Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake

Jacqueline Brendel[1], Rune Fiedler[1], Felix Günther[2], Christian Janson[1], and Douglas Stebila[3]

[1] Technische Universität Darmstadt `firstname.lastname@cryptoplexity.de`
[2] ETH Zürich `mail@felixguenther.info`
[3] University of Waterloo `dstebila@uwaterloo.ca`

Version 1.2*, March 2022

Abstract. The key exchange protocol that establishes initial shared secrets in the handshake of the Signal end-to-end encrypted messaging protocol has several important characteristics: (1) it runs asynchronously (without both parties needing to be simultaneously online), (2) it provides implicit mutual authentication while retaining deniability (transcripts cannot be used to prove either party participated in the protocol), and (3) it retains security even if some keys are compromised (forward secrecy and beyond). All of these properties emerge from clever use of the highly flexible Diffie–Hellman protocol.
While quantum-resistant key encapsulation mechanisms (KEMs) can replace Diffie–Hellman key exchange in some settings, there is no replacement for the Signal handshake solely from KEMs that achieves all three aforementioned properties, in part due to the inherent asymmetry of KEM operations. In this paper, we show how to construct asynchronous deniable key exchange by combining KEMs and designated verifier signature (DVS) schemes, matching the characteristics of Signal. There are several candidates for post-quantum DVS schemes, either direct constructions or via ring signatures. This yields a template for an efficient post-quantum realization of the Signal handshake with the same asynchronicity and security properties as the original Signal protocol.

Keywords: authenticated key exchange · deniability · asynchronous · Signal protocol · post-quantum · designated verifier signatures

1 Introduction

The Signal protocol [72,71], designed by Marlinspike and Perrin, has enabled mass adoption of end-to-end encrypted messaging in consumer applications such as WhatsApp, Signal, Facebook Messenger, Skype, and more. From a cryptographic perspective, the Signal protocol consists of an initial handshake and key exchange (called "X3DH" [72], a simplified version of which is shown in Figure 1), asymmetric and symmetric key exchange "ratchets" that establish new keys for every new chat message sent (called the "double ratchet" algorithm [71]), and symmetric authenticated encryption for application data. Each of these components contributes to Signal's interesting and useful security features:

– *Implicit mutual authentication in the handshake*: The session key K established in the handshake can only be computed by the intended peer. This comes from the terms involving the long-term secret keys a and b in Figure 1.
– *Forward secrecy in the handshake*: The session key K established in the handshake remains secret even if long-term keys are later compromised. This comes from the terms involving the ephemeral keys x and y in Figure 1.
– *Offline deniability of the handshake*: A judge seeing a transcript of an honest communication session cannot be convinced that a particular party was actually involved in the session. This comes from the use of Diffie–Hellman for authentication rather than signatures; all of the DH shared secrets input to the key derivation function in Figure 1 could have been computed unilaterally either by Alice or by Bob (e.g., both Alice and Bob can compute g^{as}, using a and s respectively). We provide a new formalization of deniability reflecting the specification of Signal more closely. We discuss the differences between the deniability notions in Section 4 and in more detail in Appendix A. While a formal proof that X3DH fulfills our new notion is not known to the authors, we expect it to hold without any additional assumptions. See [87] for a detailed analysis of the deniability of X3DH with respect to the deniability notion of [28].

*A preliminary version of this paper appears in the proceedings of the *25th International Conference on Practice and Theory of Public-Key Cryptography (PKC 2022)*, © IACR 2022. DOI: 10.1007/978-3-030-97131-1_1. This is the full version. Please see Appendix C for an overview of major changes from earlier versions of this work.

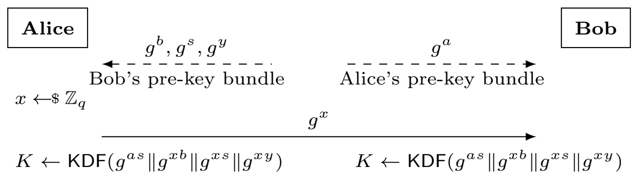
Alice
Bob
g^b, g^s, g^y
Bob's pre-key bundle
g^a
Alice's pre-key bundle
$x \leftarrow\$ \mathbb{Z}_q$
g^x
$K \leftarrow \mathsf{KDF}(g^{as} \| g^{xb} \| g^{xs} \| g^{xy})$
$K \leftarrow \mathsf{KDF}(g^{as} \| g^{xb} \| g^{xs} \| g^{xy})$

Fig. 1. Simplified version of Signal's X3DH handshake.
Long-term keys a and b; semi-static key s; ephemeral keys x and y.

- *Asynchronicity*: The two communicating parties need never be online simultaneously, and can leave packets at an untrusted relay server until the other party comes back online. The handshake is made asynchronous by allowing each party to upload a *pre-key bundle* to an untrusted server in advance, consisting of long-term, medium-term, and ephemeral public keys, and an initiator can start sending text messages before their peer comes online. The restrictions on communication flow in an asynchronous protocol are weaker than those of non-interactive key exchange [42].
- *Forward secrecy and post-compromise security [22] in long-lived conversations*: Keys are updated using a new DH key exchange with each chat message via the asymmetric ratchet, enabling secrecy of past and future messages after a compromise.

1.1 Making Signal Post-quantum

Since the Diffie–Hellman problem upon which much of Signal relies is not secure against quantum adversaries, it is important to have a post-quantum alternative available.

The symmetric ratchet and authenticated encryption components of Signal are built on symmetric primitives, and thus are not in immediate danger from quantum algorithms. The asymmetric ratchet was phrased by Marlinspike and Perrin [71] and analyzed by Cohn-Gordon, Cremers, Dowling, Garratt, and Stebila [21] in terms of Diffie–Hellman. Alwen, Coretti, and Dodis [1] generalized it into a primitive called continuous key agreement that can be built from KEMs, yielding post-quantum security. Hence, our focus in the rest of this paper is on the handshake.

The post-quantum primitives to be standardized by the United States National Institute of Standards and Technology (NIST) post-quantum standardization project are signatures and key encapsulation mechanisms (KEMs), so these would be most preferable to employ. It is certainly possible to generically construct an authenticated key exchange protocol from signatures and KEMs, but it is not possible to use *only* KEMs and signatures in a generic way to create a post-quantum replacement for Signal with all of the properties listed above. Suppose one tried to use KEMs instead of Diffie–Hellman in Figure 1. Recall that, to use a KEM for key exchange, one party uses the key generation algorithm to create a public-key/secret-key pair and transmits the public key to their peer; the peer encapsulates against that public key, producing a ciphertext and a shared secret, then transmits the ciphertext, which the first party decapsulates using their secret key to compute the shared secret. In the Signal handshake, one could try using KEM public keys to replace the Diffie–Hellman shares in Alice and Bob's pre-key bundles. We can still obtain ephemeral key exchange (by having Alice encapsulate against Bob's ephemeral public key) and implicit Bob-to-Alice authentication (by having Alice encapsulate against Bob's long-term public key). However, we cannot obtain Alice-to-Bob authentication using KEMs without adding an extra flow: Bob cannot produce a ciphertext for Alice to decapsulate without knowing Alice's public key first, so he cannot asynchronously produce a pre-key bundle for Alice to immediately use. This highlights the difference between DH and KEMs: in DH, both parties' shares are objects of the same type and can be generated independently, but in generic KEMs, public keys and ciphertexts are in principle objects of differing types and a ciphertext is generated with respect to a given public key. To obtain Alice-to-Bob authentication without adding an extra communication round, Alice could of course produce a signature for Bob to verify, but this undermines deniability.

The problem, in a nutshell, is to create an *asynchronous deniable authenticated key exchange protocol* that can be instantiated in the post-quantum setting, preferably with an efficient construction based on standardized primitives or at least cryptographic assumptions used in standardized primitives.

1.2 Options for PQ Asynchronous DAKE

There are several examples of authenticated key exchange protocols built generically from KEMs which have the potential for deniability [13,12,43,26,81] but do not have the desired asynchronicity property for reasons similar to the discussion above.

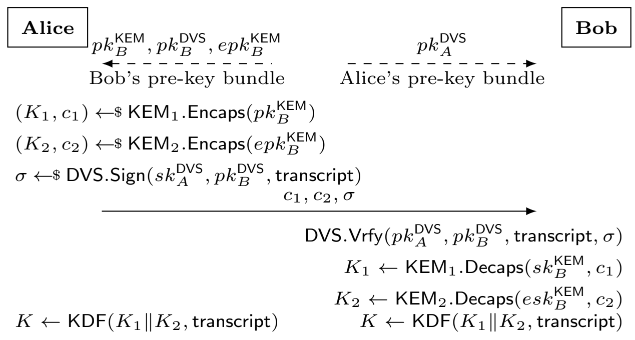
Alice
Bob
$pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}, epk_B^{\mathsf{KEM}}$
Bob's pre-key bundle
pk_A^{DVS}
Alice's pre-key bundle
$(K_1, c_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(pk_B^{\mathsf{KEM}})$
$(K_2, c_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(epk_B^{\mathsf{KEM}})$
$\sigma \leftarrow\!\$\ \mathsf{DVS}.\mathsf{Sign}(sk_A^{\mathsf{DVS}}, pk_B^{\mathsf{DVS}}, \mathsf{transcript})$
c_1, c_2, σ
$\mathsf{DVS}.\mathsf{Vrfy}(pk_A^{\mathsf{DVS}}, pk_B^{\mathsf{DVS}}, \mathsf{transcript}, \sigma)$
$K_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_B^{\mathsf{KEM}}, c_1)$
$K_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(esk_B^{\mathsf{KEM}}, c_2)$
$K \leftarrow \mathsf{KDF}(K_1 \| K_2, \mathsf{transcript})$
$K \leftarrow \mathsf{KDF}(K_1 \| K_2, \mathsf{transcript})$

Fig. 2. Our core asynchronous DAKE protocol, combining static and ephemeral key encapsulation schemes KEM_1 and KEM_2, and a designated verifier signature DVS.

One post-quantum option that avoids the problem with KEMs described above is to use CSIDH [20], a primitive based on supersingular isogenies that yields a commutative group action which enables non-interactive key exchange. CSIDH could be used to achieve implicit Alice-to-Bob authentication while maintaining asynchronicity and deniability; indeed several key exchange protocols from CSIDH have been proposed [56,55]. Unfortunately, there are several reasons CSIDH may not be a fully satisfactory solution: it is much more computationally expensive than most other forms of post-quantum cryptography, and there is ongoing debate about the security of its concrete parameters [75,11].

Most other post-quantum assumptions used in KEMs, including SIDH [53] and learning-with-errors (LWE) [78], are insecure against key reuse attacks without additional protection such as the Fujisaki–Okamoto transform [44] that leaves them unable to be used for non-interactive key exchange (since the ciphertext must be generated with respect to a given public key). There have been several attempts at SIDH-based non-interactive key exchange which have ended up being insecure [2,35,30,31], and one attempt relying on an additional novel assumption [10] the security of which is unknown.

Brendel, Fischlin, Günther, Janson, and Stebila [15] previously considered the question of building a post-quantum version of the Signal handshake, highlighting many of these problems. They proposed decomposing the three operations of a KEM into a 4-operation "split KEM", and showed how a Signal-like handshake could be built from a split KEM meeting a suitably strong security notion. They showed how CSIDH and LWE could be used to build split KEMs meeting a weaker security notion, but these constructions did not achieve the strong security notion required for their Signal-like handshake, effectively leaving the overall problem unsolved.

Unger and Goldberg [85,86] also consider deniable authenticated key exchange (DAKE) protocols for secure messaging. Their protocol permits the optional use of a PQ KEM for ephemeral key exchange to achieve forward secrecy against future-quantum adversaries. To achieve deniability, they employ ring signatures with classical security and further rely on dual receiver encryption, which does not yet appear to have a PQ instantiation in the literature. Observe that their formalization of deniability is given in the UC model.

The recent work by Hashimoto, Katsumata, Kwiatkowski, and Prest [48] is closest to ours. Their core protocol is meant to replace the Signal handshake based on (post-quantum) KEMs and signatures. It achieves security against exposure of long-term keys and session state and a weaker deniability level. Unlike Signal (and our proposed protocol), it does however not provide security against randomness exposure and lacks support for semi-static keys to mitigate the exhaustion of ephemeral pre-keys. Hashimoto et al. provide an implementation for their weakly-deniable protocol and further discuss two additional variants achieving stronger deniability. The second protocol achieves deniability against semi-honest adversaries based on ring signatures, while their third protocol additionally uses non-interactive zero-knowledge arguments and strong knowledge-type assumptions for plaintext-aware [4] KEMs to achieve deniability against malicious adversaries.

Dobson and Galbraith [29] recently proposed using SIDH key exchange to replace the DH key exchange in the (slightly modified) X3DH protocol. Even though SIDH is in general insecure against adaptive attacks, Dobson and Galbraith show that carefully adding a zero-knowledge proof enables them to prove that the long-term SIDH public keys are generated honestly. In order to prove deniability, they require strong knowledge-type assumptions following [87].

1.3 Our Contributions

We show how to construct an asynchronous deniable authenticated key exchange protocol generically from designated verifier signature schemes and key encapsulation mechanisms.

Introduced by Jakobsson, Sako, and Impagliazzo [52], a *designated verifier signature (DVS) scheme* allows a signer to convince a chosen recipient, called the designated verifier, of the authenticity of a message, but in such a way that the designated verifier cannot convince any other party of the authenticity. In a DVS scheme, both the signer and the verifier have a public-key/secret-key pair; signing requires both the signer's secret key and the verifier's public key, and verification uses both parties' public keys. To achieve the non-transferability property (called "source hiding"), a DVS scheme is accompanied by an additional simulation algorithm with which the designated verifier can, using its own secret key, construct a signature indistinguishable from one generated by the signer.

Asynchronous DAKE construction. We combine a DVS with a KEM to achieve an asynchronous deniable authenticated key exchange as shown in Figure 2. As expected, Bob-to-Alice authentication comes from an implicitly authenticated key exchange in which Alice encapsulates to Bob's long-term KEM key (KEM_1 with long-term public key pk_B^{KEM} and ciphertext c_1 in Figure 2), and forward secrecy comes from a key exchange using an ephemeral KEM key (KEM_2 with public key epk_B^{KEM} and ciphertext c_2). Alice-to-Bob authentication comes from Alice using the designated verifier signature scheme to sign a transcript with Bob as the designated verifier; she can obtain Bob's DVS verification key (pk_B^{DVS}) from his pre-key bundle. Since the source hiding property of the DVS scheme enables Bob to also have created a valid-looking signature from Alice with himself as the designated verifier, the transcript of the key exchange protocol could have been constructed by either Alice or Bob, yielding the desired deniability property.

Deniability. We model the informal deniability requirement from the Signal specification [72, §4.4] through a new deniability notion (for asynchronous DAKE) capturing the following scenario: Alice wants to convince a judge that a certain conversation took place between her and Bob. Hence, Alice gives the corresponding transcript to the judge. The judge may coerce Alice and Bob to give up their secret keys (e.g., by law). Under these circumstances, the judge should not be able to tell if this transcript stems from a real conversation or if Alice faked the transcript on her own without Bob's interaction. On the one hand, our new notion is weaker than the definition of [28] in the sense that we limit Alice to stick to the protocol description (i.e., be semi-honest) and allow the use of a secret key for faking a transcript. On the other hand, our new notion is stronger in the sense that we allow the judge to know *all* secret keys. On a more technical note, we provide a game-based definition while [28] uses the simulation paradigm. In a nutshell, a strength of our notion is that it provides deniability against powerful judges that can compromise secret keys of users. A consequence of our new, incomparable deniability definition is that we can achieve it without strong knowledge assumptions needed for X3DH [87] and in the work of Hashimoto et al. [48,49]; we conjecture both protocols can likewise be shown to be deniable wrt. our definition without such assumptions.

Post-quantum designated verifier signatures. To achieve our goal of post-quantum asynchronous DAKE, we thus need a post-quantum designated verifier signature scheme. While there is a long line of research on DVS schemes from pre-quantum assumptions (including [52,80,60,83,64,90,16,25]), comparatively little is available in the literature on post-quantum DVS schemes. An isogeny-based DVS scheme was proposed in [84] but is insecure due to key reuse attacks identified in [45]. There are several lattice-based DVS schemes which may fit the bill [88,89,74,62,93], but these have not received much scrutiny in the mainstream cryptographic literature; we summarize this literature in Section 3.1. These lattice-based DVS schemes are direct constructions not based on any NIST candidates, so they would require their own thorough analysis.

