distinct honestly generated DH public keys). In this game, we ensure there is no collision on any honestly generated DH public keys. The challenger maintains a list of all DH private values generated during the game; if any DH private value appears twice, the challenger aborts the simulation and the adversary automatically wins. The maximum number of honestly generated DH key pairs is $n_\mathsf{P} n_\mathsf{B} + n_\mathsf{P} n_\mathsf{S} n_\mathsf{s}$ and thus

$$\mathsf{Adv}_0 \leq \frac{1}{q} \binom{n_\mathsf{P} n_\mathsf{B} + n_\mathsf{P} n_\mathsf{S} n_\mathsf{s}}{2} + \mathsf{Adv}_1$$

where q is the order of the group.

Game 2 (Guess index of target session). In this game, the challenger guesses in advance the index $(S, s, 0) \in \{1, \ldots, n_\mathsf{P}\} \times \{1, \ldots, n_\mathsf{S}\} \times \{0\}$ of the session and stage against which the Test query is issued, and aborts if the guess is incorrect. Thus

$$\mathsf{Adv}_1 = n_\mathsf{P} n_\mathsf{S} \mathsf{Adv}_2$$

Game 3 (Guess peer of target session). In this game, the challenger guesses in advance the identity $R \in \{1, \ldots, n_\mathsf{P}\}$ of the peer of the target session, and aborts if the guess is incorrect. Thus

$$\mathsf{Adv}_2 = n_\mathsf{P} \mathsf{Adv}_3$$

Game 4 (Abort if forged signature of prekey bundle). In this game, the challenger aborts if SessionStartS($\pi_{S,s}, \dots$) accepted the signature on the prekey bundle $\mathsf{PKB}_{R,i}$ it received, but that prekey bundle was never signed by party R, and $\pi_{S,s}.[\mathtt{asym}_0]$ accepted before R's long-term key was revealed. If the abort event occurs, then $\mathsf{PKB}_{R,i}.\sigma$ constitutes a forgery on the string $\mathtt{label}_{\text{registration}} \| \mathsf{prepqpk}_{R,i} \| \mathsf{preecpk}_{R,i} \| \mathtt{label}_{\text{version}}$ under key idecvk_R.

Reduction $\mathcal{B}_4$ against EUF-CMA *security of* Σ: Reduction $\mathcal{B}_4$ receives as input a challenge signature public key pk^* and a signing oracle O, and must produce a forgery.

The reduction behaves similarly to Game 4, except as follows. For party R, it uses pk^* as the long-term identity key instead of generating a new one. Whenever party R needs to sign something, the reduction uses oracle O. Suppose the abort event of Game 4 occurs. Note that this implies that $\mathsf{RevLongTermKey}(R)$ is not called. The abort event implies that $\mathsf{PKB}_{R,i}.\sigma$ is a valid signature on $\mathtt{label}_{\text{registration}} \| \mathsf{prepqpk}_{R,i} \| \mathsf{preecpk}_{R,i} \| \mathtt{label}_{\text{version}}$ but this prekey bundle was never signed by party R during prekey bundle generation. Moreover, since signatures are domain separated with $\mathtt{label}_{\text{registration}}$, this string was never signed by party R during any other part of the protocol. Thus $\mathtt{label}_{\text{registration}} \| \mathsf{prepqpk}_{R,i} \| \mathsf{preecpk}_{R,i} \| \mathtt{label}_{\text{version}}$ was never queried to the signing oracle O, so the reduction can return this string and signature as a winning forgery in the EUF-CMA game for Σ. Thus

$$\mathsf{Adv}_3 \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\Sigma}(\mathcal{B}_4) + \mathsf{Adv}_4$$

Game 5 (Guess index of peer pre-key bundle). In this game, the challenger guesses in advance the index $i \in \{1, \dots, n_{\mathsf{B}}\}$ indicating which of R's prekey bundles will be used in the target session, and aborts if the guess is incorrect. Thus

$$\mathsf{Adv}_4 = n_{\mathsf{B}} \mathsf{Adv}_5$$

Branch A: elliptic curve security. The proof now branches into two cases: one based on the secrecy of ECDH shared secrets, one based on the secrecy of KEM shared secrets.

Game 6 (Undo distinct DH public keys for the test session). In this game, the challenger *does not abort* if the two DH public keys used in this session ($\mathsf{preecpk}_{R,i}$ and $\mathsf{rchecpk}_{S,s,1}$) are the same. This is required since a later proof step will substitute a DH challenge which could (with small but non-zero probability) have the two challenge public keys equal. Since the latter is honestly generated independently of the former and the size of the group is q, we have that

$$\mathsf{Adv}_5 = \frac{1}{q} + \mathsf{Adv}_6$$

Game 7 (Replace key ext_1 derived from DH shared secret with random). In this game, the challenger replaces the ext_1 value in the call to $\mathsf{KDFRKCK}$ with a random bitstring of the same length.

Reduction $\mathcal{B}_7$ against PRF-ODH security of the group with $\mathsf{HKDF.Extract}$: Reduction $\mathcal{B}_7$ receives as input a DH challenge $U = g^u, V = g^v$, and real-or-random value W, and PRF-ODH oracles OU and OV, and must return its guess of whether W was real or random.

The reduction behaves similarly to Game 7, except as follows. For $\mathsf{preecpk}_{R,i}$, it uses U; note $\mathsf{rev\_prekey}[R, i, \mathsf{ec}] = \mathsf{false}$ by the freshness condition. For $\mathsf{rchecpk}_{S,s,1}$, it uses V; note $\mathsf{rev\_rchkey}[S, s, 1, \mathsf{ec}] = \mathsf{false}$ by the freshness condition. For ext_1 in the call to $\mathsf{KDFRKCK}$ in $\pi_{S,s}$ stage $[\mathtt{asym}_0]$, use W; similarly in the matching session $\pi_{R,r}$ stage $[\mathtt{asym}_0]$ if it exists.

Since honest responder R will use each of its prekey bundles at most once, there is at most one session at R which uses $\mathsf{preecpk}_{R,i}$. If that session received a different value V' as the peer EC ratchet public key, then compute ext_1 in that session by calling PRF-ODH oracle OU with V and the salt value unchanged. If that session received V as the peer EC ratchet public key, then replace ext_1 consistently in that session as well. Either way, in that session, in the second half of SessionStartR for the computation of ecss', compute that second ext_1 by querying the PRF-ODH oracle OU with the received peer EC ratchet public key and the salt value unchanged.

Additionally, in the first execution of AsymRatchet for $\pi_{S,s}$, compute ext_1 in the first call to $\mathsf{KDFRKCK}$ by querying the PRF-ODH oracle OV with the received peer EC ratchet public key and the salt value unchanged.

Reduction $\mathcal{B}_7$ outputs as its answer to the PRF-ODH challenger the same b' output by $\mathcal{A}$. When W is real, $\mathcal{B}_7$ exactly simulates Game 6 to $\mathcal{A}$, whereas when W is random, it simulates Game 7 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_6 \leq \mathsf{Adv}^{\text{prf}_1-\text{odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_7) + \mathsf{Adv}_7$$

Game 8 (Replace key ext_2 derived from ext_1 with random). In this game, the challenger replaces the ext_2 value in the call to KDFRKCK with a random bitstring of the same length.

Reduction $\mathcal{B}_8$ against PRF security of HKDF.Extract *in its first argument*: Reduction $\mathcal{B}_8$ has access to an oracle O which either evaluates HKDF.Extract or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 8, except as follows. In $\pi_{S,s}$ stage $[\texttt{asym}_0]$ in the calculation of ext_2 inside KDFRKCK, it calls oracle O with the salt argument (either $salt = 0$ or $salt = ss_2$). If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_8$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_8$ exactly simulates Game 7 since ext_1 is random, whereas when O is random, it simulates Game 8 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_7 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Extract}}(\mathcal{B}_8) + \mathsf{Adv}_8$$

Game 9 (Replace key $z = (rk, ck)$ derived from ext_2 with random). In this game, the challenger replaces the $z = (rk, ck)$ value in the call to KDFRKCK with a random bitstring of the same length.