DVS from ring signatures. Rather than constructing DVS schemes directly, it is possible to use a *ring signature scheme* [79] as a designated verifier signature scheme. In a ring signature scheme, one signer can sign a message intended to verify under a *ring* of public keys, only one of which is theirs; yet no one should be able to determine which signer produced such a signature. Following ideas sketched in [79,7], we show in Section 3.2 how to use a 2-user ring signature scheme to build a DVS scheme: the ring used by the signer consists of the public keys of themselves and the one designated verifier. There are several candidates for post-quantum ring signatures whose properties we discuss in Section 3.2.

In a concurrent update to their work, Hashimoto et al. have shown the reverse, i.e., constructing a ring signature scheme from a DVS scheme [49] (which is the full version of [48]). Hence, in the 2-user case ring signatures and DVS are equivalent under the security notions put forward in this paper.

Given this equivalence, observe that our core asynchronous DAKE protocol (Figure 2) is indeed similar to the second construction of [48]. While our construction sends the DVS signature as is, their construction employs a ring signature that is masked with the output of a PRF evaluation.

Application to the Signal handshake. We present a version of the Signal X3DH handshake which we call SPQR—Signal in a Post-Quantum Regime—based on our asynchronous DAKE design that uses KEMs and a designed verifier signature scheme. We show that the SPQR handshake achieves strong ("maximal-exposure") session key security in a variant of the security model of [21] covering compromises of long- and medium-term keys and ephemeral randomness, as well as deniability.

Outline of the paper. In Section 2 we introduce preliminaries. Section 3 focuses on the security properties of designated verifier schemes and how to construct these in a post-quantum setting, including existing direct constructions as well as via ring signatures, and gives a discussion of our failed attempts at constructing DVS from chameleon hash functions in an earlier version of this work. In Section 4 we present a security model for key exchange that captures session key indistinguishability with implicit mutual authentication and weak forward secrecy, as well as offline deniability. In Section 5 we show that our core asynchronous deniable authenticated key exchange protocol from Figure 2 fulfills these security notions; in particular, offline deniability is based on the source hiding property of the DVS scheme. In Section 6 we introduce a complete post-quantum version of the Signal handshake that extends our core protocol to include additional components present in the Signal handshake (e.g., semi-static keys). In Section 7 we provide a security model for our full protocol and prove, in Section 8, its session key indistinguishability and deniability. In Section 9, we conclude with a discussion of the results and some limitations.

2 Preliminaries

We begin by introducing notation and recapping some basic components.

2.1 Notation

To sample an element x uniformly at random from a set $\mathcal{S}$ (or a distribution on an underlying set) we write $x \leftarrow_\$ \mathcal{S}$. For deterministic algorithms A we denote by $y \leftarrow \mathsf{A}(x)$ the execution of A on input x with output y. Similarly, $y \leftarrow_\$ \mathsf{A}(x)$ denotes the probabilistic execution of A, and $y \leftarrow \mathsf{A}(x; r)$ the deterministic execution of a probabilistic algorithm A with its random coins fixed to r. Adversaries are typically denoted by $\mathcal{A}$ and we write $\mathcal{A}^{\textsc{Oracle}}$ to indicate that $\mathcal{A}$ has access to the oracle $\textsc{Oracle}$. Adversaries can have local quantum computation power but their oracle access and outputs are still classical. For an integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$. Double square brackets $[\![\cdot]\!]$ that enclose a boolean statement return the bit 1 if the statement is true, and 0 otherwise.

2.2 Key Encapsulation Mechanisms

The main building block for our post-quantum secure initial key agreement of Signal are so-called *key encapsulation mechanisms* that allow an *encapsulator* to transfer a shared secret key K via a ciphertext c to the *decapsulator*.

Definition 1 (Key Encapsulation Mechanisms). *A* key encapsulation mechanism KEM *is a triple of algorithms* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$*. In more detail:*

- $\mathsf{KGen}() \,{}_\$\!\!\rightarrow (pk, sk)$*: A probabilistic algorithm that outputs a public-key/secret-key pair with* $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$*.*
- $\mathsf{Encaps}(pk) \,{}_\$\!\!\rightarrow (c, K)$*: A probabilistic algorithm taking as input a public key* $pk \in \mathcal{PK}$ *and outputs a ciphertext* $c \in \mathcal{C}$ *and the therein encapsulated key* $K \in \mathcal{K}$*.*
- $\mathsf{Decaps}(sk, c) \rightarrow K'$*: A deterministic algorithm taking as input a ciphertext* $c \in \mathcal{C}$ *and secret key* sk *and outputs* $K' \in \mathcal{K} \cup \{\perp\}$*, where* $\perp$ *indicates an error.*

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

$$\Pr[K' \neq K \mid K' \leftarrow \mathsf{Decaps}(sk, c)] \leq \delta.$$

We call KEM (perfectly) correct *if* $\delta = 0$*.*

Security of KEMs is defined in terms of indistinguishability of encapsulated keys from random given the decapsulator's public key and the encapsulating ciphertext:

```
$\mathcal{G}^{\mathsf{indcpa}}_{\mathsf{KEM}}(\mathcal{A})$:
$(pk, sk) \leftarrow\$ \mathsf{KGen}()$
$(c^*, K_0^*) \leftarrow\$ \mathsf{Encaps}(pk)$
$K_1^* \leftarrow\$ \mathcal{K}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}(pk, c^*, K_b^*)$
return $[\![b' = b]\!]$
```

```
$\mathcal{G}^{\mathsf{indcca}}_{\mathsf{KEM}}(\mathcal{A})$:
$(pk, sk) \leftarrow\$ \mathsf{KGen}()$
$(c^*, K_0^*) \leftarrow\$ \mathsf{Encaps}(pk)$
$K_1^* \leftarrow\$ \mathcal{K}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\textsc{Decaps}}(pk, c^*, K_b^*)$
return $[\![b' = b]\!]$
```

```
Decaps($c$):
if $c = c^*$
    return $\perp$
else
    return $\mathsf{Decaps}(sk, c)$
```

Fig. 3. IND-CPA and IND-CCA security for $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with key space $\mathcal{K}$.

Definition 2 (IND-ATK Security of KEMs). *Let* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be a KEM with key space* $\mathcal{K}$. *We say that* KEM *is* (t, ϵ)-IND-CPA-secure*, resp.* (t, ϵ, Q_D)-IND-CCA*-secure, if for any adversary* $\mathcal{A}$ *with running time at most* t *and (for* IND-CCA*) making at most* Q_D *queries to the* Decaps *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{indatk}}_{\mathsf{KEM}}(\mathcal{A}) := \left| \Pr\left[\mathcal{G}^{\mathsf{indatk}}_{\mathsf{KEM}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{indatk}}_{\mathsf{KEM}}(\mathcal{A})$ *(with* $\mathsf{atk} = \mathsf{cpa}$*, resp.* $\mathsf{atk} = \mathsf{cca}$*) is defined in Figure 3.*

2.3 (Twisted) Pseudorandom Functions

Beyond classical pseudorandom functions for key derivation, another crucial component for our SPQR protocol are special pseudorandom functions called *twisted pseudorandom functions* [43,59]. In the following we recall the respective definitions and security games.

Definition 3. *Let* $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^\iota \to \{0,1\}^\omega$ *be an efficient keyed function with key length* κ*, input length* ι*, and output length* ω*.*

Let $\mathcal{G}^{\mathsf{prfsec}}_{\mathsf{F}}(\mathcal{A})$ *be defined as in the top of Figure 4. We call* F *a* $(t, \epsilon, Q_{\mathsf{F}})$-pseudorandom function *(or simply* $(t, \epsilon, Q_{\mathsf{F}})$-PRFSEC*), if for any adversary* $\mathcal{A}$ *with running time at most* t *and making at most* Q_{F} *queries to the* PRFChallenge *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{prfsec}}_{\mathsf{F}}(\mathcal{A}) := \left| \Pr\left[\mathcal{G}^{\mathsf{prfsec}}_{\mathsf{F}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \epsilon.$$

Let $\mathcal{G}^{\mathsf{tprfsec}}_{\mathsf{F}}(\mathcal{A})$ *be defined as in the bottom of Figure 4. We call* F *a* (t, ϵ, q)-twisted pseudorandom function *(or simply* (t, ϵ, q)-tPRFSEC*), if for any adversary* $\mathcal{A}$ *with running time at most* t*, we have that*

$$\mathsf{Adv}^{\mathsf{tprfsec}}_{\mathsf{F},q}(\mathcal{A}) := \left| \Pr\left[\mathcal{G}^{\mathsf{tprfsec}}_{\mathsf{F},q}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \epsilon.$$

Note that one can easily build a twisted PRF tPRF from a PRF F in the standard model. Following Kurosawa and Furukawa [59], a secure construction doubling the key and label lengths is:

$$\mathsf{tPRF}((k, k'), (e, e')) = \mathsf{F}(k, e) \oplus \mathsf{F}(e', k').$$

3 Designated Verifier Signatures

Designated verifier signature (DVS) schemes were introduced by Jakobsson, Sako, and Impagliazzo [52]. Their goal is for a signer to convince a chosen recipient (the "designated verifier") that a message is authentic but in such a way that the designated verifier cannot convince any other party of the authenticity of the message[4]. This property is typically modeled by requiring that the designated verifier can efficiently simulate signatures that are indistinguishable from signatures produced by the signer.

Definition 4. *A* designated verifier signature scheme *(DVS) is a tuple of algorithms* $\mathsf{DVS} = (\mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Sim})$ *along with a message space* $\mathcal{M}$.

[4] In contrast, a *strong* DVS scheme allows only the designated verifier to verify a signature by requiring the verifier's secret key as input to the verification algorithm.

```
$\mathcal{G}^{\mathsf{prfsec}}_{\mathsf{F}}(\mathcal{A})$:
1 $K \leftarrow\$ \{0,1\}^\kappa$
2 $g \leftarrow\$ \{\text{functions } f : \{0,1\}^\iota \to \{0,1\}^\omega\}$
3 $b \leftarrow\$ \{0,1\}$
4 $b' \leftarrow\$ \mathcal{A}^{\text{PRFChallenge}}()$
5 return $[\![b' = b]\!]$

$\text{PRFChallenge}(x)$:
6 if $b = 0$
7     return $\mathsf{F}(K, x)$
8 else
9     return $g(x)$
```

```
$\mathcal{G}^{\mathsf{tprfsec}}_{\mathsf{F},q}(\mathcal{A})$:
1 $g \leftarrow\$ \{\text{functions } f : \{0,1\}^\iota \to \{0,1\}^\omega\}$
2 $g' \leftarrow\$ \{\text{functions } f : \{0,1\}^\kappa \to \{0,1\}^\omega\}$
3 $K, K' \leftarrow\$ \{0,1\}^{2\kappa}$
4 $b \leftarrow\$ \{0,1\}$
5 $x, x_1, x_2, \ldots, x_q \leftarrow\$ \{0,1\}^{(q+1)\iota}$
6 $s_0 \leftarrow \{(x_1, \mathsf{F}(K, x_1)), (x_2, \mathsf{F}(K, x_2)), \ldots, (x_q, \mathsf{F}(K, x_q)), (K', \mathsf{F}(K', x))\}$
7 $s_1 \leftarrow \{(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_q, g(x_q)), (K', g'(K'))\}$
8 $b' \leftarrow\$ \mathcal{A}(s_b)$
9 return $[\![b' = b]\!]$
```

Fig. 4. Pseudorandomness ($\mathcal{G}^{\mathsf{prfsec}}_{\mathsf{F}}(\mathcal{A})$, top) and twisted pseudorandomness ($\mathcal{G}^{\mathsf{tprfsec}}_{\mathsf{F}}(\mathcal{A})$, bottom) of a function F.

– $\mathsf{SKGen}() \xrightarrow{\$} (pk_S, sk_S)$*: A probabilistic key generation algorithm that outputs a public-/secret-key pair for the signer.*
– $\mathsf{VKGen}() \xrightarrow{\$} (pk_D, sk_D)$*: A probabilistic key generation algorithm that outputs a public-/secret-key pair for the verifier.*
– $\mathsf{Sign}(sk_S, pk_D, m) \xrightarrow{\$} \sigma$*: A probabilistic signing algorithm that uses a signer secret key sk_S to produce a signature σ for a message $m \in \mathcal{M}$ for a designated verifier with public key pk_D.*
– $\mathsf{Vrfy}(pk_S, pk_D, m, \sigma) \to \mathsf{true}/\mathsf{false}$*: A deterministic verification algorithm that checks a message m and signature σ against a signer public key pk_S and verifier public key pk_D.*
– $\mathsf{Sim}(pk_S, sk_D, m) \xrightarrow{\$} \sigma$*: A probabilistic signature simulation algorithm that uses the verifier's secret key sk_D to produce a signature σ on message m for signer public key pk_S.*

A DVS scheme DVS *is* correct*, if, for any honestly generated key pairs $(pk_S, sk_S), (pk_D, sk_D)$ and every message $m \in \mathcal{M}$, it holds that*

$$\Pr[\mathsf{Vrfy}(pk_S, pk_D, m, \mathsf{Sign}(sk_S, pk_D, m)) = \mathsf{true}] = 1.$$

We follow Laguillaumie and Vergnaud [60] in defining separate key generation algorithms for signers and designated verifiers; in some cases these two algorithms may be identical.

A long line of research has scrutinized the security of DVS schemes in different settings, e.g. strong DVS schemes, including [52,80,60,83,64,90,16,25]. For the purpose of this paper, it suffices to define the security notions of *unforgeability* and *source hiding*. Unforgeability for DVS schemes is similar to that for standard signature schemes, providing the adversary with a signing oracle and asking it to forge a signature on a (fresh) message of its choice. Prior work restricts the signing oracle to the challenge designated verifier key. In contrast, and to account for settings where a signer's key is used with many other users' verifier keys (cf. Section 5), we allow the adversary to pick the designated verifier key to be used in the signing oracle from a set of additional, honestly generated key pairs.

Definition 5. *A designated verifier signature scheme* DVS *is* (t, ϵ, n, Q_S)-unforgeable *if, for any adversary $\mathcal{A}$ with running time at most t, having access to n additional DVS verifier key pairs beyond the challenge keys, and making at most Q_S queries to the* Sign *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A}) = 1\right] \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A})$ is as in Figure 5.

The second property we consider is called source hiding [60], demanding that it should be infeasible for an adversary to determine whether a given signature has been generated by the signer (using Sign) or by the designated verifier (using Sim), even if the adversary learns the secret keys of both parties.

```
$\mathcal{G}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A})$:
$Q \leftarrow \emptyset$
$\mathcal{L} \leftarrow \emptyset$
$(pk_S, sk_S) \leftarrow\!\$\ \mathsf{DVS.SKGen}()$
$(pk_D, sk_D) \leftarrow\!\$\ \mathsf{DVS.VKGen}()$
**for** $i \in [n]$
    $(pk_i, sk_i) \leftarrow\!\$\ \mathsf{DVS.VKGen}()$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(pk_i, sk_i)\}$
$(m^*, \sigma^*) \leftarrow\!\$\ \mathcal{A}^{\textsc{Sign}}(pk_S, pk_D, \mathcal{L})$
$d \leftarrow \mathsf{DVS.Vrfy}(pk_S, pk_D, m^*, \sigma^*)$
**return** $[\![d = \mathsf{true} \ \wedge \ m^* \notin Q]\!]$
```

```
$\textsc{Sign}(pk, m)$:
**if** $pk = pk_D$
    $Q \leftarrow Q \cup \{m\}$
**else if** $(pk, \cdot) \notin \mathcal{L}$
    **return** $\perp$
$\sigma \leftarrow\!\$\ \mathsf{DVS.Sign}(sk_S, pk, m)$
**return** $\sigma$
```

```
$\mathcal{G}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A})$:
$(pk_S, sk_S) \leftarrow\!\$\ \mathsf{DVS.SKGen}()$
$(pk_D, sk_D) \leftarrow\!\$\ \mathsf{DVS.VKGen}()$
$b \leftarrow\!\$\ \{0, 1\}$
$b' \leftarrow\!\$\ \mathcal{A}^{\textsc{Chall}}(pk_S, sk_S, pk_D, sk_D)$
**return** $[\![b' = b]\!]$
```

```
$\textsc{Chall}(m)$:
**if** $b = 0$
    $\sigma \leftarrow\!\$\ \mathsf{DVS.Sign}(sk_S, pk_D, m)$
**else**
    $\sigma \leftarrow\!\$\ \mathsf{DVS.Sim}(pk_S, sk_D, m)$
**return** $\sigma$
```

Fig. 5. Unforgeability (top) and source hiding (bottom) of a designated verifier signature scheme DVS.