Reduction $\mathcal{B}_9$ against PRF security of HKDF.Expand *in its first argument*: Reduction $\mathcal{B}_9$ has access to an oracle O which either evaluates HKDF.Expand or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 9, except as follows. In $\pi_{S,s}$ stage $[\texttt{asym}_0]$ in the calculation of z inside KDFRKCK, it calls oracle O with the given label argument. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_9$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_9$ exactly simulates Game 8 since ext_2 is random, whereas when O is random, it simulates Game 9 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_8 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_9) + \mathsf{Adv}_9$$

Branch B: post-quantum security

Game 10 (Replace pqss with random value). In this game, the challenger replaces the pqss value with a random bitstring of the same length.

Reduction $\mathcal{B}_{10}$ against the IND-CCA security of KEM Π_1: Reduction $\mathcal{B}_{10}$ receives as input a challenge KEM public key pk^*, challenge ciphertext ct^*, and real-or-random value ss^*, and decapsulation oracle O[Dec], and must return its guess of whether ss^* was real or random.

The reduction behaves similarly to Game 10, except as follows. For $\mathsf{prepqpk}_{R,i}$, it uses pk^*; note $\mathsf{rev\_prekey}[R, i, \mathsf{pqsk}] = \mathsf{false}$ by the freshness condition. For $\mathsf{prepqct}_{S,s}$ it uses ct^*; note $\mathsf{rev\_prekey}[S, s, \mathsf{pqrnd}] = \mathsf{false}$ by the freshness condition. For pqss in $\pi_{S,s}$ stage $[\texttt{asym}_0]$, it uses ss^*.

Since honest responder R will use each of its prekey bundles at most once, there is at most one session at R which uses $\mathsf{prepqpk}_{R,i}$. If that session received a different value ct' as the peer PQ prekey ciphertext, then compute pqss in that session by calling the decapsulation oracle O[Dec] with ct'. If that session received ct^* as the peer PQ prekey ciphertext, then replace pqss in that session with ss^* as well.

Reduction $\mathcal{B}_{10}$ outputs as its answer to the IND-CCA challenger the same b' output by $\mathcal{A}$. When ss^* is real, $\mathcal{B}_{10}$ exactly simulates Game 5 to $\mathcal{A}$, whereas when ss^* is random, it simulates Game 10 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_9 \leq \mathsf{Adv}^{\mathrm{ind-cca}}_{\Pi_1}(\mathcal{B}_{10}) + \mathsf{Adv}_{10}$$

Game 11 (Replace output of KDFRKCK with random). In this game, the challenger replaces the (rk, ck) output of KDFRKCK with a random bitstring of the same length.

Reduction $\mathcal{B}_{11}$ against PRF security of KDFRKCK *in its third argument*: Reduction $\mathcal{B}_{11}$ has access to an oracle O which either evaluates KDFRKCK or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 11, except as follows. In $\pi_{S,s}$ stage $[\texttt{asym}_0]$ in the calculation of (rk, ck), it calls oracle O with inputs 0, ecss, and the session identifier. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_{11}$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_{11}$ exactly simulates Game 10 since pqss is random, whereas when O is random, it simulates Game 11 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_{10} \leq \mathsf{Adv}^{\mathrm{prf}_3}_{\mathsf{KDFRKCK}}(\mathcal{B}_{11}) + \mathsf{Adv}_{11}$$

Conclusion. As of Game 9 and Game 11, $\pi_{S,s}.rk[\mathtt{asym}_0]$ and $\pi_{S,s}.ck[\mathtt{out}_{0,0}]$ are indistinguishable from random. Thus, the probability that the adversary can distinguish $\pi_{S,s}.rk[\mathtt{asym}_0]$ and $\pi_{S,s}.ck[\mathtt{out}_{0,0}]$ from random when $\mathsf{valid}(S,s,[\mathtt{asym}_0]) \wedge \mathsf{unrev}(S,s,[\mathtt{asym}_0])$ holds is at most

$$\frac{1}{q}\binom{n_{\mathsf{P}} n_{\mathsf{B}} + n_{\mathsf{P}} n_{\mathsf{S}} n_{\mathsf{s}}}{2} + n_{\mathsf{P}}^2 n_{\mathsf{S}} \left(\begin{array}{l} \mathsf{Adv}^{\text{EUF-CMA}}_{\Sigma}(\mathcal{B}_4) \\ +n_{\mathsf{B}} \min \left\{ \begin{array}{l} \frac{1}{q} + \mathsf{Adv}^{\mathrm{prf}_1-\mathrm{odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_7) + \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Extract}}(\mathcal{B}_8) + \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_9), \\ \mathsf{Adv}^{\text{IND-CCA}}_{\Pi_1}(\mathcal{B}_{10}) + \mathsf{Adv}^{\mathrm{prf}_3}_{\mathsf{KDFRKCK}}(\mathcal{B}_{11}) \end{array} \right\} \end{array} \right)$$

□

5.4 Lemma 4: Initial key establishment for the responder

Lemma 4. *Consider session $\pi_{R,r}$ such that $\pi_{R,r}.\mathsf{role} = \mathtt{resp}$. Assume that Σ is* EUF-CMA*, that* HKDF.Expand *and* HKDF.*Extract are both PRFs in their first argument, and that* KDFRKCK *is a PRF in its third argument. Assume that either the PRF-ODH assumption holds for the elliptic curve group with* HKDF.Extract *in its first argument, or that Π_1 is an* IND-CCA*-secure KEM. If $\mathsf{valid}(R,r,[\mathtt{asym}_0]) \wedge \mathsf{unrev}(R,r,[\mathtt{asym}_0])$ holds, then $\pi_{R,r}.rk[\mathtt{asym}_0]$ and $\pi_{R,r}.ck[\mathtt{in}_{0,0}]$ are indistinguishable from random.*

Proof.

Game 0 (Original game). This game is the original multi-stage key indistinguishability security experiment for the PQ3 protocol. Thus

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\text{IND}}_{\Pi,n_{\mathsf{P}},n_{\mathsf{B}},n_{\mathsf{S}},n_{\mathsf{s}}}(\mathcal{A})$$

Game 1 (Require distinct honestly generated DH public keys). In this game, we ensure there is no collision on any honestly generated DH public keys. The challenger maintains a list of all DH private values generated during the game; if any DH private value appears twice, the challenger aborts the simulation and the adversary automatically wins. The maximum number of honestly generated DH key pairs is $n_{\mathsf{P}} n_{\mathsf{B}} + n_{\mathsf{P}} n_{\mathsf{S}} n_{\mathsf{s}}$ and thus

$$\mathsf{Adv}_0 \leq \frac{1}{q}\binom{n_{\mathsf{P}} n_{\mathsf{B}} + n_{\mathsf{P}} n_{\mathsf{S}} n_{\mathsf{s}}}{2} + \mathsf{Adv}_1$$

where q is the order of the group.

Game 2 (Guess index of target session). In this game, the challenger guesses in advance the index $(R,r,0) \in \{1,\ldots,n_{\mathsf{P}}\} \times \{1,\ldots,n_{\mathsf{S}}\} \times \{0\}$ of the session and stage against which the Test query is issued, and aborts if the guess is incorrect. Thus

$$\mathsf{Adv}_1 = n_{\mathsf{P}} n_{\mathsf{S}} \mathsf{Adv}_2$$

Game 3 (Guess index of peer pre-key bundle). In this game, the challenger guesses in advance the index $i \in \{1,\ldots,n_{\mathsf{B}}\}$ indicating which of R's prekey bundles will be used in the target session, and aborts if the guess is incorrect. Thus

$$\mathsf{Adv}_2 = n_{\mathsf{B}} \mathsf{Adv}_3$$

Game 4 (Guess matching session of target session). In this game, the challenger guesses in advance the index $(S,s,0) \in \{1,\ldots,n_{\mathsf{P}}\} \times \{1,\ldots,n_{\mathsf{S}}\} \times \{0\}$ of the matching session of the target session, and aborts if the guess is incorrect. Thus

$$\mathsf{Adv}_3 = n_{\mathsf{P}} n_{\mathsf{S}} \mathsf{Adv}_4$$

Game 5 (Abort if forged signature of incoming message). In this game, the challenger aborts if SessionStartR($\pi_{R,r},\ldots$) accepted the signature on the incoming message, but that message was never signed by party S, and $\pi_{R,r}.[\mathtt{asym}_0]$ accepted before S's long-term key was revealed. If the abort event occurs, then that signature constitutes a forgery on the incoming message under key idecvk_S.