Definition 6. *A designated verifier signature scheme* DVS *is* (t, ϵ, Q_{Ch})-source hiding *if, for any adversary* $\mathcal{A}$ *with running time at most* t *and making at most* Q_{Ch} *to the* Chall *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A}) = \left| \Pr\left[\mathcal{G}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A})$ *is defined in Figure 5.*

The property of source hiding also appears under different terms in the literature such as the designated verifier property [52,80], non-transferability [83], source deniable [41], untransferability [16], and recently off-the-record [25]. While all these definitions share the intuition that the sender can blame another party (in particular, the designated receiver) as the originator of a signature, they vary in the adversary capabilities, i.e., whether the adversary is unbounded or whether it gets access to the secret keys.

3.1 Post-quantum DVS Schemes: Prior Work and Failed Attempts

For this work, we are interested in DVS constructions that promise post-quantum security. Despite the long line of research on DVS schemes, there are only a few candidate post-quantum constructions available in the literature; furthermore, most of those have not received much scrutiny in the mainstream cryptographic literature.

This led us to attempt building, in a prior version of this paper (see Appendix C), a generic construction of post-quantum-secure DVS schemes from chameleon hash functions through both full-domain-hash and Fiat–Shamir-style signature schemes, drawing from post-quantum building blocks much closer to schemes involved in NIST standardization. In the following, we summarize prior direct constructions and our own attempts, which ultimately failed, before turning to generic constructions from ring signatures in Section 3.2.

Post-quantum DVS constructions in the literature. An isogeny-based strong DVS scheme was proposed by Sun, Tian, and Wang [84] which turned out to be insecure due to key reuse attacks identified by Galbraith, Petit, Shani, and Ti [45].

Wang, Hu, and Wang [88] construct a strong DVS scheme directly from lattice assumptions (LWE and SIS) by combining the Bonsai tree lattice trapdoor of [19] with the GPV lattice-based signature scheme [46]; a subsequent paper of theirs [89] extends this to the identity-based setting.

Noh and Jeong [74] improve on [88,89] by giving direct constructions from lattices that can be proven without relying on random oracles; they do so by replacing the random oracle with a chameleon hash function.

Li, Liu, and Yang [62] construct a universal DVS scheme directly from ideal lattice assumptions (ring-SIS) by combining a ring version of the GPV signature scheme [69] with a ring chameleon hash function [34] and adding a Fiat–Shamir-with-aborts technique [67,68].

Zhang, Liu, Tang, and Tian [93] also give a universal DVS constructed directly from SIS by adapting the Lyubashevsky signature scheme [68].

Construction attempts: GPV and Fiat–Shamir. In a previous version of this paper (see Appendix C) we gave two attempted generic DVS constructions, to be instantiated from post-quantum building blocks close to schemes involved in NIST standardization.

- Our first DVS construction was based on the full-domain-hash signature scheme [5], although following the variant by Gentry, Peikert, and Vaikuntanathan [46] which uses a trapdoor function rather than a trapdoor permutation as in [5].
- Our second DVS construction was based on the method of Fiat and Shamir [40] for constructing a signature scheme from an honest-verifier zero-knowledge canonical identification protocol.

In both of these signature schemes, signatures were constructed in the normal "forward" direction by the signer using the hashing and signing algorithms in the normal way. One can attempt to construct signatures in the "backward" direction without the secret key by applying the permutation (for the full-domain hash scheme) or generating an identification protocol transcript (in the Fiat–Shamir case), but a forger will get stuck without a way to make the hash of the message match the hash digest picked during the backwards signature generation. The key idea in both of our constructions was to replace the standard hash function with a *chameleon hash function* (CHF) [58,19], which allows preimages of the hash function to be found with knowledge of a trapdoor, which will be held by the verifier.

The security proof for these constructions falsely modeled the CHF as random oracle, which is not faithful. For transparency and educational purposes we describe our insights in Appendix B.

3.2 Building Post-quantum DVS Schemes from Ring Signatures

We now turn to building DVS schemes generically from ring signatures, show which properties are required to obtain a post-quantum-secure instantiation and evaluate several ring signature candidates. Our constructions draws from the idea sketched in [79,7], with syntax and security closely following the exposition of Bender, Katz, and Morselli [7].

Definition 7. *A* ring signature scheme *is a tuple of algorithms* $\mathsf{Ring} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *along with a message space* $\mathcal{M}$*.*

- $\mathsf{KGen}() \xrightarrow{\$} (pk, sk)$*: A probabilistic key generation algorithm that outputs a public-/secret-key pair.*
- $\mathsf{Sign}(sk_s, m, \mathsf{R}) \xrightarrow{\$} \sigma$*: A probabilistic signing algorithm that uses a secret key* sk_s *to produce a signature* σ *for a message* $m \in \mathcal{M}$ *w.r.t. to a list of distinct public keys* R*, where* (pk_s, sk_s) *is an honestly generated key pair and* $pk_s \in \mathsf{R}$*.*
- $\mathsf{Vrfy}(\mathsf{R}, m, \sigma) \rightarrow \mathsf{true}/\mathsf{false}$*: A deterministic verification algorithm that checks a message* m *and signature* σ *against a ring* R*.*

A 2-user ring signature *is a ring signature fixed to rings of size 2. A ring signature scheme* Ring *is* correct*, if, for honestly generated key pairs* $\{(pk_i, sk_i)\}_{i=1}^n$*, any* $s \in [n]$*, and any message* $m \in \mathcal{M}$*, it holds that*

$$\Pr[\mathsf{Vrfy}(\{pk_i\}_{i=1}^n, m, \mathsf{Sign}(sk_s, m, \{(pk_i)\}_{i=1}^n)) = \mathsf{true}] = 1.$$

The unforgeability and anonymity property we require for ring signatures are subtly different from prior literature. Like in the unforgeability notion w.r.t. insider corruption defined in [7], we consider an unforgeability adversary with access to a corruption oracle CORR. However, our unforgeability adversary is limited to rings consisting of honestly generated public keys for both its final forgery as well as the queries to the signing oracle (like the unforgeability against chosen-subring attacks defined in [7]). It is easy to see that unforgeability w.r.t. insider corruption implies our unforgeability notion. Herranz [50] informally discusses a similar notion.

Definition 8. *A ring signature scheme* Ring *is* $(t, \epsilon, n, Q_S, Q_{Co})$-unforgeable w.r.t. honest-ring insider corruption *if, for any adversary* $\mathcal{A}$ *with running time at most* t*, having access to* n *public keys, and making at most* Q_S *queries to the* SIGN *oracle and* Q_{Co} *queries to the* CORR *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A}) = 1\right] \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A})$ *is as in Figure 6.*

We consider an anonymity notion based on anonymity against full key exposure [7]. The first difference is that we directly give all secret keys to the adversary instead of providing a signing and a corruption oracle to the adversary, where the latter in [7] returns the key generation randomness. The other difference is that we parameterize the game in the number of queries Q_{Ch} allowed to the challenge oracle. As a result, anonymity against full key exposure implies our anonymity notion with $Q_{Ch} = 1$. Similarly, the anonymity notions of [65] and [39], where the attacker has access to a key generation oracle, imply our anonymity notion with $Q_{Ch} = 1$.

```
$\mathcal{G}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A})$:
$Q_S \leftarrow \emptyset$
$Q_{Co} \leftarrow \emptyset$
$\mathcal{L} \leftarrow \emptyset$
for $i \in [n]$
    $(pk_i, sk_i) \leftarrow\!\!\$\ \mathsf{Ring.KGen}()$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{pk_i\}$
$(\mathsf{R}^\star, m^\star, \sigma^\star) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Sign},\textsc{Corr}}(\mathcal{L})$
$d_1 \leftarrow \mathsf{Ring.Vrfy}(\mathsf{R}^\star, m^\star, \sigma^\star)$
$d_2 \leftarrow [\![(m^\star, \mathsf{R}^\star) \notin Q_S]\!]$
$d_3 \leftarrow [\![\mathsf{R}^\star \subseteq \mathcal{L} \backslash Q_{Co}]\!]$
return $[\![d_1 \wedge\ d_2 \wedge\ d_3]\!]$

$\textsc{Sign}(s, m, \mathsf{R})$:
if $pk_s \notin \mathsf{R} \vee s \notin [n]$   //sign wrt. honest key
    return $\perp$
if $\mathsf{R} \not\subseteq \mathcal{L}$   //sign wrt. honest ring
    return $\perp$
$Q_S \leftarrow Q_S \cup \{(m, \mathsf{R})\}$
$\sigma \leftarrow\!\!\$\ \mathsf{Ring.Sign}(sk_S, m, \mathsf{R})$
return $\sigma$

$\textsc{Corr}(i)$:
$Q_{Co} \leftarrow Q_{Co} \cup \{pk_i\}$
return $sk_i$
```

```
$\mathcal{G}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A})$:
$\mathcal{L} \leftarrow \emptyset$
for $i \in [n]$
    $(pk_i, sk_i) \leftarrow \mathsf{Ring.KGen}()$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(pk_i, sk_i)\}$
$b \leftarrow\!\!\$\ \{0, 1\}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Chall}}(\mathcal{L})$
return $[\![b' = b]\!]$

$\textsc{Chall}(m, i_0, i_1, \mathsf{R})$:
if $\{pk_{i_0}, pk_{i_1}\} \not\subseteq \mathsf{R}$   //challenge signers in ring
    return $\perp$
if $\{i_0, i_1\} \not\subseteq [n]$   //sign with honest keys only
    return $\perp$
$\sigma \leftarrow\!\!\$\ \mathsf{Ring.Sign}(sk_{i_b}, m, \mathsf{R})$
return $\sigma$
```

Fig. 6. Unforgeability w.r.t. honest-ring insider corruption (top) and anonymity against key exposure (bottom) of a ring signature scheme Ring. The latter game is specialized for the ring size 2.

```
$\mathsf{RingDVS.SKGen}()$:
$(pk_S, sk_S) \leftarrow\!\!\$\ \mathsf{Ring.KGen}()$
return $(pk_S, sk_S)$

$\mathsf{RingDVS.VKGen}()$:
$(pk_D, sk_D) \leftarrow\!\!\$\ \mathsf{Ring.KGen}()$
return $(pk_D, sk_D)$

$\mathsf{RingDVS.Sign}(sk_S, pk_D, m)$:
return $\mathsf{Ring.Sign}(sk_S, m, \{pk_S, pk_D\})$

$\mathsf{RingDVS.Sim}(pk_S, sk_D, m)$:
return $\mathsf{Ring.Sign}(sk_D, m, \{pk_S, pk_D\})$

$\mathsf{RingDVS.Vrfy}(pk_S, pk_D, m, \sigma)$:
return $\mathsf{Ring.Vrfy}(\{pk_S, pk_D\}, m, \sigma)$
```

Fig. 7. Designated verifier signature scheme $\mathsf{RingDVS} = \mathsf{RingDVS}[\mathsf{Ring}]$ constructed from a 2-user ring signature scheme Ring.

Definition 9. *A ring signature scheme* Ring *is* (t, ϵ, n, Q_{Ch})-anonymous against key exposure *if, for any adversary $\mathcal{A}$ with running time at most t, having access to n key pairs, and making at most Q_{Ch} queries to the* $\textsc{Chall}$ *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A}) = 1\right] \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A})$ is as in Figure 6.

It is easy to see that one can transform any $(t, \epsilon, n, 1)$-anonymous (as per Definition 9) ring signature scheme into a $(t, \epsilon \cdot Q_{Ch}, n, Q_{Ch})$-anonymous scheme via a hybrid argument.

The construction. Our construction, denoted RingDVS, is a straightforward adaption of a 2-user ring signature Ring to the DVS setting as detailed in Figure 7. The security of the resulting DVS scheme hinges on the unforgeability and anonymity of the ring signature as per Definitions 8 and 9.

Theorem 1 (Unforgeability of RingDVS). *If* Ring *is a $(t, \epsilon, n + 2, Q_S, Q_{Co})$-unforgeable w.r.t. honest-ring insider corruption 2-user ring signature scheme, then* RingDVS *defined in Figure 7 is (t', ϵ, n, Q_S)-unforgeable, with $t' \approx t$.*

Proof. We reduce the unforgeability of RingDVS to the unforgeability w.r.t. honest-ring insider corruption of Ring.

Initialization of $\mathcal{A}$. The adversary $\mathcal{B}$ against unforgeability of the ring signature receives as input a list $\mathcal{L}$ of honestly generated public keys $\{pk_i\}_{i=1}^{n+2}$. Next, $\mathcal{B}$ corrupts all keys except the first two via its $\textsc{Corr}$ oracle. It sets the first two public keys as challenge keys for $\mathcal{A}$ as $pk_S \leftarrow pk_1$ and $pk_D \leftarrow pk_2$. (Observe that we choose these two indices wlog. for easier bookkeeping.) The reduction then initializes the adversary $\mathcal{A}$ against unforgeability of the DVS on input $(pk_S, pk_D, \{(pk_i, sk_i)\}_{i=3}^{n+2})$.

Queries to Sign. Queries of $\mathcal{A}$ to the Sign oracle are of the form (pk, m). If pk is not one of the honestly generated keys that the reduction gave to $\mathcal{A}$, return $\perp$. For each query, $\mathcal{B}$ queries its own signing oracle on $(1, m, \{pk_1, pk\})$ and returns the answer directly to $\mathcal{A}$. If $pk = pk_2$, record m in Q.

Forgery. At some point, $\mathcal{A}$ outputs a DVS forgery (m^*, σ^*) wrt. pk_S and pk_D. The reduction outputs $(m^*, \sigma^*, pk_1, pk_2)$ as its own forgery.

The reduction soundly simulates the unforgeability game against RingDVS. It simulates the signing oracle truthfully by using its own signing oracle.

If $\mathcal{A}$ outputs a valid DVS forgery wrt. sender key $pk_S = pk_1$ and verifier key $pk_D = pk_2$, the output of $\mathcal{B}$ is a valid ring forgery wrt. the ring $\{pk_1, pk_2\}$ by construction of RingDVS. Furthermore, since $m \notin Q$, $\mathcal{A}$ has not queried its Sign oracle on m and pk_D. Thus, the message-ring pair $(m, \{pk_1, pk_2\})$ was not queried by $\mathcal{B}$ to its oracle either. Lastly, the forgery is wrt. the keys $\{pk_1, pk_2\}$, which $\mathcal{B}$ did not corrupt. Hence, all winning conditions for the ring unforgeability game are met.

The running time t of $\mathcal{B}$ is dominated by the running time t' of $\mathcal{A}$ and we write $t \approx t'$; simulating the signing oracle and querying the corruption oracle n times are not expensive. If $\mathcal{A}$ outputs a successful DVS forgery with probability ϵ, then $\mathcal{B}$ is able to produce a valid ring forgery with the same probability. □

Theorem 2 (Source hiding of RingDVS). *If Ring is a (t, ϵ, n, Q_{Ch})-anonymous against key exposure 2-user ring signature for $n \geq 2$, then RingDVS as shown in Figure 7 is (t', ϵ, Q_{Ch})-source hiding, with $t' \approx t$.*

Proof. We reduce the source hiding of RingDVS to the anonymity against key exposure of Ring.

Initialization of $\mathcal{A}$. The adversary $\mathcal{B}$ against anonymity of the ring signature receives as input a list of honestly generated key pairs $\{(pk_i, sk_i)\}_{i=1}^{n}$. It sets the first two public keys as challenge keys for $\mathcal{A}$ as $pk_S \leftarrow pk_1$ and $pk_D \leftarrow pk_2$. The reduction then initializes the source hiding adversary $\mathcal{A}$ on input (sk_S, pk_S, sk_D, pk_D).

Queries to Chall. $\mathcal{A}$'s queries to the Chall oracle are of the form m. For each of the Q_{Ch} queries, $\mathcal{B}$ forwards the query to its own Chall oracle as $(m, 1, 2, \{pk_1, pk_2\})$ and returns the answer it gets directly to $\mathcal{A}$.

Output. When $\mathcal{A}$ outputs its guess b', the reduction outputs b'.

The reduction soundly simulates the source hiding game against RingDVS for $\mathcal{A}$. The runtime of $\mathcal{B}$ is essentially the runtime of $\mathcal{A}$ plus the runtime to forward the challenge queries and responses and we write $t \approx t'$.

Adversary $\mathcal{A}$ distinguishing between outputs of RingDVS.Sign and RingDVS.Sim amounts to distinguishing between Ring signatures under the two signing keys sk_1 and sk_2 in the ring $\{pk_1, pk_2\}$. Hence, $\mathcal{B}$ inherits $\mathcal{A}$'s winning probability ϵ. □

Implications and the inverse direction. Our construction above establishes that DVS schemes with the security properites needed for this work (i.e., unforgeability and source hiding) can be generically constructed from 2-user ring signatures that provide unforgeability w.r.t. honest-ring insider corruption and anonymity against key exposure. We note that the latter security properties are weaker than those put forward by Bender, Katz, and Morselli [7].

Hashimoto et al. have recently shown in the full version of their work [49] that it is indeed possible to construct also the reverse direction (in contrast to an earlier statement of ours). For their construction each ring member has a signer key pair and a designated verifier key pair. In the signing procedure, depending on the lexicographical order of the signer public keys either DVS.Sign or DVS.Sim is executed generating a ring signature. Verification follows analogously.

Post-quantum ring signature candidates. Several post-quantum ring signature schemes were suggested in the literature. In the following, we list a selection of schemes having concrete instantiations and report on the signature sizes and other practical parameters provided in the corresponding works to illustrate their practicality. All schemes except Raptor (listed first) come with security proofs for unforgeability and anonymity definitions that imply our notions.

Lu, Au, and Zhang [66] introduce Raptor, which uses a chameleon hash function based on the NIST finalist FALCON [77], producing signatures of size approximately 5 KB for a 2-user ring. However, they argue that the best-known attack is inefficient instead of proving unforgeability and anonymity.

Yuen, Esgin, Liu, Au, and Ding [91] propose DualRing-LB (which is a lattice-based instantiation of their generic construction DualRing) with a signature size of 4.4 KB for rings of size 2. They prove anonymity (against full key exposure) of their scheme under a slightly different notion, where only the first-stage attacker has access to a signing oracle and only the second-stage attacker gets the randomness used in creating all keys (i.e., access to the secret keys).

The following two schemes use zero-knowledge proofs based on symmetric primitives, akin to the NIST alternate candidate Picnic [92]: Derler, Ramacher, and Slamanig [27] provide a scheme using NIZK proofs and accumulators. For their smallest reported ring size 2^5, signatures can have a size of 719 KB. Katz, Kolesnikov, and Wang [54] use NIZKPoK with the MPC-in-the-head paradigm. For their smallest ring size 2^7, signing takes 2 seconds and produces signatures of size 285 KB.

In terms of lattice-based constructions, a series of works [38,37,39] by Esgin et al. provide constructions relying on the hardness of M-LWE and M-SIS. The most recent candidate has a signature size of 18 KB for a 2-user ring. A construction by Lyubashevsky, Nguyen, and Seiler [70] relies on (variants of) M-LWE and M-SIS and their smallest signature for rings of size 2^5 is 16 KB. Beullens, Katsumata, and Pintore [8] introduce Falafl that also relies on M-LWE and M-SIS and produces signatures of size 29 KB in less than 100 milliseconds.

Sheikhi-Garjan, Kiliç, and Cenk [82] recently presented an isogeny-based ring signature in which signing and verifying scale in the product nq of the ring size n and isogeny security parameter q.

4 Security Model for Asynchronous Deniable Key Exchange

From a formal perspective, an asynchronous authenticated key exchange protocol is just a traditional authenticated key exchange protocol with a specific type of message flow. In particular, asynchronicity allows one party to post pre-key bundles containing long-term and possibly ephemeral public keys, provided that they can be constructed without knowing the intended partner. We will formalize security for this setting based on a Bellare–Rogaway-type model [3] with implicit authentication and (weak) forward secrecy using post-specified peers [18,57]. The model presented in this section is simplified to deal with basic Bellare–Rogaway-type security with only long-term keys as a warm-up; in Section 7 we present a more granular model that accommodates the complex characteristics found in the Signal protocol handshake, including semi-static keys and stronger security against maximal exposure.

Parties and sessions. Let $\mathcal{P}$ be the set of n_p parties, each of whom has a long-term public-key/secret-key pair generated by an algorithm KGenLT. Each party may run multiple instances of the protocol simultaneously or sequentially, each of which is called a session. The ith session at party P is denoted π_P^i. For each session, the party maintains the following collection of session-specific information:

- $\mathsf{oid} \in \mathcal{P}$: The identity of the session owner.
- $\mathsf{pid} \in \mathcal{P} \cup \{\star\}$: The identity of the intended peer, which may initially be unknown (indicated by $\star$).
- $\mathsf{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$: The role of the party.
- $\mathsf{st}_{\mathsf{exec}} \in \{\bot, \mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$: The status of this session's execution.
- $\mathsf{sid} \in \{0,1\}^* \cup \{\bot\}$: A session identifier defining partnering.
- $\mathsf{cid} \in \{0,1\}^* \cup \{\bot\}$: A contributive identifier, defining a preliminary form of partnering (often as a substring or prefix of the session identifier) for the case the session is not yet bound to an authenticated peer [33].
- $\mathsf{K} \in \mathcal{K}_{\mathsf{KE}} \cup \{\bot\}$: The session key established in this session.
- Any additional protocol-specific data used during execution.

Protocol specification. A 2-party key exchange protocol consists of the following algorithms:

- $\mathsf{KGenLT}() \xrightarrow{\$} (pk, sk)$: A probabilistic long-term key generation algorithm that outputs a public-key/secret-key pair.
- $\mathsf{Run}(sk, \vec{pk}, \pi, m) \xrightarrow{\$} (\pi', m')$: A probabilistic session execution algorithm that takes as input a party's long-term secret key sk, a list of long-term public keys for all honest parties $\vec{pk}$, a session state π, and an incoming message m, and outputs an updated session state π' and a (possibly empty) outgoing message m'. To set up the session sending the first message, Run is called with a distinguished message $m = \mathsf{create}$.

In a deniable key exchange protocol, we will demand the existence of an additional algorithm:

- $\mathsf{Fake}(pk_U, sk_V) \xrightarrow{\$} (\mathsf{K}, \mathsf{T})$: A probabilistic transcript simulation algorithm that takes as input one party's public key and the other party's secret key and generates asession key K and a transcript T of a protocol interaction between them.

Asynchronous key exchange. In principle, a key exchange protocol can have an arbitrary number of message flows, which correspond to multiple calls to Run for a single session. In normal execution of an asynchronous authenticated key exchange protocol, the following three calls to Run occur: 1) a call to Run at the responder (Bob)[5] with $m = \mathsf{create}$, which sets up the responder session and outputs the responder's pre-key bundle, including an ephemeral public key; 2) a call to Run at the initiator with the responder's pre-key bundle (long-term public and ephemeral public keys) which generates a session key and outputs a key exchange message; and 3) a call to Run at the responder with the initiator's long-term public key and key exchange message which generates a session key and has no output message.

Partnering. Two sessions π_U^i and π_V^j are said to be *partners* if they agree on the session identifier ($\pi_U^i.\mathsf{sid} = \pi_V^j.\mathsf{sid}$). An honest partner session is a partner session that is honest, i.e., not under adversarial control.

Session key indistinguishability. The first security property we want of an authenticated key exchange protocol is indistinguishability of session keys. At the start of the security experiment, long-term public-key/secret-key pairs are generated for all n_p honest parties and the public keys $\vec{pk}$ are provided to the adversary, as well as a random challenge bit b_{test} fixed for the duration of the experiment. The adversary is then able to interact with honest parties via the following queries:

– $\textsc{Send}(U, i, m)$: Sends message m to session π_U^i, which corresponds to executing $\mathsf{Run}(sk_U, \vec{pk}, \pi_U^i, m)$, saving the updated session state π' as π_U^i, and returning the outgoing message m' to the adversary.
– $\textsc{CorruptLTKey}(U)$: Returns party U's long-term secret key sk_U to the adversary.
– $\textsc{RevealSessKey}(U, i)$: If session π_U^i has accepted, return its session key $\pi_U^i.\mathsf{K}$ to the adversary.
– $\textsc{Test}(U, i)$: If the $\textsc{Test}$ query has been called before or session π_U^i has not accepted, then return $\perp$. Otherwise, if $b_{\mathsf{test}} = 0$, return $\pi_U^i.\mathsf{K}$, otherwise return an element of $\mathcal{K}_{\mathsf{KE}}$ chosen uniformly at random. Record $\pi^* \leftarrow \pi_U^i$.

The test session $\pi^* = \pi_{U^*}^{i^*}$ is called *fresh* if the following all hold:

1. $\textsc{RevealSessKey}(U^*, i^*)$ was never called.
2. $\textsc{RevealSessKey}(V, j)$ was never called for any V, j such that $\pi^*.\mathsf{sid} = \pi_V^j.\mathsf{sid}$.
3. Either
   (a) there exists an honest partner session π_{p}^* ($\pi_{\mathsf{p}}^*.\mathsf{sid} = \pi^*.\mathsf{sid}$ if π^* is a responder, and $\pi_{\mathsf{p}}^*.\mathsf{cid} = \pi^*.\mathsf{cid}$ if π^* is an initiator), covering weak forward secrecy, or
   (b) $\textsc{CorruptLTKey}(\pi^*.\mathsf{oid})$ and $\textsc{CorruptLTKey}(\pi^*.\mathsf{pid})$ were never called, covering implicit authentication.

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

Deniability. The second security property we want is deniability. At the start of this experiment, long-term public-key/secret-key pairs are generated for all n_p honest parties and the public *and* secret keys are provided to the adversary. A random challenge bit b is fixed for the duration of the experiment. The adversary is given repeated access to a $\textsc{Chall}$ oracle which takes as input two party identifiers U and V. If b is 0, then $\textsc{Chall}$ will generate an honest transcript of an interaction between U and V using the Run algorithm and each party's secret keys. If b is 1, then $\textsc{Chall}$ will generate a simulated transcript of an interaction between U and V using the Fake algorithm. At the end of the experiment, the adversary outputs a guess b' of b. The experiment outputs 1 if $b' = b$ and 0 otherwise. The adversary's advantage in the deniability game is the absolute value of the difference between $\frac{1}{2}$ and the probability the experiment outputs 1.

There are several prior works giving definitions of offline deniability for key exchange [28,23,24,85,86]. Our definition differs from previous ones threefold: Firstly, the challenge oracle executes Run on behalf of the framing party, i.e., we consider semi-honest adversaries only. Secondly, the Fake algorithm (corresponding to the simulator in simulation-based definitions) has access to the receiver's secret key. Thirdly, the adversary (the judge in simulation-based settings) has access to all secret keys. This restricts the deniability to semi-honest adversaries and 1-out-of-2 (one needs a secret key of either party to create a transcript) but lifts us to the so-called big brother setting. The strong point of this deniability notion is that you get some deniability guarantees even against strong judges, who know all secret keys. This models the informal deniability requirement from the Signal specification [72, §4.4]. See Appendix A for a more detailed discussion.

[5]Note that we call Bob the *responder* in our model despite Bob outputting the first, asynchronous key exchange message. Based on the high-level protocol interaction, we deem it more natural to call Alice, who decides to *initiate* a Signal session with Bob, the initiator (in contrast to, e.g., [85,21,86]).

5 Security of the Core Protocol

We now show that our core protocol Π from Figure 2 achieves the security properties defined in Section 4. Key indistinguishability of Π depends on the $\mathsf{IND\text{-}CCA}$ security of the two KEMs, the unforgeability of the DVS, and the security of the KDF; deniability of Π depends on the source hiding of the DVS. Both proofs are in the standard model.

To formally capture Π in the security model of Section 4, we need to specify a few more details:

- Alice takes the $\mathsf{initiator}$ role, Bob the $\mathsf{responder}$ role.
- The transcript in Figure 2 corresponds to the session identifier and consists of the parties' identities and long-term public keys, the responder's ephemeral public key, and the KEM ciphertexts; the contributive identifier corresponds to the pre-key bundle part of the transcript, received by Alice from Bob:

$$\mathsf{transcript} = \mathsf{sid} = (A, B, pk_A^{\mathsf{DVS}}, pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}, epk_B^{\mathsf{KEM}}, c_1, c_2),$$
$$\mathsf{cid} = (B, pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}, epk_B^{\mathsf{KEM}}).$$

Note that the session identifier does not include the DVS signature itself to avoid that the latter needs to be non-malleable (akin to strong unforgeability of regular signatures) [63].

5.1 Key Indistinguishability

Theorem 3 (Key indistinguishability of Π). *Let* DVS *be a* $(t, \epsilon_{\mathsf{DVS}}, n_p, Q_S)$*–unforgeable DVS scheme,* KEM_1 *be a* $(t, \epsilon_{\mathsf{KEM}_1}, n_s)$*–*$\mathsf{IND\text{-}CCA}$*-secure KEM,* KEM_2 *be a* $(t, \epsilon_{\mathsf{KEM}_2}, 1)$*–*$\mathsf{IND\text{-}CCA}$*-secure KEM, and* KDF *be a* $(t, \epsilon_{\mathsf{KDF}}, n_s)$*–*$\mathsf{PRF}$*-secure key derivation function when keyed through either of the key components K_1 and K_2. Then the asynchronous DAKE protocol Π from Figure 2 provides key indistinguishability (as defined in Section 4) in that the advantage ϵ' of any adversary $\mathcal{A}$ running in time $t' \approx t$ is upper bounded as*

$$\epsilon' \le n_s \cdot \begin{pmatrix} n_s \cdot (\epsilon_{\mathsf{KEM}_2} + \epsilon_{\mathsf{KDF}}) \\ + n_p \cdot (\epsilon_{\mathsf{KEM}_1} + \epsilon_{\mathsf{KDF}}) \\ + n_p^2 \cdot (\epsilon_{\mathsf{DVS}} + n_s \cdot (\epsilon_{\mathsf{KEM}_2} + \epsilon_{\mathsf{KDF}})) \end{pmatrix},$$

where $n_s \le Q_{Snd}$ is the maximum number of sessions (upper bounded by the number Q_{Snd} of SEND *queries) and n_p the number of parties.*

Proof. We proceed via a sequence of game hops starting from the key indistinguishability game for an adversary $\mathcal{A}$. We bound the difference between each hop until we reach a game where the adversary's advantage is 0.

Game 0. The initial key indistinguishability game for Π, denoted $\mathcal{G}_0$, letting $\epsilon' := \mathsf{Adv}_{\Pi}^{\mathcal{G}_0}(\mathcal{A}) = |\Pr[\mathcal{G}_0 = 1] - \frac{1}{2}|$.

Game 1 (Guess test session π^*). We first guess the tested session π^* and "invalidate" the game by overwriting the adversary's bit guess with 0 if the adversary calls TEST on a different session. Guessing among the n_s many sessions (where n_s is at most the number Q_{Snd} of calls to the SEND oracle),

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_0}(\mathcal{A}) \le n_s \cdot \mathsf{Adv}_{\Pi}^{\mathcal{G}_1}(\mathcal{A}).$$

For the remaining proof, we distinguish the following three cases for the test session being fresh:

A. There exists an honest partner session π_{p}^* ($\pi_{\mathsf{p}}^*.\mathsf{sid} = \pi^*.\mathsf{sid}$ if π^* is a responder, and $\pi_{\mathsf{p}}^*.\mathsf{cid} = \pi^*.\mathsf{cid}$ if π^* is an initiator).
B. The tested session is an initiator ("Alice") session and CORRUPTLTKEY($\pi^*.\mathsf{pid}$) was never called.[6]
C. The tested session is a responder ("Bob") session and neither CORRUPTLTKEY($\pi^*.\mathsf{oid}$) nor CORRUPTLTKEY($\pi^*.\mathsf{pid}$) was ever called.[7]

Treating theses cases as events in $\mathcal{G}_1$, and writing $\mathcal{G}_1[X]$ to indicate that event X occurs, by the union bound we have:

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_1}(\mathcal{A}) \le \mathsf{Adv}_{\Pi}^{\mathcal{G}_1[\mathrm{A}]}(\mathcal{A}) + \mathsf{Adv}_{\Pi}^{\mathcal{G}_1[\mathrm{B}]}(\mathcal{A}) + \mathsf{Adv}_{\Pi}^{\mathcal{G}_1[\mathrm{C}]}(\mathcal{A}).$$

[6] This is slightly stronger than what freshness condition 3 (b) demands. In the security result for our full SPQR protocol (see Section 6), this is captured more precisely.

[7] In our full SPQR protocol (see Section 6), we will strengthen this case by having Bob use *semi-static* DVS keys. This limits the time window for a key-compromise impersonation (KCI) attack [9] against Bob, as in the Signal handshake [72, §4.6].

Case A (Honest partner). In the first proof case, there exists a session π^*_{p} that agrees with the tested session π^* on the responder's ephemeral KEM public key epk^{KEM} used. We will leverage this to embed a challenge into the ephemeral KEM ciphertext c_2.

Game A.1 (Guess partnered session). We first guess a session π^*_{p} which is partnered via sid (if π^* is a responder) or cid (if π^* is an initiator) to the test session π^*, and let the adversary lose if the guess is incorrect. By this case's prerequisites, (at least) one partner session exists and is guessed with probability at least $1/n_s$, hence

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_1[\mathrm{A}]}(\mathcal{A}) \leq n_s \cdot \mathsf{Adv}_{\Pi}^{\mathcal{G}_{A.1}}(\mathcal{A}).$$

Game A.2 (Ephemeral KEM). We now replace the KEM key K_2 with a random key $\widetilde{K_2}$ in π^* and also in π^*_{p} (unless the latter is a responder and receives a different ciphertext c_2 than sent by π^*).