Reduction $\mathcal{B}_5$ against EUF-CMA *security of Σ*: Reduction $\mathcal{B}_5$ receives as input a challenge signature public key pk^* and a signing oracle O, and must produce a forgery.

The reduction behaves similarly to Game 5, except as follows. For party S, it uses pk^* as the long-term identity key instead of generating a new one. Whenever party S needs to sign something, the reduction uses oracle O. Suppose the abort event of Game 5 occurs. Note that this implies that $\mathsf{RevLongTermKey}(S)$ is not called. The abort event implies that the signature σ is a valid signature on the incoming protocol message m

but this protocol message was never signed by party S. Thus, the protocol message was never queried to the signing oracle O, so the reduction can return this string and corresponding signature as a winning forgery in the EUF-CMA game for Σ. Thus

$$\mathsf{Adv}_4 \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\Sigma}(\mathcal{B}_5) + \mathsf{Adv}_5$$

Branch A: elliptic curve security. The proof now branches into two cases: one based on the secrecy of ECDH shared secrets, one based on the secrecy of KEM shared secrets.

Game 6 (Undo distinct DH public keys for the test session). In this game, the challenger *does not abort* if the two DH public keys used in this session ($\mathsf{preecpk}_{R,i}$ and $\mathsf{rchecpk}_{S,s,1}$) are the same. This is required since a later proof step will substitute a DH challenge which could (with small but non-zero probability) have the two challenge public keys equal. Since the former is honestly generated independently of the latter and the size of the group is q, we have that

$$\mathsf{Adv}_5 = \frac{1}{q} + \mathsf{Adv}_6$$

Game 7 (Replace key ext_1 derived from first DH shared secret with random). In this game, the challenger replaces the ext_1 value in the call to $\mathsf{KDFRKCK}$ involving ecss with a random bitstring of the same length.

Reduction $\mathcal{B}_7$ against PRF-ODH security of the group with $\mathsf{HKDF.Extract}$: Reduction $\mathcal{B}_7$ receives as input a DH challenge $U = g^u, V = g^v$, and real-or-random value W, and PRF-ODH oracles OU and OV, and must return its guess of whether W was real or random.

The reduction behaves similarly to Game 7, except as follows. For $\mathsf{preecpk}_{R,i}$, it uses U; note $\mathsf{rev\_prekey}[R, i, \mathsf{ec}] = \mathsf{false}$ by the freshness condition. For $\mathsf{rchecpk}_{S,s,1}$, it uses V; note $\mathsf{rev\_rchkey}[S, s, 1, \mathsf{ec}] = \mathsf{false}$ by the freshness condition. Note that there is a unique session at party S that uses $\mathsf{rchecpk}_{S,s,1}$: it exists because of Game 5, and it's unique by Game 1. For ext_1 in the call to $\mathsf{KDFRKCK}$ in $\pi_{R,r}$ stage $[\mathtt{asym}_0]$, use W; similarly in the matching session $\pi_{S,s}$ stage $[\mathtt{asym}_0]$ if it exists.

In the first execution of AsymRatchet for $\pi_{S,s}$, compute ext_1 in the first call to $\mathsf{KDFRKCK}$ by querying the PRF-ODH oracle OV with the received peer EC ratchet public key and the salt value unchanged.

Reduction $\mathcal{B}_7$ outputs as its answer to the PRF-ODH challenger the same b' output by $\mathcal{A}$. When W is real, $\mathcal{B}_7$ exactly simulates Game 6 to $\mathcal{A}$, whereas when W is random, it simulates Game 7 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_6 \leq \mathsf{Adv}^{\text{prf}_1-\text{odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_7) + \mathsf{Adv}_7$$

Game 8 (Replace key ext_2 derived from ext_1 with random). In this game, the challenger replaces the ext_2 value in the call to $\mathsf{KDFRKCK}$ involving ecss with a random bitstring of the same length.

Reduction $\mathcal{B}_8$ against PRF security of $\mathsf{HKDF.Extract}$ *in its first argument*: Reduction $\mathcal{B}_8$ has access to an oracle O which either evaluates $\mathsf{HKDF.Extract}$ or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 8, except as follows. In $\pi_{R,r}$ stage $[\mathtt{asym}_0]$ in the calculation of ext_2 inside $\mathsf{KDFRKCK}$, it calls oracle O with the salt argument (either $salt = 0$ or $salt = ss_2$). If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_8$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_8$ exactly simulates Game 7 since ext_1 is random, whereas when O is random, it simulates Game 8 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_7 \leq \mathsf{Adv}^{\text{prf}_1}_{\mathsf{HKDF.Extract}}(\mathcal{B}_8) + \mathsf{Adv}_8$$

Game 9 (Replace key $z = (rk, ck)$ derived from ext_2 with random). In this game, the challenger replaces the $z = (rk, ck)$ value in the call to $\mathsf{KDFRKCK}$ involving ecss with a random bitstring of the same length.

Reduction $\mathcal{B}_9$ against PRF security of $\mathsf{HKDF.Expand}$ *in its first argument*: Reduction $\mathcal{B}_9$ has access to an oracle O which either evaluates $\mathsf{HKDF.Expand}$ or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 9, except as follows. In $\pi_{R,r}$ stage $[\mathtt{asym}_0]$ in the calculation of z inside $\mathsf{KDFRKCK}$, it calls oracle O with the given label argument. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_9$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_9$ exactly simulates Game 8 since ext_2 is random, whereas when O is random, it simulates Game 9 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_8 \leq \mathsf{Adv}^{\text{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_9) + \mathsf{Adv}_9$$

Branch B: post-quantum security

Game 10 (Replace pqss with random value). In this game, the challenger replaces the pqss value with a random bitstring of the same length.

Reduction $\mathcal{B}_{10}$ against the IND-CCA security of KEM Π_1: Reduction $\mathcal{B}_{10}$ receives as input a challenge KEM public key pk^*, challenge ciphertext ct^*, and real-or-random value ss^*, and decapsulation oracle $\mathsf{O}[\mathsf{Dec}]$, and must return its guess of whether ss^* was real or random.

The reduction behaves similarly to Game 10, except as follows. For $\mathsf{prepqpk}_{R,i}$, it uses pk^*; note $\mathsf{rev\_prekey}[R, i, \mathsf{pqsk}] = \mathsf{false}$ by the freshness condition. For pqss in $\pi_{R,r}$ stage $[\mathtt{asym}_0]$, it uses ss^*.

In all honest initiator sessions that use $\mathsf{prepqpk}_{R,i}$ other than $\pi_{S,s}$, generate the PQ encapsulation honestly. For $\mathsf{prepqct}_{S,s}$ use ct^*; note $\mathsf{rev\_prekey}[S, s, \mathsf{pqrnd}] = \mathsf{false}$ by the freshness condition.

Reduction $\mathcal{B}_{10}$ outputs as its answer to the IND-CCA challenger the same b' output by $\mathcal{A}$. When ss^* is real, $\mathcal{B}_{10}$ exactly simulates Game 5 to $\mathcal{A}$, whereas when ss^* is random, it simulates Game 10 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_9 \leq \mathsf{Adv}^{\mathrm{ind-cca}}_{\Pi_1}(\mathcal{B}_{10}) + \mathsf{Adv}_{10}$$

Game 11 (Replace output of KDFRKCK with random). In this game, the challenger replaces the (rk, ck) output of KDFRKCK involving pqss with a random bitstring of the same length.