We bound the difference introduced by this step through a reduction to the $\mathsf{IND\text{-}CCA}$ security of the KEM_2 scheme, which simulates $\mathcal{G}_{A.1}$ truthfully except for the following changes and runs in time $t \approx t'$. It embeds the obtained challenge public key pk into the ephemeral KEM public key epk of the responder session among π^* and π^*_{p}, the challenge ciphertext c^* as c_2 of the initiator session (among π^* and π^*_{p}), and the challenge (real-or-random) key K_b^*) as K_2 into both π^* and π^*_{p}. If π^* is an initiator session, it uses its DECAPS oracle (at most once, i.e., $Q_D \leq 1$) to decrypt a potentially different ciphertext $c_2' \neq c_2 = c^*$ received by π^*_{p}. Depending on the $\mathsf{IND\text{-}CCA}$ KEM challenge bit, the reduction perfectly simulates $\mathcal{G}_{A.1}$ or $\mathcal{G}_{A.2}$, hence

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_{A.1}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_2} + \mathsf{Adv}_{\Pi}^{\mathcal{G}_{A.2}}(\mathcal{A}).$$

Game A.3 (KDF). We finally replace the key derivation function KDF in both π^* and π^*_{p} (in the latter only if it uses $\widetilde{K_2}$) with a random function, in particular replacing the session key K of π^* with a randomly sampled key $\widetilde{K}$.

We bound the introduced advantage difference via a reduction to the pseudorandomness of the key derivation function KDF, treated as a PRF keyed through the second key component K_2 and taking $(K_1, \mathsf{transcript})$ as label. The reduction runs in time $t \approx t'$ and simulates Game $\mathcal{G}_{A.2}$ truthfully, except that it does not sample $\widetilde{K_2}$ itself but instead uses its oracle PRFCHALLENGE to compute the session key values derived from $\widetilde{K_2}$. It calls its oracle at most twice, once for π^* and possibly once for π^*_{p} on a different label, hence $Q_{PRF} \leq n_s$. Depending on whether its oracle output is the true KDF evaluation or that of a random function, the reduction perfectly simulates $\mathcal{G}_{A.2}$ or $\mathcal{G}_{A.3}$, thus

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_{A.2}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}_{\Pi}^{\mathcal{G}_{A.3}}(\mathcal{A}).$$

In Game $\mathcal{G}_{A.3}$, the challenge key K_{test} for π^* is a uniformly random key, independent of b_{test}. Furthermore, by the first two freshness conditions, $\mathcal{A}$ cannot reveal K_{test} via a REVEALSESSKEY query on π^* or any partnered session who might hold the same key. Thus, in $\mathcal{G}_{A.3}$, $\mathcal{A}$ cannot do better than guessing, leaving it with advantage $\mathsf{Adv}_{\Pi}^{\mathcal{G}_{A.3}}(\mathcal{A}) = 0$.

Case B (Initiator tested, peer uncorrupted). In the second proof case, we have that the tested initiator session π^* has an uncorrupted intended peer. We will leverage this to embed a challenge into the static KEM ciphertext c_1.

Game B.1 (Guess responder identity). We first guess the test session's intended peer, $V = \pi^*.\mathsf{pid}$, among the n_p many parties in the game and let the adversary lose if we guess incorrectly. This reduces the adversary's advantage by a factor at most n_p:

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_1[\mathrm{B}]}(\mathcal{A}) \leq n_p \cdot \mathsf{Adv}_{\Pi}^{\mathcal{G}_{B.1}}(\mathcal{A}).$$

Game B.2 (Static KEM). We can now replace the KEM key K_1 in π^* (and any responder session of V receiving the same ciphertext c_1) with a random key $\widetilde{K_1}$.

We bound the advantage difference introduced by this step through a reduction to the $\mathsf{IND\text{-}CCA}$ security of the KEM_1 scheme. The reduction runs in time $t \approx t'$ and simulates $\mathcal{G}_{B.1}$ truthfully, but embeds the obtained challenge public key pk as V's public KEM key pk_V^{KEM} at the outset of the game. It further embeds the challenge ciphertext c^* as c_1 sent by π^* and the challenge (real-or-random) key K_b^*) as K_1 into π^* (and any responder session of V receiving c^*). The reduction uses the DECAPS oracle to decapsulate any ciphertexts $c_1 \neq c^*$ received by sessions of V (calling the oracle at most n_s times), and never has to respond to CORRUPTLTKEY(V) queries as otherwise π^* would not be fresh. Depending on the $\mathsf{IND\text{-}CCA}$ KEM challenge bit, the reduction perfectly simulates $\mathcal{G}_{B.1}$ or $\mathcal{G}_{B.2}$, hence

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_{B.1}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_1} + \mathsf{Adv}_{\Pi}^{\mathcal{G}_{B.2}}(\mathcal{A}).$$

Game B.3 (KDF). We finally replace the key derivation function KDF in π^* (and any other session using $\widetilde{K_1}$) with a random function, in particular replacing the session key K of π^* with a randomly sampled key $\widetilde{K}$.

Analogous to Game $\mathcal{G}_{A.3}$, we can bound the introduced advantage difference by the pseudorandomness of KDF when keyed through the first key component K_1 and taking $(K_2, \mathsf{transcript})$ as label. The challenge static KEM key $\widetilde{K_1}$ may possibly be decapsulated in many responder sessions of V, who use distinct $\mathsf{transcript}$ labels unless they are partnered with π^*; the PRF reduction, running in time $t \approx t'$, may hence make up to n_s queries to its PRFChallenge oracle. Simulating either of the two games in the reduction, we get

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_{B.2}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}_{\Pi}^{\mathcal{G}_{B.3}}(\mathcal{A}).$$

At this point, the challenge key K_{test} for π^* is uniformly random and independent, as only partnered sessions will use the same $\mathsf{transcript}$ label to derive their session keys, but for π^* to be fresh those cannot be revealed. Thus $\mathsf{Adv}_{\Pi}^{\mathcal{G}_{B.3}}(\mathcal{A}) = 0$.

Case C (Responder tested, both parties uncorrupted). In the final proof case, we know that the tested responder session π^* has an uncorrupted intended peer. We will leverage this to ensure that there is a partnered initiator session (which signed the transcript) and then embed a challenge into the ephemeral KEM ciphertext c_2 between these two sessions.

Game C.1 (Guess initiator and responder identities). We first guess the (responder) test session's owner $V = \pi^*.\mathsf{oid}$ and intended (initiator) peer $U = \pi^*.\mathsf{pid}$ among the n_p many parties in the game and "invalidate" the game (overwriting $\mathcal{A}$'s bit guess by 0) if we guess incorrectly. Guessing both parties induces a quadratic loss in n_p:

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_1[\mathrm{C}]}(\mathcal{A}) \leq n_p^2 \cdot \mathsf{Adv}_{\Pi}^{\mathcal{G}_{C.1}}(\mathcal{A}).$$

Game C.2 (Signature unforgeability). We now "invalidate" the game (overwriting $\mathcal{A}$'s bit guess by 0) if the test session π^* accepts a DVS signature σ on a transcript that no session of U has issued.

We bound this event by a reduction against the unforgeability of DVS, running in time $t \approx t'$ and simulating $\mathcal{G}_{C.1}$ with the following modification: Instead of generating parties' DVS keys itself, the reduction embeds the unforgeability game's challenge public keys as $pk_U = pk_S$ and $pk_V = pk_D$, and assigns the additional DVS public-secret key pairs from the unforgeability game's list $\mathcal{L}$ to the remaining parties. (Note that the reduction obtains the secret keys for the latter keys, allowing it to fully simulate those parties.) The reduction uses its signing oracle to compute signatures under $pk_U = pk_S$ (and for any peer public key pk). As U and V remain uncorrupted in this proof case, the reduction never has to answer a CorruptLTKey(U) or CorruptLTKey(V) query. In the case that π^* receives a valid DVS transcript-signature pair $(\mathsf{transcript}, \sigma)$ that no session of U sent (and hence $\mathsf{transcript}$ was not queried to the DVS Sign oracle), the reduction outputs this pair as its forgery and wins. Therefore,

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

Game C.3 (Guess partnered session). As of $\mathcal{G}_{C.2}$, we know that π^* receives a DVS signature on a transcript value $\mathsf{transcript} = \pi^*.\mathsf{sid}$ sent by some session of U. We now guess this (sid-partnered) session π_{p}^* (among the n_s many sessions) and, invalidating the game (overwriting $\mathcal{A}$'s bit guess by 0) upon wrong guess, get

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

Game C.4 (Ephemeral KEM). We next replace the KEM key K_2 with a random key $\widetilde{K_2}$ in π^* and π_{p}^*.

As in Game $\mathcal{G}_{A.2}$, we bound the introduced advantage difference by the $\mathsf{IND\text{-}CCA}$ security of the KEM_2 scheme. The reduction runs in time $t \approx t'$, embeds the challenge pk and c^* into π^*'s ephemeral KEM public key, resp. π_{p}^*'s c_2 ciphertext, and uses the challenge key K_b^* in place of K_2 in both sessions. It does not need to use its Decaps oracle (i.e., $Q_D = 0$), since pk is not used in another session and we are at this point guaranteed that π^*

receives π_{p}^*'s ephemeral ciphertext. (So in fact we only need $\mathsf{IND\text{-}CPA}$ security of KEM_2 here.) The reduction simulates the difference between $\mathcal{G}_{C.3}$ and $\mathcal{G}_{C.4}$, so

$$\mathsf{Adv}_{\Pi}^{\mathcal{G}_{C.3}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_2} + \mathsf{Adv}_{\Pi}^{\mathcal{G}_{C.4}}(\mathcal{A}).$$

Game C.5 (KDF). In the final game hop, we replace KDF in both π^* and π_{p}^* with a random function, replacing the session key K of π^* with a randomly sampled key $\widetilde{K}$.

As in Game $\mathcal{G}_{A.3}$, this is bounded by the pseudorandomness of KDF with key K_2 and label $(K_1, \mathsf{transcript})$. Due to π^* and π_{p}^* agreeing on the $\mathsf{transcript}$ input to KDF, the corresponding reduction only makes one query, $Q_{PRF} = 1 \leq n_s$, running in time $t \approx t'$. Simulating the game difference through this reduction, we get

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

This completes the last proof case, as the challenge key K_{test} for π^* is now uniformly random and independent (beyond partnered sessions), leaving $\mathcal{A}$ with advantage $\mathsf{Adv}_{\Pi}^{\mathcal{G}_{C.5}}(\mathcal{A}) = 0$. □

5.2 Deniability

Observe that we use a different deniability notion compared to prior works as discussed in Section 4. A more thorough discussion of the different deniability notions can be found in Appendix A. In consequence, we can forgo the strong knowledge assumptions that [87,48,29] used to prove deniability of X3DH and their own constructions, respectively. We conjecture that their constructions can likewise be shown to be deniable wrt. our definition without strong knowledge assumptions.

Theorem 4 (Deniability of Π). *Let* $\mathsf{DVS} = (\mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Sim})$ *be a* $(t, \epsilon_{\mathsf{srchid}}, Q_{Ch})$*-source hiding DVS scheme. Then the asynchronous DAKE protocol* Π *from Figure 2 provides deniability (as defined in Section 4) in that the advantage* ϵ' *of any adversary* $\mathcal{A}$ *running in time* $t' \approx t$ *and making up to* Q_{Ch} *challenge queries is upper bounded as* $\epsilon' \leq n_p^2 \cdot \epsilon_{\mathsf{srchid}}$*, where* n_p *is the number of parties.*

Proof. The proof follows by a standard hybrid argument. Let $\mathcal{A}$ be a successful adversary against deniability of Π, then we can construct a reduction $\mathcal{B}$ against the source hiding property of DVS. Observe that $\mathcal{B}$ computes for each of the n_p parties a long-term key pair. It randomly guesses the identifiers of two parties $\mathsf{iid}^*, \mathsf{rid}^* \in [n_p]$ for which $\mathcal{A}$ can distinguish between Run and Fake. Let a number $i \in [n_p^2]$ uniquely denote two independent values $\mathsf{iid}, \mathsf{rid}$ in a query (e.g., encoded as $(\mathsf{iid} - 1) \cdot n_p + \mathsf{rid}$) and let $i^* \in [n_p^2]$ denote the specific guess $\mathsf{iid}^*, \mathsf{rid}^*$ of $\mathcal{B}$. For party iid^*, $\mathcal{B}$ replaces the sampled long-term key with its challenge key pair (pk_S, sk_S) and similarly it replaces the long-term key for party rid^* with (pk_D, sk_D).

In case $\mathcal{A}$ makes a query i for $1 \leq i < i^*$, then $\mathcal{B}$ answers as if $b = 0$, i.e., it runs $\mathsf{DVS.Sign}$. For all $i^* < i \leq n_p^2$, if $\mathcal{A}$ makes a query, then $\mathcal{B}$ answers as if $b = 1$, i.e., it runs $\mathsf{DVS.Sim}$. If $\mathcal{A}$ queries $i = i^*$, then $\mathcal{B}$ passes it to its own oracle. In all cases $\mathcal{B}$ returns the transcript and the session key K to $\mathcal{A}$. Finally, when $\mathcal{A}$ returns its guess bit b', $\mathcal{B}$ returns b' as its guess.

Observe that $\mathcal{B}$ faithfully simulates the deniability game for $\mathcal{A}$. Moreover, the runtime of $\mathcal{B}$ is essentially the runtime of $\mathcal{A}$ plus the runtime to generate the keys and answer the oracle queries.

Now we analyze the winning probability of $\mathcal{A}$ against deniability. For this, we define the hybrids $H_0, \ldots, H_{n_p^2}$ with H_i being the hybrid that answers all challenge queries for indices $1, \ldots, i$ with Run and the challenge queries for indices $i + 1, \ldots, n_p^2$ with Fake. The extreme hybrids are $H_{n_p^2}$, which answers all the challenge queries with Run, and H_0, which answers all queries with Fake. Observe that H_{i-1} and H_i only differ in one execution of Run or Fake. Hence, the probability of distinguishing between H_{i-1} and H_i is bounded by $\epsilon_{\mathsf{srchid}}$. Since there are n_p^2 many hybrids, we overall obtain that $\mathcal{A}$'s probability of winning the deniability game is bounded by $\epsilon' \leq n_p^2 \cdot \epsilon_{\mathsf{srchid}}$. □

6 Signal in a Post-quantum Regime

We now extend our core protocol Π from Figure 2 to capture all the characteristics of the Signal handshake. The core protocol already captures implicit mutual authentication, forward secrecy, offline deniability, and asynchronicity. Signal's X3DH has a few more subtle aspects and security features to consider, which we address in our extended asynchronous DAKE protocol: SPQR (Signal in a Post-Quantum Regime), depicted in Figure 8.

```
KGenLT():
  $(pk^{\mathsf{KEM}}, sk^{\mathsf{KEM}}) \leftarrow\$ \mathsf{KEM}_1.\mathsf{KGen}()$
  $(pk^{\mathsf{DVS}}, sk^{\mathsf{DVS}}) \leftarrow\$ \mathsf{DVS.SKGen}()$
  $tk \leftarrow\$ \mathsf{tPRF.KGen}()$
  $pk \leftarrow (pk^{\mathsf{KEM}}, pk^{\mathsf{DVS}})$
  $sk \leftarrow (sk^{\mathsf{KEM}}, sk^{\mathsf{DVS}}, tk)$
  return $(pk, sk)$

KGenEP():
  return $(epk, esk) \leftarrow\$ \mathsf{KEM}_3.\mathsf{KGen}()$

KGenSS():
  $(sspk^{\mathsf{KEM}}, sssk^{\mathsf{KEM}}) \leftarrow\$ \mathsf{KEM}_2.\mathsf{KGen}()$
  $(sspk^{\mathsf{DVS}}, sssk^{\mathsf{DVS}}) \leftarrow\$ \mathsf{DVS.VKGen}()$
  $sspk \leftarrow (sspk^{\mathsf{KEM}}, sspk^{\mathsf{DVS}})$
  $sssk \leftarrow (sssk^{\mathsf{KEM}}, sssk^{\mathsf{DVS}})$
  return $(sspk, sssk)$
```

```
Alice                                   Signal Server                                   Bob

Initiator Registration                                                 Responder Registration
$(pk_A, sk_A) \leftarrow\$ \mathsf{KGenLT}()$                          $(pk_B, sk_B) \leftarrow\$ \mathsf{KGenLT}()$
                                                                       $(sspk_B, sssk_B) \leftarrow\$ \mathsf{KGenSS}()$

                                                                       Responder Ephemeral Key Generation
Send Pre-Key Bundle to Initiator                                       $(epk_B, esk_B) \leftarrow\$ \mathsf{KGenEP}()$
        $B, pk_B, sspk_B, epk_B$
  <- - - - - - - - - - - - - - - - - - - -

              define: $cid := (B, pk_B, sspk_B, epk_B)$
              define: $sid := (A, B, pk_A, pk_B, sspk_B, epk_B, n, c_1, c_2, c_3)$

Initiator Key Agreement and Protocol Message
  $(sk_A^{\mathsf{KEM}}, sk_A^{\mathsf{DVS}}, tk_A) \leftarrow sk_A$
  $(pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}) \leftarrow pk_B$
  $(sspk_B^{\mathsf{KEM}}, sspk_B^{\mathsf{DVS}}) \leftarrow sspk_B$
  $(n, r) \leftarrow\$ \{0,1\}^\lambda \times \mathcal{R}_{\mathsf{tPRF}}$
  $r_1 \| r_2 \| r_3 \| r_4 \leftarrow \mathsf{tPRF}(tk_A, r)$
  $(K_1, c_1) \leftarrow \mathsf{KEM}_1.\mathsf{Encaps}(pk_B^{\mathsf{KEM}}; r_1)$
  $(K_2, c_2) \leftarrow \mathsf{KEM}_2.\mathsf{Encaps}(sspk_B^{\mathsf{KEM}}; r_2)$
  if $epk_B \neq \bot$
    $(K_3, c_3) \leftarrow \mathsf{KEM}_3.\mathsf{Encaps}(epk_B; r_3)$
  else $(K_3, c_3) \leftarrow (\varepsilon, \varepsilon)$
  $\mathsf{ms} \leftarrow K_1 \| K_2 \| K_3$
  $\sigma \leftarrow \mathsf{DVS.Sign}(sk_A^{\mathsf{DVS}}, sspk_B^{\mathsf{DVS}}, sid; r_4)$
  $K \leftarrow \mathsf{KDF}(\mathsf{ms}, sid)$
  $m \leftarrow (A, pk_A, n, c_1, c_2, c_3, \sigma)$
  return $(K, sid, \mathsf{accepted}, m)$

Responder Key Agreement (on input $m$)
  $(sk_B^{\mathsf{KEM}}, sk_B^{\mathsf{DVS}}, tk_B) \leftarrow sk_B$
  $(sssk_B^{\mathsf{KEM}}, sssk_B^{\mathsf{DVS}}) \leftarrow sssk_B$
  $(pk_A^{\mathsf{KEM}}, pk_A^{\mathsf{DVS}}) \leftarrow pk_A$
  $(sspk_B^{\mathsf{KEM}}, sspk_B^{\mathsf{DVS}}) \leftarrow sspk_B$
  if $\mathsf{DVS.Vrfy}(pk_A^{\mathsf{DVS}}, sspk_B^{\mathsf{DVS}}, sid, \sigma) = \mathsf{false}$
    return $(\bot, \bot, \mathsf{rejected}, \bot)$
  $K_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_B^{\mathsf{KEM}}, c_1)$
  $K_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sssk_B^{\mathsf{KEM}}, c_2)$
  if $esk_B \neq \bot$
    $K_3 \leftarrow \mathsf{KEM}_3.\mathsf{Decaps}(esk_B, c_3)$
  else $(K_3, c_3) \leftarrow (\varepsilon, \varepsilon)$
  $\mathsf{ms} \leftarrow K_1 \| K_2 \| K_3$
  $K \leftarrow \mathsf{KDF}(\mathsf{ms}, sid)$
  return $(K, sid, \mathsf{accepted}, \varepsilon)$

        $m = (A, pk_A, n, c_1, c_2, c_3, \sigma)$
  ---------------------------------------->
```

```
Responder Fake transcript
run Responder Ephemeral Key Generation, and Initiator Key Agreement with a modified randomness sampling and DVS generation:
$(K_1, c_1) \leftarrow\$ \mathsf{KEM}_1.\mathsf{Encaps}(pk_B^{\mathsf{KEM}})$
$(K_2, c_2) \leftarrow\$ \mathsf{KEM}_2.\mathsf{Encaps}(sspk_B^{\mathsf{KEM}})$
if $epk_B \neq \bot$    $(K_3, c_3) \leftarrow\$ \mathsf{KEM}_3.\mathsf{Encaps}(epk_B)$
else    $(K_3, c_3) \leftarrow (\varepsilon, \varepsilon)$
$\sigma \leftarrow\$ \mathsf{DVS.Sim}(sssk_B^{\mathsf{DVS}}, pk_A^{\mathsf{DVS}}, sid)$
$K \leftarrow \mathsf{KDF}(\mathsf{ms}, sid)$
return $(K, m = (B, pk_B, sspk_B, epk_B, A, pk_A, n, c_1, c_2, c_3, \sigma))$
```

Fig. 8. The SPQR protocol (top: key generation, middle: protocol flow, bottom: fake transcript generation), combining static, semi-static and ephemeral key encapsulation schemes KEM_1, KEM_2, and KEM_3, a designated verifier signature DVS, and a twisted pseudorandom function tPRF.

Semi-static keys In Signal, asynchronicity is facilitated by a central, untrusted server which stores the users' pre-key bundles. To enable multiple users to asynchronously contact some responder user, say Bob, the latter uploads multiple ephemeral public pre-keys to the Signal server, of which one is handed to any initiator session that wants to contact Bob (along with the other pre-key bundle elements) and then deleted from the Signal server.

Bob will periodically upload new ephemeral pre-keys; however, if Bob has been offline for a long time, those pre-keys may run out. Therefore, the Signal protocol also includes a *semi-static* key in user pre-key bundles, and always includes key derivations based on that semi-static key. If the Signal server runs out of ephemeral pre-keys, the corresponding key share is not derived and left out; in that case the semi-static key share still provides delayed forward secrecy [14]. We capture this similarly in SPQR by encapsulating a key-ciphertext pair (K_3, c_3) against Bob's ephemeral KEM public key epk_B only if the latter is present.

Maximal-exposure security Signal aims for very strong security guarantees, considering beyond long-term and session key compromise and also compromise of semi-static and ephemeral keys (via the randomness of sessions) [17,61,21]. We model this in an accordingly strong key exchange model (in Section 7) and prove (in Section 8) that SPQR achieves equivalent security in the post-quantum setting as Signal does in the classical setting. In particular, we show that session keys remain secret, as long as any of the (Alice–Bob) secret combinations ephemeral–ephemeral, ephemeral–semi-static, ephemeral–long-term, and long-term–semi-static are uncompromised. Secrecy from the first three is straightforwardly achieved via encapsulations against the corresponding ephemeral, semi-static, and long-term KEM keys of Bob. To achieve secrecy from the last one (i.e., when all initiator randomness is compromised), beyond relying on the DVS scheme for initiator authentication, we apply a NAXOS-like [61] trick to extract randomness from Alice's long-term secrets via a twisted PRF [43,59]. Twisted PRFs can be generically instantiated from regular PRFs (see Section 2.3) and yield output indistinguishable from random as long as a session's long-term secret *or* randomness is uncompromised.

We present our formal security results for SPQR in Section 8 after introducing the full security model next.

7 Full Security Model

In this section we present the extensions to the core asynchronous DAKE model that we will use to prove our post-quantum Signal construction SPQR depicted in Figure 8 secure. The key-indistinguishability game is fully specified in Figure 9 and the deniability game in Figure 10. The main differences to the core security models are as follows

- Signal employs *semi-static* keys; these keys are authenticated via signatures using the long-term key of the respective party, reused, and updated regularly. We thus establish multiple of these keys for each party, identifying each key pair uniquely via an identifier $\mathsf{ssid} \in [n_{ss}]$. Sessions receive semi-static keys in an authenticated manner in the model (just like long-term keys). The adversary is able to corrupt semi-static keys of a user U via the $\textsc{CorruptSSKey}(U, \mathsf{ssid})$ oracle, similar to the corruption of long-term keys via $\textsc{CorruptLTKey}(U)$.
- The usage of ephemeral pre-keys is optional in Signal (as the pre-keys stored on the Signal server may run out). We model this by introducing two types of sessions, full and $\mathsf{reduced}$, depending on whether an ephemeral pre-key is received by the initiator in the pre-key bundle or not.
- The adversary is now granted maximal-exposure capabilities by also revealing the randomness used in a party's execution Run. To this end, we make the used randomness explicit in syntax via the session state variable coins, which during setup of the session samples random coins from the appropriate randomness spaces. The adversary then has access to a $\textsc{RevealRandom}(U, i)$ oracle that returns the coins sampled in session π_U^i, and marks them as revealed via a flag $\mathsf{revrand}$.

Full protocol specification. We adapt the syntax to account for semi-static and ephemeral keys for an asynchronous deniable authenticated key exchange protocol $\mathsf{KE} = (\mathsf{KGenLT}, \mathsf{KGenSS}, \mathsf{KGenEP}, \mathsf{Run}, \mathsf{Fake})$:

- $\mathsf{KGenLT}() \overset{\$}{\rightarrow} (pk_U, sk_U)$: As before, a probabilistic long-term key generation algorithm that outputs a public-key/secret-key pair.
- $\mathsf{KGenSS}() \overset{\$}{\rightarrow} (sspk_U^{\mathsf{ssid}}, sssk_U^{\mathsf{ssid}})$: A probabilistic semi-static key generation algorithm that outputs a public-key/secret-key pair of user U with semi-static identifier ssid.
- $\mathsf{KGenEP}() \overset{\$}{\rightarrow} (epk_U^{\mathsf{epid}}, esk_U^{\mathsf{epid}})$: A probabilistic ephemeral key generation algorithm that outputs a public-key/secret-key pair of user U with ephemeral identifier epid.

– $\mathsf{Run}(sk_U, \vec{sssk}_U, \vec{pk}, \vec{sspk}, \pi, m) \overset{\$}{\rightarrow} (\pi', m')$: A probabilistic session execution algorithm that takes as input a party's long-term secret key sk_U, a list of that party's semi-static secret keys $\vec{sssk}_U$, lists of long-term and semi-static public keys for all honest parties $\vec{pk}$ and $\vec{sspk}$, a session state π, and an incoming message m, and outputs an updated session state π' and a (possibly empty) outgoing message m'. To set up the session sending the first message, Run is called with a distinguished message $m = \mathsf{create}$.
– $\mathsf{Fake}(pk_U, sk_V, \vec{sssk}_V, \mathsf{ssid}) \overset{\$}{\rightarrow} (\mathsf{K}, \mathsf{T})$: A probabilistic transcript simulation algorithm that takes as input one party's long-term public key, the other party's long-term secret key, a list of the other party's semi-static secret keys, and an identifier for a semi-static key and generates a session key K and a transcript T of a protocol interaction between them, where the semi-static key ssid of V is used.

In our case, only the responder uses a semi-static key. If both parties use semi-static keys, the Fake algorithm would take two semi-static identifiers as argument to denote which semi-static key to use for either party. For the sake of simplicity we omit this in the following.

7.1 Key Indistinguishability

Definition 10. *An asynchronous DAKE key exchange protocol* KE *is* $(t, \epsilon, (Q_{Snd}, Q_{CorrLT}, Q_{CorrSS}, Q_{RevR}, Q_{RevSK}))$–key-indistinguishable *if for any adversary* $\mathcal{A}$ *with running time at most* t*, we have that*

$$\mathsf{Adv}^{\mathsf{adake\text{-}kind}}_{\mathsf{KE}}(\mathcal{A}) = \left| \Pr\left[\mathcal{G}^{\mathsf{adake\text{-}kind}}_{\mathsf{KE}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{adake\text{-}kind}}_{\mathsf{KE}}(\mathcal{A})$ *is defined in Figure 9 and* Q_x *for* $x \in \{Snd, CorrLT, CorrSS, RevR, RevSK\}$ *denotes the number of queries to the oracles* Send, CorruptLTKey, CorruptSSKey, RevealRandom *and* RevealSessKey*, respectively. The model restricts the adversary to a single query to the* Test *oracle.*

In addition to the state variables given for the core protocol in Section 4, the following protocol-specific variables are introduced:

– $\mathsf{ssid} \in [n_{ss}]$ denotes the identifier of the responder's semi-static key used in this session. If $\pi.\mathsf{role} = \mathsf{initiator}$ this refers to $sspk^{\mathsf{ssid}}_{\mathsf{pid}}$, if $\pi.\mathsf{role} = \mathsf{responder}$ this refers to $sspk^{\mathsf{ssid}}_{\mathsf{oid}}$.
– $\mathsf{type} \in \{\mathsf{full}, \mathsf{reduced}\}$ indicates whether an ephemeral pre-key was included in the pre-key bundle, or not. Setting $\mathsf{type} = \mathsf{full}$ indicates that an ephemeral pre-key has been received and used by the initiator, whereas $\mathsf{type} = \mathsf{reduced}$ means that no ephemeral pre-key has been received resp. used by the initiator.
– $\mathsf{coins} \in \mathcal{R}_{\mathsf{KE}}$ denotes the random coins from the randomness space $\mathcal{R}_{\mathsf{KE}}$ used in the execution of Run.
– $\mathsf{revrand} \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the random coins $\pi.\mathsf{coins}$ have been revealed via a RevealRandom query. The default value is false.

In order to fully describe the security game $\mathcal{G}^{\mathsf{adake\text{-}kind}}_{\mathsf{KE}}(\mathcal{A})$ that is played between the adversary and the challenger, we introduce the following game-specific flags associated with a user $U \in [n_p]$. They indicate whether a party U's long-term or one of it its semi-static secret keys have been compromised by the adversary:

– $\mathsf{corrltk}_U \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the long-term secret key of party U has been compromised by the adversary via a CorruptLTKey(U) query. The default value is false.
– $\mathsf{corrssk}^{\mathsf{ssid}}_U \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether the semi-static secret key with index ssid of party U has been compromised by the adversary via a CorruptSSKey query. The default value is false.

Session Partnering and Correctness. As in the core model, (full) session partnering is defined via session identifiers: We say that a session π^i_U owned by U is *partnered* with a session π^j_V owned by V if they agree on the session identifier, i.e., $\pi^i_U.\mathsf{sid} = \pi^j_V.\mathsf{sid} \neq \bot$. In order to identify sessions which may eventually derive the same key but are not fully partnered (yet), we have introduced the concept of *contributive identifiers* cid. More precisely, we say that a session π^i_U owned by U is *contributively partnered* with a session π^j_V owned by V, if they agree on their contributive session identifier, i.e., whenever $\pi^i_U.\mathsf{cid} = \pi^j_V.\mathsf{cid} \neq \bot$.