Reduction $\mathcal{B}_{11}$ against PRF security of KDFRKCK *in its third argument*: Reduction $\mathcal{B}_{11}$ has access to an oracle O which either evaluates KDFRKCK or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 11, except as follows. In $\pi_{R,r}$ stage $[\mathtt{asym}_0]$ in the calculation of (rk, ck), it calls oracle O with inputs 0, ecss, and the session identifier. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_{11}$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_{11}$ exactly simulates Game 10 since pqss is random, whereas when O is random, it simulates Game 11 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_{10} \leq \mathsf{Adv}^{\mathrm{prf_3}}_{\mathsf{KDFRKCK}}(\mathcal{B}_{11}) + \mathsf{Adv}_{11}$$

Conclusion. As of Game 9 and Game 11, $\pi_{R,r}.rk[\mathtt{asym}_0]$ and $\pi_{R,r}.ck[\mathtt{in}_{0,0}]$ are indistinguishable from random. Thus, the probability that the adversary can distinguish these from random when $\mathsf{valid}(R, r, [\mathtt{asym}_0]) \wedge \mathsf{unrev}(R, r, [\mathtt{asym}_0])$ holds is at most

$$\frac{1}{q}\binom{n_{\mathsf{P}} n_{\mathsf{B}} + n_{\mathsf{P}} n_{\mathsf{S}} n_{\mathsf{s}}}{2} + n_{\mathsf{P}}^2 n_{\mathsf{S}}^2 n_{\mathsf{B}} \left(\begin{array}{l} \mathsf{Adv}^{\mathrm{EUF\text{-}CMA}}_{\Sigma}(\mathcal{B}_5) \\ + \min \left\{ \begin{array}{l} \frac{1}{q} + \mathsf{Adv}^{\mathrm{prf_1-odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_7) + \mathsf{Adv}^{\mathrm{prf_1}}_{\mathsf{HKDF.Extract}}(\mathcal{B}_8) + \mathsf{Adv}^{\mathrm{prf_1}}_{\mathsf{HKDF.Expand}}(\mathcal{B}_9), \\ \mathsf{Adv}^{\mathrm{IND\text{-}CCA}}_{\Pi_1}(\mathcal{B}_{10}) + \mathsf{Adv}^{\mathrm{prf_3}}_{\mathsf{KDFRKCK}}(\mathcal{B}_{11}) \end{array} \right\} \end{array} \right)$$

□

5.5 Lemmas 5 and 6: Asymmetric ratchet

Recall that during the PQ3 asymmetric ratchet in Figure 6, the asymmetric ratchet actually advances two steps: one step for the incoming chain, one step for the outgoing chain. We break the proof up for the security of the asymmetric ratchet into two lemmas: Lemma 5 covering the first half of the asymmetric ratchet (deriving the next root key and the chain key for the incoming chain), and Lemma 6 covering the second half of the asymmetric ratchet (deriving the subsequent root key and the chain key for the outgoing chain).

Lemma 5. *Consider session $\pi_{P,p}$ stage $[\mathtt{asym}_j]$, where $[\mathtt{asym}_j]$ is the first of the two stages in AsymRatchet. (In other words, either $\pi_{P,p}.\mathsf{role} = \mathtt{init}$ and $j > 0$ odd, or $\pi_{P,p}.\mathsf{role} = \mathtt{resp}$ and $j > 0$ even.) Assume that Σ is* EUF-CMA*, that* HKDF.Expand *and* HKDF.*Extract are both PRFs in their first argument, and that* KDFRKCK *is a PRF in its first and third arguments. Assume that either the PRF-ODH assumption holds for the elliptic curve group with* HKDF.Extract *in its first argument, or that Π_2 is an* IND-CCA*-secure KEM. If $\mathsf{valid}(P, p, [\mathtt{asym}_j]) \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_j])$ holds, then $\pi_{P,p}.rk[\mathtt{asym}_j]$ and $\pi_{P,p}.ck[\mathtt{in}_{j,0}]$ are indistinguishable from random.*

Proof. Since $\mathsf{unrev}(P, p, [\mathtt{asym}_j])$ holds, then either $\mathsf{unrev}_{\mathsf{state}}(P, p, [\mathtt{asym}_j] \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_{j-1}])$ holds or $\big(\neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j+1, \mathsf{ec}) \wedge \neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{pqsk}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j, \mathsf{pqrnd})\big)$ holds.

Game 0 (Starting game). This game is the last game in the proof of Lemma 3 or Lemma 6 as appropriate. Define its advantage as Adv_0.

Game 1 (Abort if forged signature of incoming message). In this game, the challenger aborts if AsymRatchet accepted the signature on the incoming message, but that message was never signed by peer $P' = \pi_{P,p}.\mathsf{peerid}$, and $\pi_{P,p}.[\mathtt{asym}_j]$ accepted before P''s long-term key was revealed. If the abort event occurs, then that signature constitutes a forgery on the incoming message under key idecvk_S.

Reduction $\mathcal{B}_1$ against EUF-CMA *security of* Σ: Reduction $\mathcal{B}_1$ receives as input a challenge signature public key pk^* and a signing oracle O, and must produce a forgery.

The reduction behaves similarly to Game 1, except as follows. For party P', it uses pk^* as the long-term identity key instead of generating a new one. Whenever party P' needs to sign something, the reduction uses oracle O. Suppose the abort event of Game 1 occurs. Note that this implies that $\mathsf{RevLongTermKey}(P')$ is not called. The abort event implies that the signature σ is a valid signature on the incoming protocol message m but this protocol message was never signed by party P'. Thus, the protocol message was never queried to the signing oracle O, so the reduction can return this string and corresponding signature as a winning forgery in the EUF-CMA game for Σ. Thus

$$\mathsf{Adv}_0 \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\Sigma}(\mathcal{B}_1) + \mathsf{Adv}_1$$

Case A: Suppose first that $\mathsf{unrev}_{\mathsf{state}}(P, p, [\mathtt{asym}_j] \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_{j-1}])$ holds. Note that $\mathsf{valid}(P, p, [\mathtt{asym}_{j-1}])$ also holds. By Lemma 3 or Lemma 6, $\pi_{P,p}.rk[\mathtt{asym}_{j-1}]$ is indistinguishable from random.

Game 2 (Replace output of KDFRKCK with random). In this game, the challenger replaces the (rk, ck) output of the first call to KDFRKCK in AsymRatchet for $\pi_{P,p}$ stage $[\mathtt{asym}_j]$ with a random bitstring of the same length.

Reduction $\mathcal{B}_2$ against PRF security of KDFRKCK *in its first argument*: Reduction $\mathcal{B}_2$ has access to an oracle O which either evaluates KDFRKCK or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 2, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_j]$ in the calculation of (rk, ck), it calls oracle O with inputs ecss, pqss, and the session identifier $\pi_{P,p}.\mathsf{sid}[\mathtt{asym}_j]$. If the matching session exists at the peer, then it does the same there. Note that this stage (and the matching session at the peer, if it exists) is the only occurrence of this session identifier $\pi_{P,p}.\mathsf{sid}[\mathtt{asym}_j]$. It is unique at the session owner due to Lemma 2, and similarly at the honest peer. Reduction $\mathcal{B}_2$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_2$ exactly simulates Game 1 since $\pi_{P,p}.rk[\mathtt{asym}_{j-1}]$ is random and unrevealed, whereas when O is random, it simulates Game 2 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_1 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{KDFRKCK}}(\mathcal{B}_2) + \mathsf{Adv}_2$$

Case B: Now suppose that $\big(\neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j+1, \mathsf{ec}) \wedge \neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{pqsk}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j, \mathsf{pqrnd})\big)$ holds.

Branch A: elliptic curve security. The proof now branches into two cases: one based on the secrecy of ECDH shared secrets, one based on the secrecy of KEM shared secrets.

Game 3 (Undo distinct DH public keys for the test session). In this game, the challenger *does not abort* if the two DH public keys used in this stage ($\mathsf{rchecpk}_{P',p',j+1}$ and $\mathsf{rchecpk}_{P,p,j}$) are the same. This is required since a later proof step will substitute a DH challenge which could (with small but non-zero probability) have the two challenge public keys equal. Since the former is honestly generated independently of the latter and the size of the group is q, we have that

$$\mathsf{Adv}_2 = \frac{1}{q} + \mathsf{Adv}_3$$

Game 4 (Replace key ext_1 derived from first DH shared secret with random). In this game, the challenger replaces the ext_1 value in the call to KDFRKCK involving ecss with a random bitstring of the same length.

Reduction $\mathcal{B}_4$ against PRF-ODH security of the group with HKDF.Extract: Reduction $\mathcal{B}_4$ receives as input a DH challenge $U = g^u, V = g^v$, and real-or-random value W, and PRF-ODH oracles OU and OV, and must return its guess of whether W was real or random.

The reduction behaves similarly to Game 4, except as follows. For $\mathsf{rchecpk}_{P',p',j+1}$, it uses U; note $\mathsf{rev\_rchkey}[P', p', j{+}1, \mathsf{ec}] = \mathsf{false}$ by the freshness condition. For $\mathsf{rchecpk}_{P,p,j}$, it uses V; note $\mathsf{rev\_rchkey}[P, p, j, \mathsf{ec}] = \mathsf{false}$ by the freshness condition. Note that there is a unique session at party P that uses $\mathsf{rchecpk}_{P,p,j}$: it exists because of Game 1, and it's unique by Game 1. For ext_1 in the call to KDFRKCK in $\pi_{P,p}$ stage $[\mathtt{asym}_j]$, use W; similarly in the matching session $\pi_{P',p'}$ stage $[\mathtt{asym}_{j+1}]$ if it exists.

In the previous execution of AsymRatchet for $\pi_{P,p}$, compute ext_1 in the second call to KDFRKCK by querying the PRF-ODH oracle OV with the received peer EC ratchet public key and the salt value unchanged.

Reduction $\mathcal{B}_4$ outputs as its answer to the PRF-ODH challenger the same b' output by $\mathcal{A}$. When W is real, $\mathcal{B}_4$ exactly simulates Game 3 to $\mathcal{A}$, whereas when W is random, it simulates Game 4 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_3 \leq \mathsf{Adv}^{\mathrm{prf}_1-\mathrm{odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_4) + \mathsf{Adv}_4$$

Game 5 (Replace key ext_2 derived from ext_1 with random). In this game, the challenger replaces the ext_2 value in the call to KDFRKCK involving ecss with a random bitstring of the same length.

Reduction $\mathcal{B}_5$ against PRF security of HKDF.Extract *in its first argument*: Reduction $\mathcal{B}_5$ has access to an oracle O which either evaluates HKDF.Extract or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 5, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_j]$ in the calculation of ext_2 inside KDFRKCK, it calls oracle O with the salt argument (either $salt = 0$ or $salt = ss_2$). If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_5$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_5$ exactly simulates Game 4 since ext_1 is random, whereas when O is random, it simulates Game 5 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_4 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Extract}}(\mathcal{B}_5) + \mathsf{Adv}_5$$

Game 6 (Replace key $z = (rk, ck)$ derived from ext_2 with random). In this game, the challenger replaces the $z = (rk, ck)$ value in the call to KDFRKCK involving ecss with a random bitstring of the same length.

Reduction $\mathcal{B}_6$ against PRF security of HKDF.Expand *in its first argument*: Reduction $\mathcal{B}_6$ has access to an oracle O which either evaluates HKDF.Expand or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 6, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_j]$ in the calculation of z inside KDFRKCK, it calls oracle O with the given label argument. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_6$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_6$ exactly simulates Game 5 since ext_2 is random, whereas when O is random, it simulates Game 6 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_5 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_6) + \mathsf{Adv}_6$$

Branch B: post-quantum security

Game 7 (Replace pqss with random value). In this game, the challenger replaces the pqss value with a random bitstring of the same length.

Reduction $\mathcal{B}_7$ against the IND-CCA security of KEM Π_2: Reduction $\mathcal{B}_7$ receives as input a challenge KEM public key pk^*, challenge ciphertext ct^*, and real-or-random value ss^*, and decapsulation oracle O[Dec], and must return its guess of whether ss^* was real or random.

The reduction behaves similarly to Game 7, except as follows. For $\mathsf{rchpqpk}_{P,p,j}$, it uses pk^*; note $\mathsf{rev\_rchkey}[P, p, j, \mathsf{pqsk}] = \mathsf{false}$ by the freshness condition. For pqss in $\pi_{P,p}$ stage $[\mathtt{asym}_j]$, it uses ss^*.

In all honest sessions at the peer that use $\mathsf{rchpqpk}_{P,p,j}$ other than the matching session, generate the PQ encapsulation honestly. For $\mathsf{rchpqct}_{P',p',j}$ use ct^*; note $\mathsf{rev\_rchkey}[P', p', j, \mathsf{pqrnd}] = \mathsf{false}$ by the freshness condition.

Reduction $\mathcal{B}_7$ outputs as its answer to the IND-CCA challenger the same b' output by $\mathcal{A}$. When ss^* is real, $\mathcal{B}_7$ exactly simulates Game 1 to $\mathcal{A}$, whereas when ss^* is random, it simulates Game 7 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_6 \leq \mathsf{Adv}^{\mathrm{ind}-\mathrm{cca}}_{\Pi_2}(\mathcal{B}_7) + \mathsf{Adv}_7$$

Game 8 (Replace output of KDFRKCK with random). In this game, the challenger replaces the (rk, ck) output of KDFRKCK involving pqss with a random bitstring of the same length.

Reduction $\mathcal{B}_8$ against PRF security of KDFRKCK *in its third argument*: Reduction $\mathcal{B}_8$ has access to an oracle O which either evaluates KDFRKCK or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 8, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_j]$ in the calculation of (rk, ck), it calls oracle O with inputs $\pi_{P,p}.rk[\mathtt{asym}_{j-1}]$, ecss, and the session identifier. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_8$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_8$ exactly simulates Game 7 since pqss is random, whereas when O is random, it simulates Game 8 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_7 \leq \mathsf{Adv}^{\mathrm{prf}_3}_{\mathsf{KDFRKCK}}(\mathcal{B}_8) + \mathsf{Adv}_8$$

Conclusion. As of Game 6 and Game 8, $\pi_{P,p}.rk[\mathtt{asym}_j]$ and $\pi_{P,p}.ck[\mathtt{in}_{j,0}]$ are indistinguishable from random. Thus, the probability that the adversary can distinguish these from random when $\mathsf{valid}(P, p, [\mathtt{asym}_j]) \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_j])$ holds is at most

$$\begin{aligned}&\mathsf{Adv}^{\text{EUF-CMA}}_{\Sigma}(\mathcal{B}_1)\\ &+\max\left\{\mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{KDFRKCK}}(\mathcal{B}_2), \min\left\{\begin{array}{l}\frac{1}{q} + \mathsf{Adv}^{\mathrm{prf}_1-\mathrm{odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_4) + \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Extract}}(\mathcal{B}_5) + \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_6),\\ \mathsf{Adv}^{\text{IND-CCA}}_{\Pi_2}(\mathcal{B}_7) + \mathsf{Adv}^{\mathrm{prf}_3}_{\mathsf{KDFRKCK}}(\mathcal{B}_8)\end{array}\right\}\right\}\end{aligned}$$

□

Lemma 6. *Consider session $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$, where $[\mathtt{asym}_{j+1}]$ is the second of the two stages in AsymRatchet. (In other words, either $\pi_{P,p}.\mathsf{role} = \mathtt{init}$ and $j > 0$ odd, or $\pi_{P,p}.\mathsf{role} = \mathtt{resp}$ and $j > 0$ even.) Assume that Σ is* EUF-CMA*, that* HKDF.Expand *and* HKDF.*Extract are both PRFs in their first argument, and that* KDFRKCK *is a PRF in its first and third arguments. Assume that either the PRF-ODH assumption holds for the elliptic curve group with* HKDF.Extract *in its first argument, or that Π_2 is an* IND-CCA*-secure KEM. If $\mathsf{valid}(P, p, [\mathtt{asym}_{j+1}]) \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_{j+1}])$ holds, then $\pi_{P,p}.rk[\mathtt{asym}_{j+1}]$ and $\pi_{P,p}.ck[\mathtt{out}_{j+1,0}]$ are indistinguishable from random.*

Proof. Since $\mathsf{unrev}(P, p, [\mathtt{asym}_j])$ holds, then either $\mathsf{unrev}_{\mathsf{state}}(P, p, [\mathtt{asym}_j] \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_{j-1}])$ holds or $\big(\neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j+1, \mathsf{ec}) \wedge \neg\mathsf{rev\_rchkey}[P, p, j, \mathsf{pqsk}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P, p, j, \mathsf{pqrnd})\big)$ holds.