We say that an asynchronous authenticated key exchange protocol KE with randomness space $\mathcal{R}_{\mathsf{KE}}$ is *correct* if any protocol execution between honest parties without interference by the adversary results in two sessions which accept with the same session key and session identifier.

```
$\mathcal{G}^{\text{adake-kind}}_{\text{KE}}(\mathcal{A})$:
$b_{\text{test}} \leftarrow\!\!\$\ \{0,1\}$ //sample test bit
$\pi^* \leftarrow \perp$ //variable for test session
for $U \in [n_p]$
    $(pk_U, sk_U) \leftarrow\!\!\$\ \mathsf{KGenLT}()$ //long-term key generation
    for $\mathsf{ssid} \in [n_{ss}]$
        $(sspk_U^{\mathsf{ssid}}, sssk_U^{\mathsf{ssid}}) \leftarrow\!\!\$\ \mathsf{KGenSS}()$ //semi-static key generation
$\vec{pk} \leftarrow \{pk_U\}_{U \in [n_p]}$;    $\vec{sspk} \leftarrow \{sspk_U^{\mathsf{ssid}}\}_{U \in [n_p]}^{\mathsf{ssid} \in [n_{ss}]}$
$b' \leftarrow\!\!\$\ \mathcal{A}(\vec{pk}, \vec{sspk})$ //run adversary
if $\mathsf{sound}() = \mathsf{false}$ //adversary wins if it breaks soundness
    return 1
if $\mathsf{fresh}(\pi^*) = \mathsf{false}$ //attack invalid if test session is not fresh
    $b' \leftarrow 0$
return $[\![b' = b_{\text{test}}]\!]$ //determine win or loss
```

```
$\mathsf{fresh}(\pi^*)$:
if $\pi^*.\mathsf{revealed} = \mathsf{true}$  then return false //test session is revealed
if $\exists\ \pi_V^j \neq \pi^* : (\pi_V^j.\mathsf{sid} = \pi^*.\mathsf{sid} \land \pi_V^j.\mathsf{revealed} = \mathsf{true})$  then return false
    //test session's partner is revealed
return $\big(\pi^*.\mathsf{type} = \mathsf{full}$ and $\mathsf{clean}_{\mathsf{full}}(\pi^*)\big)$
    //test session in full handshake mode and test session key is clean
  or $\big(\pi^*.\mathsf{type} = \mathsf{reduced}$ and $\mathsf{clean}_{\mathsf{reduced}}(\pi^*)\big)$
    //test session in reduced handshake mode and test session key is clean
```

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

```
$\textsc{Send}(U, i, m)$:
if $\pi_U^i = \perp$ //initiate session: for responders, $m = \mathsf{create}$ carries semi-static key identifier
    $\pi_U^i.\mathsf{oid} \leftarrow U$ //set owner identity
    if $m = \mathsf{create}$  then
        $\pi_U^i.\mathsf{role} \leftarrow \mathsf{responder}$;    $\pi_U^i.\mathsf{ssid} \leftarrow m.\mathsf{ssid}$
        //set responder role and semi-static key identifier (carried in $m$)
    else $\pi_U^i.\mathsf{role} \leftarrow \mathsf{initiator}$ //set initiator role ($m$ is first protocol message)
    $\pi_U^i.\mathsf{coins} \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{KE}}$ //sample session randomness
    $\pi_U^i.\mathsf{st}_{\mathsf{exec}} \leftarrow \mathsf{running}$
$(\pi_U^i, m') \leftarrow \mathsf{Run}(sk_U, \vec{sssk}_U, \vec{pk}, \vec{sspk}, \pi_U^i, m; \pi_U^i.\mathsf{coins})$
//run session, with explicit random coins
return $(m', \pi_U^i.\mathsf{st}_{\mathsf{exec}})$ //return message and session state
```

```
$\textsc{Test}(U, i)$:
if $\pi_U^i = \perp$ or $\pi_U^i.\mathsf{st}_{\mathsf{exec}} \neq \mathsf{accepted}$ or $\pi^* \neq \perp$
//session does not exist, has not accepted yet, or test already asked
    return $\perp$
$\pi^* \leftarrow \pi_U^i$ //record test session
if $b_{\text{test}} = 0$
    $K_{\text{test}} \leftarrow \pi_U^i.\mathsf{K}$ //real session key
else
    $K_{\text{test}} \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{KE}}$ //random key from key space
return $K_{\text{test}}$ //return challenge key
```

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

```
$\textsc{CorruptSSKey}(U, \mathsf{ssid})$:
$\mathsf{corrssk}_U^{\mathsf{ssid}} \leftarrow \mathsf{true}$ //mark semi-static key as corrupted
return $sssk_U^{\mathsf{ssid}}$ //return semi-static secret key
```

```
$\textsc{RevealRandom}(U, i)$:
if $\pi_U^i = \perp$  then return $\perp$ //session does not exist
$\pi_U^i.\mathsf{revrand} \leftarrow \mathsf{true}$ //mark randomness as revealed
return $\pi_U^i.\mathsf{coins}$ //return session's random coins
```

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

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

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

```
$\mathsf{clean}_{\mathsf{EE}}(\pi^*)$:
return $\pi^*.\mathsf{revrand} = \mathsf{false}$ and $\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$
    //randomness of test session is unrevealed and ephemeral contribution of peer is clean
```

```
$\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$:
return
    $\big(\pi^*.\mathsf{role} = \mathsf{initiator}$ and $\exists \pi \neq \pi^*$ :
        $(\pi.\mathsf{role} = \mathsf{responder}$ and $\pi^*.\mathsf{cid} = \pi.\mathsf{cid}$ and $\pi.\mathsf{revrand} = \mathsf{false})\big)$
    //there exists a contributively partnered responder session exists whose randomness is unrevealed
or $\big(\pi^*.\mathsf{role} = \mathsf{responder}$ and $\exists \pi \neq \pi^*$ :
        $(\pi.\mathsf{role} = \mathsf{initiator}$ and $\pi^*.\mathsf{sid} = \pi.\mathsf{sid}$ and $\pi.\mathsf{revrand} = \mathsf{false})\big)$
    //there exists a partnered initiator session (which is unique by sound) whose randomness is unrevealed
```

```
$\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$:
return
    $\big(\pi^*.\mathsf{role} = \mathsf{initiator}$ and $\mathsf{corrltk}_{\pi^*.\mathsf{oid}} = \mathsf{false}$ and $\mathsf{corrssk}_{\pi^*.\mathsf{pid}}^{\pi^*.\mathsf{ssid}} = \mathsf{false}\big)$
    //long-term secret of initiator test session and semi-static key of responder peer are uncorrupted
or $\big(\pi^*.\mathsf{role} = \mathsf{responder}$ and $\mathsf{corrltk}_{\pi^*.\mathsf{pid}} = \mathsf{false}$ and $\mathsf{corrssk}_{\pi^*.\mathsf{oid}}^{\pi^*.\mathsf{ssid}} = \mathsf{false}\big)$
    //long-term secret of responder peer and semi-static key of initiator test session are uncorrupted
```

```
$\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$:
return
    $\big(\pi^*.\mathsf{role} = \mathsf{initiator}$ and $\pi^*.\mathsf{revrand} = \mathsf{false}$ and $\mathsf{corrltk}_{\pi^*.\mathsf{pid}} = \mathsf{false}\big)$
    //randomness of initiator test session is unrevealed and long-term secret of responder peer is uncorrupted
or $\big(\pi^*.\mathsf{role} = \mathsf{responder}$ and $\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$ and $\mathsf{corrltk}_{\pi^*.\mathsf{oid}} = \mathsf{false}\big)$
    //long-term secret of responder test session is uncorrupted and ephemeral contribution of initiator peer is clean
```

```
$\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$:
return
    $\big(\pi^*.\mathsf{role} = \mathsf{initiator}$ and $\pi^*.\mathsf{revrand} = \mathsf{false}$ and $\mathsf{corrssk}_{\pi^*.\mathsf{pid}}^{\pi^*.\mathsf{ssid}} = \mathsf{false}\big)$
    //randomness of initiator test session is unrevealed and semi-static secret of responder peer is uncorrupted
or $\big(\pi^*.\mathsf{role} = \mathsf{responder}$ and $\mathsf{clean}_{\mathsf{peerE}}(\pi^*)$ and $\mathsf{corrssk}_{\pi^*.\mathsf{oid}}^{\pi^*.\mathsf{ssid}} = \mathsf{false}\big)$
    //semi-static secret of responder test session is uncorrupted and ephemeral contribution of initiator peer is clean
```

Fig. 9. Key indistinguishability game $\mathcal{G}^{\text{adake-kind}}_{\text{KE}}(\mathcal{A})$ for key exchange protocol KE against adversary $\mathcal{A}$ with access to oracles Send, Test, CorruptLTKey, CorruptSSKey, RevealRandom, and RevealSessKey; composed of the main game (top section), oracles (middle section), and clean predicates defining freshness (bottom section). Without loss of generality, we assume that all queries that the adversary makes to the oracles are *well-defined* and *valid*, i.e., of the expected type and in the appropriate ranges.

Soundness. Soundness, captured in the predicate sound, describes the behavior with respect to a correct protocol execution. If an adversary $\mathcal{A}$ manages to create one of the following situations, it will win the game immediately:

(i) Two sessions accept with the same session identifier, but derive different session keys, indicate different handshake types (full vs. $\mathsf{reduced}$), or do not agree on their contributive identifiers (Fig. 9, Line 18).
(ii) Two initiator sessions accept with the same session identifier (Fig. 9, Line 19).
(iii) Three sessions accept with the same session identifier in full handshake type (Fig. 9, Line 20).

Freshness. Granting the adversary $\mathcal{A}$ access to the oracles described in Figure 9 without restriction would allow the adversary to trivially win the game, e.g., by testing a session key and then revealing it or corrupting all secrets used in the key derivation of a session. Therefore, as in the core model, we require the tested session to be *fresh* and for this introduce the predicate fresh (cf. Figure 9) which takes as input the test session π^* and prohibits all "trivial wins". On a high level, the session key derived in the test session is considered to be *fresh* if the following criteria hold:

(i) The session key of the test session has not been revealed to $\mathcal{A}$ via a RevealSessKey query (Fig. 9, Line 14).
(ii) The session key of any partnered session (i.e., any session with the same session identifier as the test session) has not been revealed to $\mathcal{A}$ via a RevealSessKey query (Fig. 9, Line 15).
(iii) $\mathcal{A}$ has not obtained sufficiently many secrets to derive the session key of the test session itself via CorruptLTKey and/or CorruptSSKey and/or RevealRandom queries (Fig. 9, Line 16).

Clean keys Following the terminology of Cohn-Gordon, Cremers, Dowling, Garratt, and Stebila [21] the last criterion of freshness is captured by a series of so-called clean predicates, which we discuss next. The formal description can also be found in Figure 9. Let π^* denote the test session. Depending on whether an ephemeral pre-key was used in the key derivation of π^* or not, we apply either the $\mathsf{clean}_{\mathsf{full}}$ or the $\mathsf{clean}_{\mathsf{reduced}}$ predicate to π^*.

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

$\mathsf{clean}_{\mathsf{LTSS}}$: This predicate indicates whether the combination of the long-term key of the initiator and the semi-static key of the responder is unknown to the adversary.

$\mathsf{clean}_{\mathsf{ELT}}$: This predicate indicates whether the combination of the ephemeral contribution[8] of the initiator and the long-term key of the responder is unknown to the adversary.

$\mathsf{clean}_{\mathsf{ESS}}$: This predicate indicates whether the combination of the ephemeral contribution of the initiator and the semi-static key of the responder is unknown to the adversary.

If the test session π^* is a responder session, the evaluation of $\mathsf{clean}_{\mathsf{ELT}}$ and $\mathsf{clean}_{\mathsf{ESS}}$ necessitates a further predicate called $\mathsf{clean}_{\mathsf{peerE}}$ (in all other cases, it is sufficient to consider the flags $\mathsf{corrltk}, \mathsf{corrssk}$, and $\mathsf{revrand}$, respectively). For responder test sessions, $\mathsf{clean}_{\mathsf{peerE}}$ indicates whether the randomness used in any partnered initiator session π^*_{p} (if test session responder) is unknown to the adversary.

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

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

Again, the predicate $\mathsf{clean}_{\mathsf{peerE}}$ helps to determine within $\mathsf{clean}_{\mathsf{EE}}$ whether the randomness of the test session's (contributive) partners is unrevealed or uncorrupted, respectively.

Main differences to the model in [21]. Our authenticated key exchange model closely follows the one used in the original Signal analysis of Cohn-Gordon et al. [21]. We make the following modifications: Since we are only concerned about the initial key agreement and not the subsequent symmetric and asymmetric ratcheting stages, we can forgo the notion of *multi-stage* AKE security, where multiple sessions keys are derived in a series of stages. Lastly, we explicitly take the *deniability* feature of Signal into account in a separate notion to avoid establishing a post-quantum solution that forgoes a key requirement of the specification.

[8] Recall that the ephemeral contribution in initiator sessions is determined by the session specific randomness $\mathsf{coins} \in \mathcal{R}_{\mathsf{KE}}$.

```
$\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{KE}}(\mathcal{A})$:
$\mathcal{L} \leftarrow \emptyset$ //list of keys for the adversary
for $U \in [n_p]$
    $(pk_U, sk_U) \leftarrow\$ \mathsf{KGenLT}()$ //long-term key generation
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(pk_U, sk_U)\}$
    for $\mathsf{ssid} \in [n_{ss}]$
        $(sspk^{\mathsf{ssid}}_U, sssk^{\mathsf{ssid}}_U) \leftarrow\$ \mathsf{KGenSS}()$ //semi-static key generation
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{(sspk^{\mathsf{ssid}}_U, sssk^{\mathsf{ssid}}_U)\}$
$\vec{pk} \leftarrow \{pk_U\}_{U\in[n_p]}$;   $\vec{sspk} \leftarrow \{sspk^{\mathsf{ssid}}_U\}^{\mathsf{ssid}\in[n_{ss}]}_{U\in[n_p]}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\textsc{Chall}}(\mathcal{L})$
return $[\![b' = b]\!]$

$\textsc{Chall}(\mathsf{iid}, \mathsf{rid}, \mathsf{ssid})$:
if $b = 0$
    $\pi_{\mathsf{rid}}.\mathsf{role} \leftarrow \mathsf{responder}$;   $\pi_{\mathsf{rid}}.\mathsf{st}_{\mathsf{exec}} \leftarrow \mathsf{running}$
    //initialize session variables
    $\pi_{\mathsf{iid}}.\mathsf{role} \leftarrow \mathsf{initiator}$;   $\pi_{\mathsf{iid}}.\mathsf{st}_{\mathsf{exec}} \leftarrow \mathsf{running}$
    $(\pi'_{\mathsf{rid}}, m) \leftarrow\$ \mathsf{Run}(sk_{\mathsf{rid}}, \vec{sssk}_{\mathsf{rid}}, \vec{pk}, \vec{sspk}, \pi_{\mathsf{rid}}, (\mathsf{create}, \mathsf{ssid}))$
    //build pre-key bundle
    $(\pi'_{\mathsf{iid}}, m') \leftarrow\$ \mathsf{Run}(sk_{\mathsf{iid}}, \vec{sssk}_{\mathsf{iid}}, \vec{pk}, \vec{sspk}, \pi_{\mathsf{iid}}, m)$
    //initiator sends message
    $(K, \mathsf{T}) \leftarrow (\pi'_{\mathsf{iid}}.K, (m, m'))$
    //save session key and transcript
else
    $(K, \mathsf{T}) \leftarrow\$ \mathsf{Fake}(pk_{\mathsf{iid}}, sk_{\mathsf{rid}}, \vec{sssk}_{\mathsf{rid}}, \mathsf{ssid})$
return $(K, \mathsf{T})$
```

Fig. 10. Security game for deniability of an asynchronous DAKE protocol KE against an adversary $\mathcal{A}$.

7.2 Deniability

Definition 11. *An asynchronous DAKE protocol* KE *is* (t, ϵ, Q_{Ch})-deniable *if for any adversary* $\mathcal{A}$ *with running time at most* t *and making at most* Q_{Ch} *many queries to its* $\textsc{Chall}$ *oracle, we have that*

$$\mathsf{Adv}^{\mathsf{adake\text{-}den}}_{\mathsf{KE}}(\mathcal{A}) = \left|\Pr\left[\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{KE}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \leq \epsilon,$$

where $\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{KE}}(\mathcal{A})$ *is defined in Figure 10.*

The main difference between the textual description of the deniability game in Section 5.2 and Figure 10 is the use of semi-static keys. Here, we generate n_{ss} many semi-static keys per party, all of which are given to the adversary. When querying the challenge oracle, the adversary may choose the semi-static key that the responder uses. The pre-key bundle of the responder may depend on the semi-static key. The Initiator key agreement and the Fake algorithm also use the semi-static key as specified by KE.

8 SPQR Security Proofs

In this section we present the security results for our SPQR protocol (Figure 8) via theorem statements and detailed proofs of both key-indistinguishability and deniability in the previously described security model (Section 7).

Before we start, we translate the informal protocol description of SPQR given in Figure 8 into the syntax of our model. The resulting protocol flow is depicted in Figure 11.