Game 0 (Starting game). This game is the last game in the proof of Lemma 4 or Lemma 5 as appropriate. Define its advantage as Adv_0.

Case A: Suppose first that $\mathsf{unrev}_{\mathsf{state}}(P, p, [\mathtt{asym}_{j+1}] \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_j])$ holds. Note that $\mathsf{valid}(P, p, [\mathtt{asym}_j])$ also holds. By Lemma 4 or Lemma 5, $\pi_{P,p}.rk[\mathtt{asym}_j]$ is indistinguishable from random.

Game 1 (Replace output of KDFRKCK with random). In this game, the challenger replaces the (rk, ck) output of the second call to KDFRKCK in AsymRatchet for $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$ with a random bitstring of the same length.

Reduction $\mathcal{B}_1$ against PRF security of KDFRKCK *in its first argument*: Reduction $\mathcal{B}_1$ has access to an oracle O which either evaluates KDFRKCK or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 1, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$ in the calculation of (rk, ck), it calls oracle O with inputs ecss′, pqss′, and the session identifier $\pi_{P,p}.\mathsf{sid}[\mathtt{asym}_{j+1}]$. If the matching session exists at the peer, then it does the same there. Note that this stage (and the matching session at the peer, if it exists) is the only occurrence of this session identifier $\pi_{P,p}.\mathsf{sid}[\mathtt{asym}_{j+1}]$. It is unique at the session owner due to Lemma 2, and similarly at the honest peer. Reduction $\mathcal{B}_1$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_1$ exactly simulates Game 0 since $\pi_{P,p}.rk[\mathtt{asym}_j]$ is random and unrevealed, whereas when O is random, it simulates Game 1 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_0 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{KDFRKCK}}(\mathcal{B}_1) + \mathsf{Adv}_1$$

Case B: Now suppose that $\big(\neg\mathsf{rev\_rchkey}[P,p,j+2,\mathsf{ec}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P,p,j+1,\mathsf{ec}) \wedge \neg\mathsf{rev\_rchkey}[P,p,j+1,\mathsf{pqrnd}] \wedge \mathsf{unrev}_{\mathsf{peer}}(P,p,j+1,\mathsf{pqsk})\big)$ holds.

Branch A: elliptic curve security. The proof now branches into two cases: one based on the secrecy of ECDH shared secrets, one based on the secrecy of KEM shared secrets.

Game 2 (Undo distinct DH public keys for the test session). In this game, the challenger *does not abort* if the two DH public keys used in this stage ($\mathsf{rchecpk}_{P',p',j+1}$ and $\mathsf{rchecpk}_{P,p,j+2}$) are the same. This is required since a later proof step will substitute a DH challenge which could (with small but non-zero probability) have the two challenge public keys equal. Since the former is honestly generated independently of the latter and the size of the group is q, we have that

$$\mathsf{Adv}_1 = \frac{1}{q} + \mathsf{Adv}_2$$

Game 3 (Replace key ext_1 derived from first DH shared secret with random). In this game, the challenger replaces the ext_1 value in the call to KDFRKCK involving ecss' with a random bitstring of the same length.

Reduction $\mathcal{B}_3$ against PRF-ODH security of the group with HKDF.Extract: Reduction $\mathcal{B}_3$ receives as input a DH challenge $U = g^u, V = g^v$, and real-or-random value W, and PRF-ODH oracles OU and OV, and must return its guess of whether W was real or random.

The reduction behaves similarly to Game 3, except as follows. For $\mathsf{rchecpk}_{P',p',j+1}$, it uses U; note $\mathsf{rev\_rchkey}[P',p',j+1,\mathsf{ec}] = \mathsf{false}$ by the freshness condition. For $\mathsf{rchecpk}_{P,P,j+1}$, it uses V; note $\mathsf{rev\_rchkey}[P,p,j+2,\mathsf{ec}] = \mathsf{false}$ by the freshness condition. Note that there is a unique session at party P that uses $\mathsf{rchecpk}_{P,p,j+2}$: it exists because of Game 1, and it's unique by Game 1. For ext_1 in the call to KDFRKCK in $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$, use W; similarly in the matching session $\pi_{P',p'}$ stage $[\mathtt{asym}_{j+1}]$ if it exists.

In the next execution of AsymRatchet for $\pi_{P,p}$, compute ext_1 in the first call to KDFRKCK by querying the PRF-ODH oracle OV with the received peer EC ratchet public key and the salt value unchanged.

Reduction $\mathcal{B}_3$ outputs as its answer to the PRF-ODH challenger the same b' output by $\mathcal{A}$. When W is real, $\mathcal{B}_3$ exactly simulates Game 2 to $\mathcal{A}$, whereas when W is random, it simulates Game 3 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_2 \leq \mathsf{Adv}^{\mathrm{prf}_1-\mathrm{odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_3) + \mathsf{Adv}_3$$

Game 4 (Replace key ext_2 derived from ext_1 with random). In this game, the challenger replaces the ext_2 value in the call to KDFRKCK involving ecss' with a random bitstring of the same length.

Reduction $\mathcal{B}_4$ against PRF security of HKDF.Extract *in its first argument*: Reduction $\mathcal{B}_4$ has access to an oracle O which either evaluates HKDF.Extract or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 4, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$ in the calculation of ext_2 inside KDFRKCK, it calls oracle O with the salt argument (either $salt = 0$ or $salt = ss_2$). If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_4$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_4$ exactly simulates Game 3 since ext_1 is random, whereas when O is random, it simulates Game 4 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_3 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Extract}}(\mathcal{B}_4) + \mathsf{Adv}_4$$

Game 5 (Replace key $z = (rk, ck)$ derived from ext_2 with random). In this game, the challenger replaces the $z = (rk, ck)$ value in the call to KDFRKCK involving ecss' with a random bitstring of the same length.

Reduction $\mathcal{B}_5$ against PRF security of HKDF.Expand *in its first argument*: Reduction $\mathcal{B}_5$ has access to an oracle O which either evaluates HKDF.Expand or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 5, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$ in the calculation of z inside KDFRKCK, it calls oracle O with the given label argument. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_5$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_5$ exactly simulates Game 4 since ext_2 is random, whereas when O is random, it simulates Game 5 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_4 \leq \mathsf{Adv}^{\mathrm{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_5) + \mathsf{Adv}_5$$

Branch B: post-quantum security

Game 6 (Replace pqss' with random value). In this game, the challenger replaces the pqss' value with a random bitstring of the same length.

Reduction $\mathcal{B}_6$ against the IND-CCA security of KEM Π_2: Reduction $\mathcal{B}_6$ receives as input a challenge KEM public key pk^*, challenge ciphertext ct^*, and real-or-random value ss^*, and decapsulation oracle $\mathsf{O}[\mathsf{Dec}]$, and must return its guess of whether ss^* was real or random.

The reduction behaves similarly to Game 6, except as follows. For $\mathsf{rchpqpk}_{P',p',j+1}$, it uses pk^*; note $\mathsf{rev\_rchkey}[P', p', j+1, \mathsf{pqsk}] = \mathsf{false}$ by the freshness condition. For pqss in $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$, it uses ss^*.

In all other honest sessions that use $\mathsf{rchpqpk}_{P',p',j+1}$, generate the PQ encapsulation honestly. For $\mathsf{rchpqct}_{P,p,j+1}$ use ct^*; note $\mathsf{rev\_rchkey}[P, p, j+1, \mathsf{pqrnd}] = \mathsf{false}$ by the freshness condition.

Reduction $\mathcal{B}_6$ outputs as its answer to the IND-CCA challenger the same b' output by $\mathcal{A}$. When ss^* is real, $\mathcal{B}_6$ exactly simulates Game 0 to $\mathcal{A}$, whereas when ss^* is random, it simulates Game 6 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_5 \leq \mathsf{Adv}^{\mathrm{ind-cca}}_{\Pi_2}(\mathcal{B}_6) + \mathsf{Adv}_6$$

Game 7 (Replace output of KDFRKCK with random). In this game, the challenger replaces the (rk, ck) output of KDFRKCK involving pqss' with a random bitstring of the same length.

Reduction $\mathcal{B}_7$ against PRF security of KDFRKCK *in its third argument*: Reduction $\mathcal{B}_7$ has access to an oracle O which either evaluates KDFRKCK or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 7, except as follows. In $\pi_{P,p}$ stage $[\mathtt{asym}_{j+1}]$ in the calculation of (rk, ck), it calls oracle O with inputs $\pi_{P,p}.rk[\mathtt{asym}_j]$, ecss', and the session identifier. If the matching session exists at the peer, then it does the same there. Reduction $\mathcal{B}_7$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_7$ exactly simulates Game 6 since pqss' is random, whereas when O is random, it simulates Game 7 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_6 \leq \mathsf{Adv}^{\mathrm{prf_3}}_{\mathsf{KDFRKCK}}(\mathcal{B}_7) + \mathsf{Adv}_7$$

Conclusion. As of Game 5 and Game 7, $\pi_{P,p}.rk[\mathtt{asym}_{j+1}]$ and $\pi_{P,p}.ck[\mathtt{out}_{j+1,0}]$ are indistinguishable from random. Thus, the probability that the adversary can distinguish these from random when $\mathsf{valid}(P, p, [\mathtt{asym}_{j+1}]) \wedge \mathsf{unrev}(P, p, [\mathtt{asym}_{j+1}])$ holds is at most

$$\max\left\{\mathsf{Adv}^{\mathrm{prf_1}}_{\mathsf{KDFRKCK}}(\mathcal{B}_1), \min\left\{\begin{array}{l}\frac{1}{q} + \mathsf{Adv}^{\mathrm{prf_1-odh}}_{g,\mathsf{HKDF.Extract}}(\mathcal{B}_3) + \mathsf{Adv}^{\mathrm{prf_1}}_{\mathsf{HKDF.Extract}}(\mathcal{B}_4) + \mathsf{Adv}^{\mathrm{prf_1}}_{\mathsf{HKDF.Expand}}(\mathcal{B}_5), \\ \mathsf{Adv}^{\text{IND-CCA}}_{\Pi_2}(\mathcal{B}_6) + \mathsf{Adv}^{\mathrm{prf_3}}_{\mathsf{KDFRKCK}}(\mathcal{B}_7)\end{array}\right\}\right\}$$

□

5.6 Lemmas 7 and 8: Symmetric ratchet

Lemma 7. *Consider session $\pi_{P,p}$ stage $[chain_{j,0}]$ with chain $\in \{\mathsf{in}, \mathsf{out}\}$. Assume that* HKDF.Expand *is a PRF in its first argument. If $\mathsf{valid}(P, p, [chain_{j,0}]) \wedge \mathsf{unrev}(P, p, [chain_{j,0}])$ holds, then $\pi_{P,p}.mk[chain_{j,0}]$ and $\pi_{P,p}.ck[chain_{j,1}]$ are indistinguishable from random.*

Proof. Since $\mathsf{unrev}(P, p, [chain_{j,0}])$ holds, then $\mathsf{unrev}(P, p, [\mathtt{asym}_j])$ holds as well. By Lemma 5 or Lemma 6, $\pi_{P,p}.ck[chain_{j,0}]$ is indistinguishable from random.

Since $\mathsf{unrev}_{\mathsf{state}}(P, p, chain[j, 0])$, there has been no reveal of intermediate key $\pi_{P,p}.ck[chain_{j,0}]$ (including at the partner session, if it exists). Thus $\pi_{P,p}.ck[chain_{j,0}]$ remains unknown to the adversary.

Game 0 (Starting game). This game is the last game in the proof of Lemma 5 or Lemma 6 as appropriate. Define its advantage as Adv_0.

Game 1 (Replace output of HKDF.Expand with random.). In this game, the challenger replaces the values $\pi_{P,p}.mk[chain_{j,0}]$ and $\pi_{P,p}.ck[chain_{j,1}]$ with random bitstrings of the same length.

Reduction $\mathcal{B}_1$ against the PRF security of HKDF.Expand *in its first argument*: Reduction $\mathcal{B}_1$ has access to an oracle O which either evaluates HKDF.Expand or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 0, except as follows. In the execution of SymRatchet for $\pi_{P,p}$ stage $[chain_{j,0}]$, it sets $\pi_{P,p}.mk[chain_{j,0}]$ by querying its oracle O on $\texttt{label}_{\text{mkderivation}}$ and it sets $\pi_{P,p}.ck[chain_{j,1}]$ by querying its oracle O on $\texttt{label}_{\text{ckderivation}}$.

Reduction $\mathcal{B}_1$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_1$ exactly simulates Game 0 since $\pi_{P,p}.ck[chain_{j,0}]$ is random, whereas when O is random, it simulates Game 1 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_0 \leq \mathsf{Adv}^{\text{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_1) + \mathsf{Adv}_1$$

Conclusion. As of Game 1, $\pi_{P,p}.mk[chain_{j,0}]$ and $\pi_{P,p}.ck[chain_{j,1}]$ are indistinguishable from random. Thus, the probability that the adversary can distinguish these from random when $\mathsf{valid}(P, p, [chain_{j,0}]) \wedge \mathsf{unrev}(P, p, [chain_{j,0}])$ holds is at most the upper-bound found in Lemma 5 or Lemma 6 plus

$$\mathsf{Adv}^{\text{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_1)$$

□

Lemma 8. *Consider session $\pi_{P,p}$ stage $[chain_{j,\ell}]$ with chainin$\{\mathsf{in}, \mathsf{out}\}$ and $\ell > 0$. Assume that* HKDF.Expand *is a PRF in its first argument. If* $\mathsf{valid}(P, p, [chain_{j,\ell}]) \wedge \mathsf{unrev}(P, p, [chain_{j,\ell}])$ *holds, then $\pi_{P,p}.mk[chain_{j,\ell}]$ and $\pi_{P,p}.ck[chain_{j,\ell+1}]$ are indistinguishable from random.*

Proof. Since $\mathsf{unrev}(P, p, [chain_{j,\ell}])$ holds, then $\mathsf{unrev}(P, p, [chain_{j,\ell-1}])$ holds as well. By Lemma 7, $\pi_{P,p}.ck[chain_{j,1}]$ is indistinguishable from random.

Since $\mathsf{unrev}_{\mathsf{state}}(P, p, chain[j, \ell])$, there has been no reveal of intermediate key $\pi_{P,p}.ck[chain_{j,\ell}]$ (including at the partner session, if it exists). Thus $\pi_{P,p}.ck[chain_{j,\ell}]$ remains unknown to the adversary.

Game 0 (Starting game). This game is the last game in the proof of Lemma 7. Define its advantage as Adv_0.

Game 1 (Replace output of HKDF.Expand with random.). In this game, the challenger replaces the values $\pi_{P,p}.mk[chain_{j,\ell}]$ and $\pi_{P,p}.ck[chain_{j,\ell+1}]$ with random bitstrings of the same length.

Reduction $\mathcal{B}_1$ against the PRF security of HKDF.Expand *in its first argument*: Reduction $\mathcal{B}_1$ has access to an oracle O which either evaluates HKDF.Expand or is a random function, and must return its guess as to whether the oracle is real or random.

The reduction behaves similarly to Game 0, except as follows. In the execution of SymRatchet for $\pi_{P,p}$ stage $[chain_{j,\ell}]$, it sets $\pi_{P,p}.mk[chain_{j,\ell}]$ by querying its oracle O on $\texttt{label}_{\text{mkderivation}}$ and it sets $\pi_{P,p}.ck[chain_{j,\ell+1}]$ by querying its oracle O on $\texttt{label}_{\text{ckderivation}}$.

Reduction $\mathcal{B}_1$ outputs as its answer to the PRF challenge the same b' output by $\mathcal{A}$. When O is real, reduction $\mathcal{B}_1$ exactly simulates Game 0 since $\pi_{P,p}.ck[chain_{j,\ell}]$ is random, whereas when O is random, it simulates Game 1 to $\mathcal{A}$. Thus

$$\mathsf{Adv}_0 \leq \mathsf{Adv}^{\text{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_1) + \mathsf{Adv}_1$$

Conclusion. As of Game 1, $\pi_{P,p}.mk[chain_{j,\ell}]$ and $\pi_{P,p}.ck[chain_{j,\ell+1}]$ are indistinguishable from random. Thus, the probability that the adversary can distinguish these from random when $\mathsf{valid}(P, p, [chain_{j,\ell}]) \wedge \mathsf{unrev}(P, p, [chain_{j,\ell}])$ holds is at most the upper-bound found in Lemma 7 plus

$$\mathsf{Adv}^{\text{prf}_1}_{\mathsf{HKDF.Expand}}(\mathcal{B}_1)$$

□

References

[ACD19] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_5.

[ACJM20] Joël Alwen, Sandro Coretti, Daniel Jost, and Marta Mularczyk. Continuous group key agreement with active security. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 261–290. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64378-2_10.

[App23] Apple Security Engineering and Architecture. Advancing iMessage security: iMessage contact key verification, October 2023. URL: https://security.apple.com/blog/imessage-contact-key-verification/.

[App24] Apple Security Engineering and Architecture. iMessage with PQ3: The new state of the art in quantum-secure messaging at scale, February 2024. URL: https://security.apple.com/blog/imessage-pq3/.

[BBL+23] Olivier Blazy, Ioana Boureanu, Pascal Lafourcade, Cristina Onete, and Léo Robert. How fast do you heal? a taxonomy for post-compromise security in secure-channel establishment. In *Proceedings of the 32nd USENIX Conference on Security Symposium*, 2023.

[BBR+23] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. RFC 9420, July 2023. URL: https://www.rfc-editor.org/info/rfc9420, doi:10.17487/RFC9420.

[BFG+20] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. Towards post-quantum security for Signal's X3DH handshake. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 404–430. Springer, Heidelberg, October 2020. doi:10.1007/978-3-030-81652-0_16.

[BFG+22a] Alexander Bienstock, Jaiden Fairoze, Sanjam Garg, Pratyay Mukherjee, and Srinivasan Raghuraman. A more complete analysis of the Signal double ratchet algorithm. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 784–813. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15802-5_27.

[BFG+22b] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 3–34. Springer, Heidelberg, March 2022. doi:10.1007/978-3-030-97131-1_1.

[BFGJ17] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 651–681. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63697-9_22.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-Record communication, or, why not to use PGP. In *Proc. 2004 ACM Workshop on Privacy in the Electronic Society (WPES)*, page 77–84. ACM, October 2004. doi:10.1145/1029179.1029200.

[BJK23] Karthikean Barghavan, Charlie Jacomme, and Franziskus Kiefer. Formal analysis of the PQXDH protocol, December 2023. URL: https://github.com/Inria-Prosecco/pqxdh-analysis.

[BMS19] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, second edition, 2019. doi:10.1007/978-3-662-58146-9.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48329-2_21.

[CCD+20] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, October 2020. doi:10.1007/s00145-020-09360-1.

[CCG16] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In Michael Hicks and Boris Köpf, editors, *CSF 2016 Computer Security Foundations Symposium*, pages 164–178. IEEE Computer Society Press, 2016. doi:10.1109/CSF.2016.19.

[CGCD+17] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. In *Proc. IEEE European Symposium on Security and Privacy (EuroS&P) 2017*. IEEE, April 2017. doi:10.1109/EuroSP.2017.27.

[CJSV22] Ran Canetti, Palak Jain, Marika Swanberg, and Mayank Varia. Universally composable end-to-end secure messaging. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15979-4_1.

[DFGS15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1197–1210. ACM Press, October 2015. doi:10.1145/2810103.2813653.

[DG22] Samuel Dobson and Steven D. Galbraith. Post-quantum Signal key agreement from SIDH. In Jung Hee Cheon and Thomas Johansson, editors, *PQCRYPTO 2022*, volume 13512 of *LNCS*, pages 422–450. Springer, 2022. doi:10.1007/978-3-031-17234-2_20.

[DGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 400–409. ACM Press, October / November 2006. doi:10.1145/1180405.1180454.

[DH23] Benjamin Dowling and Britta Hale. Authenticated continuous key agreement: Active MitM detection and prevention. Cryptology ePrint Archive, Report 2023/228, 2023. https://eprint.iacr.org/2023/228.

[DHRR22] Benjamin Dowling, Eduard Hauck, Doreen Riepel, and Paul Rösler. Strongly anonymous ratcheted key exchange. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 119–150. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22969-5_5.

[DV19] F. Betül Durak and Serge Vaudenay. Bidirectional asynchronous ratcheted key agreement with linear complexity. In Nuttapong Attrapadung and Takeshi Yagi, editors, *IWSEC 19*, volume 11689 of *LNCS*, pages 343–362. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26834-3_20.

[FG14] Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1193–1204. ACM Press, November 2014. doi:10.1145/2660267.2660308.

[FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36362-7_17.

[Gün90] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 29–37. Springer, Heidelberg, April 1990. doi:10.1007/3-540-46885-4_5.

[HBD+22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[HKKP21] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for Signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 410–440. Springer, Heidelberg, May 2021. doi:10.1007/978-3-030-75248-4_15.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 273–293. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_17.

[JMM19] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_6.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_34.

[KS23] Ehren Kret and Rolve Schmidt. The PQXDH key agreement protocol, October 2023. URL: https://signal.org/docs/specifications/pqxdh/.

[KSH23] Panos Kampanakis, Douglas Stebila, and Torben Hansen. Post-quantum hybrid key exchange in SSH. https://www.ietf.org/archive/id/draft-kampanakis-curdle-ssh-pq-ke-01.txt, April 2023. Internet-Draft.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[MCYR17] Kevin Milner, Cas Cremers, Jiangshan Yu, and Mark Ryan. Automatically detecting the misuse of secrets: Foundations, design principles, and applications. In Boris Köpf and Steve Chong, editors, *CSF 2017 Computer Security Foundations Symposium*, pages 203–216. IEEE Computer Society Press, 2017. doi:10.1109/CSF.2017.21.

[MP16] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol, November 2016. URL: https://www.signal.org/docs/specifications/x3dh/.

[Nat23] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard, August 2023. FIPS 203 (initial public draft). doi:10.6028/NIST.FIPS.203.ipd.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[PM16] Trevor Perrin and Moxie Marlinspike. The double ratchet algorithm, November 2016. URL: https://www.signal.org/docs/specifications/doubleratchet/.

[PR18] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96884-1_1.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[SFG23] Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in TLS 1.3. https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-08.txt, August 2023. Internet-Draft.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365700.

[Sig16] Signal. Technical information, 2016. Accessed December 10, 2023. URL: https://www.signal.org/docs/.

[VGIK20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the Signal protocol. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part II*, volume 12147 of *LNCS*, pages 188–209. Springer, Heidelberg, October 2020. doi:10.1007/978-3-030-57878-7_10.

[Zim95] Philip Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

A List of changes

- January 15, 2024: Initial version (public release: February 21, 2024).
- February 27, 2024: Clarify use of ML-KEM-ipd instead of Kyber.