8.1 Key Indistinguishability

Theorem 5 (Key indistinguishability of SPQR). *Let* DVS *be a* $(t, \epsilon_{\mathsf{DVS}}, n_p \cdot n_{ss}, Q_S)$*–unforgeable DVS scheme.*

Let KEM_1 *be a* $(t, \epsilon_{\mathsf{KEM}_1}, n_s)$–$\mathsf{IND\text{-}CCA}$*-secure KEM,* KEM_2 *be a* $(t, \epsilon_{\mathsf{KEM}_2}, n_s)$–$\mathsf{IND\text{-}CCA}$*-secure KEM,* KEM_3 *be a* $(t, \epsilon_{\mathsf{KEM}_3}, 1)$–$\mathsf{IND\text{-}CCA}$*-secure KEM with randomness space* $\mathcal{R}_{\mathsf{KEM}_3}$*, and* δ_{corr} *be the maximal correctness error among* KEM_1*,* KEM_2*, and* KEM_3*.*

Let KDF *be a* $(t, \epsilon_{\mathsf{KDF}}, n_s)$–$\mathsf{PRF}$*-secure key derivation function when keyed through any key component* K_1*,* K_2*,* K_3*, and* tPRF *a* $(t, \epsilon_{\mathsf{tPRF}}, n_s)$*-secure twisted pseudorandom function with label space* $\mathcal{R}_{\mathsf{tPRF}}$*.*

Then the SPQR *protocol with randomness space* $\mathcal{R}_{\mathsf{KE}} = \{0,1\}^\lambda \times \mathcal{R}_{\mathsf{tPRF}} \times \mathcal{R}_{\mathsf{KEM}_3}$ *as shown in Figure 8 and formalized in Figure 11 provides* $(t', \epsilon', (Q_{Snd}, Q_{CorrLT}, Q_{CorrSS}, Q_{RevR}, Q_{RevSK}))$*–key indistinguishability (formalized in Figure 11) for* $t \approx t'$ *and*

$$\epsilon' \leq \frac{n_s^2}{2^\lambda} + \frac{n_s^2}{2^{|\mathcal{R}_{\mathsf{tPRF}}|}} + \frac{n_s^2}{2^{|\mathcal{R}_{\mathsf{KEM}_3}|}} + 3n_s \cdot \delta_{\mathsf{corr}} \\ + n_s \cdot n_p^2 \cdot \begin{pmatrix} n_{ss} \cdot \left(\epsilon_{\mathsf{DVS}} + 2n_s \cdot (\epsilon_{\mathsf{tPRF}} + \epsilon_{\mathsf{KEM}_2} + \epsilon_{\mathsf{KDF}})\right) \\ + n_s \cdot \left(2\epsilon_{\mathsf{tPRF}} + \epsilon_{\mathsf{KEM}_1} + \epsilon_{\mathsf{KEM}_3} + 2\epsilon_{\mathsf{KDF}}\right) \end{pmatrix},$$

where $n_s \leq Q_{Snd}$ *is the maximum number of sessions (upper bounded by the number* Q_{Snd} *of* $\textsc{Send}$ *queries),* n_p *the number of parties, and* n_{ss} *the number of semi-static keys per party.*

Alice **Bob**

Bob:
```
$\mathsf{Run}(sk_B, \vec{pk}, \vec{sspk}, \pi_B, (\mathsf{create}, \mathsf{ssid}))$
$\pi_B.\mathsf{pid} \leftarrow \star$
$(\bot, \bot, r') \leftarrow \pi_B.\mathsf{coins}$
$(epk_B, esk_B) \leftarrow \mathsf{KGenEP}(; r')$
$\pi_B.\mathsf{cid} \leftarrow (B, pk_B, sspk_B^{\mathsf{ssid}}, epk_B)$
return $(\pi_B, m = (B, \mathsf{ssid}, epk_B))$
```

$$\xleftarrow{\quad m \quad}$$

Alice:
```
$\mathsf{Run}(sk_A, \vec{sssk}_A, \vec{pk}, \vec{sspk}, \pi_A, m)$
$(sk_A^{\mathsf{KEM}}, sk_A^{\mathsf{DVS}}, tk_A) \leftarrow sk_A$
$(B, \mathsf{ssid}, epk_B) \leftarrow m$
$(pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}) \leftarrow pk_B$
$(sspk_B^{\mathsf{KEM}}, sspk_B^{\mathsf{DVS}}) \leftarrow sspk_B^{\mathsf{ssid}}$
$(n, r, \bot) \leftarrow \pi_A.\mathsf{coins}$
$r_1 \| r_2 \| r_3 \| r_4 \leftarrow \mathsf{tPRF}(tk_A, r)$
$(K_1, c_1) \leftarrow \mathsf{KEM}_1.\mathsf{Encaps}(pk_B^{\mathsf{KEM}}; r_1)$
$(K_2, c_2) \leftarrow \mathsf{KEM}_2.\mathsf{Encaps}(sspk_B^{\mathsf{KEM}}; r_2)$
if $epk_B \neq \bot$
    $\pi_A.\mathsf{type} \leftarrow \mathsf{full}$
    $(K_3, c_3) \leftarrow \mathsf{KEM}_3.\mathsf{Encaps}(epk_B; r_3)$
else
    $\pi_A.\mathsf{type} \leftarrow \mathsf{reduced}$
    $(K_3, c_3) \leftarrow (\varepsilon, \varepsilon)$
$\mathsf{ms} \leftarrow K_1 \| K_2 \| K_3$
$sid \leftarrow (A, B, pk_A, pk_B, sspk_B^{\mathsf{ssid}}, epk_B, n, c_1, c_2, c_3)$
$\sigma \leftarrow \mathsf{DVS.Sign}(sk_A^{\mathsf{DVS}}, sspk_B^{\mathsf{DVS}}, sid; r_4)$
$\pi_A.\mathsf{pid} \leftarrow B$
$\pi_A.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms}, sid)$
$\pi_A.\mathsf{cid} \leftarrow (B, pk_B, sspk_B^{\mathsf{ssid}}, epk_B)$
$\pi_A.\mathsf{sid} \leftarrow sid$
$\pi_A.\mathsf{st_{exec}} \leftarrow \mathsf{accepted}$
return $(\pi_A, m' = (A, n, c_1, c_2, c_3, \sigma))$
```

$$\xrightarrow{\quad m' \quad}$$

Bob:
```
$\mathsf{Run}(sk_B, \vec{sssk}_B, \vec{pk}, \vec{sspk}, \pi_B, m')$
$(sk_B^{\mathsf{KEM}}, sk_B^{\mathsf{DVS}}, tk_B) \leftarrow sk_B$
$(sssk_B^{\mathsf{KEM}}, sssk_B^{\mathsf{DVS}}) \leftarrow sssk_B^{\mathsf{ssid}}$
$(A, n, c_1, c_2, c_3, \sigma) \leftarrow m'$
$(pk_A^{\mathsf{KEM}}, pk_A^{\mathsf{DVS}}) \leftarrow pk_A$
$(sspk_B^{\mathsf{KEM}}, sspk_B^{\mathsf{DVS}}) \leftarrow sspk_B^{\mathsf{ssid}}$
$sid \leftarrow (A, B, pk_A, pk_B, sspk_B^{ssid}, epk_B, n, c_1, c_2, c_3)$
if $\mathsf{DVS.Vrfy}(pk_A^{\mathsf{DVS}}, sspk_B^{\mathsf{DVS}}, sid, \sigma) = \mathsf{false}$
    $\pi_B.\mathsf{st_{exec}} \leftarrow \mathsf{rejected}$
    return $(\pi_B, \varepsilon)$
$K_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_B^{\mathsf{KEM}}, c_1)$
$K_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sssk_B^{\mathsf{KEM}}, c_2)$
if $c_3 \neq \varepsilon$
    $\pi_B.\mathsf{type} \leftarrow \mathsf{full}$
    $K_3 \leftarrow \mathsf{KEM}_3.\mathsf{Decaps}(esk_B, c_3)$
else
    $\pi_B.\mathsf{type} \leftarrow \mathsf{reduced}$
    $(K_3, c_3) \leftarrow (\varepsilon, \varepsilon)$
$\mathsf{ms} \leftarrow K_1 \| K_2 \| K_3$
$\pi_B.\mathsf{pid} \leftarrow A$
$\pi_B.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms}, sid)$
$\pi_B.\mathsf{sid} \leftarrow sid$
$\pi_B.\mathsf{st_{exec}} \leftarrow \mathsf{accepted}$
return $(\pi_B, \varepsilon)$
```

Fig. 11. Formal specification of the Run algorithm of asynchronous DAKE SPQR wrt. the model given in Section 7. Note that the generation of long-term and semi-static keys during registration happens at the outset of the game $\mathcal{G}_{\mathsf{KE}}^{\mathsf{adake\text{-}kind}}(\mathcal{A})$ and the Signal Server is abstracted away. Thus these elements are not included explicitly in the above description. Note that the responder saves the ephemeral secret key in its session state.

Proof. We proceed via a sequence of game hops starting from $\mathcal{G}^{\mathsf{adake\text{-}kind}}_{\mathsf{SPQR}}(\mathcal{A})$ (cf. Figure 9), branching off into the cleanness predicates. In the final games, we will have that the adversary $\mathcal{A}$ has probability exactly $\frac{1}{2}$ in guessing the test challenge bit b. Along the way, we will further establish that the soundness predicate sound is satisfied.

Game 0. The initial game, Game $\mathcal{G}_0$, is the key indistinguishability game $\mathcal{G}^{\mathsf{adake\text{-}kind}}_{\mathsf{SPQR}}(\mathcal{A})$ for SPQR played by $\mathcal{A}$. By definition,

$$\epsilon' := \mathsf{Adv}^{\mathsf{adake\text{-}kind}}_{\mathsf{SPQR}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{SPQR}}(\mathcal{A}) = \left|\Pr[\mathcal{G}_0 = 1] - \frac{1}{2}\right|.$$

Game 1 (Nonce and randomness collisions). We modify $\mathcal{G}_0$ to overwrite the adversary's output with 0, if any two initiator sessions hold the same nonce n or the same randomness value r, or if two responder sessions pick the same ephemeral KEM key pair. As initiator nonces are uniformly random λ-bit strings, the initiator randomness is a uniformly random element from tPRF's label space $\mathcal{R}_{\mathsf{tPRF}}$, and the responder randomness is a uniformly random element from KEM_3's randomness space $\mathcal{R}_{\mathsf{KEM}_3}$, we can upper-bound the probability of this happening across the at most n_s sessions by the birthday bound:

$$\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{SPQR}}(\mathcal{A}) \leq \frac{n_s^2}{2^\lambda} + \frac{n_s^2}{2^{|\mathcal{R}_{\mathsf{tPRF}}|}} + \frac{n_s^2}{2^{|\mathcal{R}_{\mathsf{KEM}_3}|}} + \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game 2 (KEM correctness). We modify $\mathcal{G}_1$ to overwrite the adversary's output with 0 if for any key pair $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}_l()$ and encapsulation $(c, K) \leftarrow\!\!\$\ \mathsf{Encaps}_l(pk)$ used in any of the sessions, where $l \in \{1, 2, 3\}$, we have that $K \neq K' \leftarrow \mathsf{Decaps}_l(sk, c)$. The probability of this happening for any tuple (pk, sk, c) is upper-bounded by the maximal correctness error δ_{corr} among KEM_1, KEM_2, and KEM_3. As there are at most three such tuples per session, we can bound any correctness errors happening by:

$$\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{SPQR}}(\mathcal{A}) \leq 3n_s \cdot \delta_{\mathsf{corr}} + \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{SPQR}}(\mathcal{A}).$$

Soundness. At this point, soundness (i.e., $\mathsf{sound}() = \mathsf{true}$) holds unconditionally and this will not change in any of the subsequent game hops. Consider the three sub-conditions of the sound predicate:

- *Shared key, type, contributive identifier* (Figure 9, Line 18): Session identifiers fix the KEM keys and ciphertexts involved in key derivation, hence by game $\mathcal{G}_2$ KEM correctness implies agreement on K_1, K_2, and (if $\mathsf{type} = \mathsf{full}$) K_3 and thus also on K under deterministic key derivation KDF. Session identifiers have distinct entries depending, and ensuring agreement, on the session type ($epk_B = \bot$ if and only if $\mathsf{type} = \mathsf{reduced}$). Since the entries in the contributive identifier are a (proper) subset of the entries of session identifiers, agreement on session identifiers also yields agreement on contributive identifiers.
- *No initiator session identifiers collide* (Figure 9, Line 19): As of Game $\mathcal{G}_1$, each initiator session picks a unique nonce. This nonce is part of the session identifier and thus ensures uniqueness of initiator session identifiers.
- *No three session identifiers collide in full mode* (Figure 9, Line 20): We ruled out initiator collisions above already. For session identifiers to collide in two responder sessions in full mode, the two sessions would need to use the same ephemeral pre-key (epk, esk). Since we ruled out collisions in the randomness space $\mathcal{R}_{\mathsf{KEM}_3}$ sampled in responder sessions, the ephemeral pre-keys derived via $\mathsf{KGenEP}()$ are unique.

Game 3 (Guess test session π^*). Next, we guess the tested session π^* among the at most n_s sessions total at the outset of the game, and "invalidate" the game by overwriting the adversary's bit guess with 0 if the adversary calls TEST on a different session. With probability $1/n_s$, the guess is correct and this change goes unnoticed, so

$$\mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{SPQR}}(\mathcal{A}) \leq n_s \cdot \mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game 4 (Guess initiator identity U). We first guess the test session's own identity if it is an initiator session, or the test session's peer identity if it is a responder session. Note that since the test session has necessarily accepted, the peer in a responder session is also set to a valid identity in $[n_p]$, i.e., is not set to $\star$ anymore. We denote the guessed initiator identity by U and overwrite the adversary's bit guess with 0 if this guess was incorrect. This step loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_3}_{\mathsf{SPQR}}(\mathcal{A}) \leq n_p \cdot \mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game 5 (Guess responder identity V). Next, we guess the identity of the involved (intended) responder. This is π^*'s own identity if it is a responder session, or its intended peer identity if π^* is an initiator session. We denote the guessed responder identity by V and again overwrite the adversary's bit guess with 0 if this guess was incorrect. This step again loses at most a factor of the number of users n_p:

$$\mathsf{Adv}^{\mathcal{G}_4}_{\mathsf{SPQR}}(\mathcal{A}) \le n_p \cdot \mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{SPQR}}(\mathcal{A}).$$

Recall that the adversary's bit guess at the end of the game is considered only if $\mathsf{fresh}(\pi^*)$ holds for the tested session π^*. Freshness requires that the session key in neither π^* nor in a partnered session was revealed and that one of these four cleanness conditions is satisfied: $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$ **or** $\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$ **or** $\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$ **or** $\big(\pi^*.\mathsf{type} = \mathsf{full}$ **and** $\mathsf{clean}_{\mathsf{EE}}(\pi^*)\big)$.

We will now branch out into four sub-cases following the structure of the cleanness predicates, bounding the adversary's winning advantage $\mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{SPQR}}(\mathcal{A})$ by the sum of its advantages when conditioning the adversary on each of the cleanness sub-conditions being satisfied (which we write as $\mathcal{G}_5[c]$ for predicate c). Via the union bound:

$$\mathsf{Adv}^{\mathcal{G}_5}_{\mathsf{SPQR}}(\mathcal{A}) \le \sum_{\substack{\mathsf{c} \in \{\mathsf{clean}_{\mathsf{LTSS}}(\pi^*),\, \mathsf{clean}_{\mathsf{ELT}}(\pi^*),\\ \mathsf{clean}_{\mathsf{ESS}}(\pi^*),\, \pi^*.\mathsf{type}=\mathsf{full} \wedge \mathsf{clean}_{\mathsf{EE}}(\pi^*)\}}} \mathsf{Adv}^{\mathcal{G}_5[\mathsf{c}]}_{\mathsf{SPQR}}(\mathcal{A}).$$

Case A ($\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$). In this proof case, we are guaranteed that either

1. π^* is an initiator session owned by U for which both its own long-term key and its intended peer V's semi-static key are uncorrupted or
2. π^* is a responder session owned by V whose own semi-static and intended peer U's long-term keys are both uncorrupted.

We will leverage this to show that the KEM ciphertext c_2 exchanged with the test session π^* was generated for an uncorrupted KEM key with good randomness, bootstrapping key indistinguishability from the corresponding encapsulated key K_2.

Game A.0. This is the game conditioned on $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)$ being satisfied.

$$\mathsf{Adv}^{\mathcal{G}_{A.0}}_{\mathsf{SPQR}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_5[\mathsf{clean}_{\mathsf{LTSS}}(\pi^*)]}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game A.1 (Guess semi-static key of V). We now guess the identifier ssid of the responder V's (uncorrupted) semi-static key $sspk_V^{\mathsf{ssid}}$. Note that depending on the role of π^* this is either the test session's own key (if $\pi^*.\mathsf{role} = \mathsf{responder}$), or of the intended peer (if $\pi^*.\mathsf{role} = \mathsf{initiator}$). We denote the guessed identifier by ssid^*, and abort, setting the adversary's output bit to 0, if this guess is incorrect, losing at most a factor n_{ss} of the number of semi-static keys per user:

$$\mathsf{Adv}^{\mathcal{G}_{A.0}}_{\mathsf{SPQR}}(\mathcal{A}) \le n_{ss} \cdot \mathsf{Adv}^{\mathcal{G}_{A.1}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game A.2 (Signature unforgeability). We now abort the game (again, returning 0 as the adversary's bit guess) in the event that the test session π^* is a responder session and accepts having received a DVS signature σ that no session of U has issued. The probability of such an abort can be bounded by the advantage of the following reduction $\mathcal{B}_1$ against the $(t, \epsilon_{\mathsf{DVS}}, Q_S)$-unforgeability of DVS.

Reduction $\mathcal{B}_1$ samples all key components itself except for the DVS keys: In place of the long-term public DVS key pk_U^{DVS} of U and the semi-static public DVS key $sspk_V^{\mathsf{DVS}}$ of V it uses the public keys pk_S and pk_D, respectively, obtained in its unforgeability game. For all semi-static DVS (verifier) keys, it uses the public-secret key pairs obtained through the list $\mathcal{L}$ in the unforgeability game, while sampling all static DVS keys itself. (Hence, it knows the secret key component for all DVS keys but pk_U^{DVS} and $sspk_V^{\mathsf{DVS}}$). In its simulation of Game $\mathcal{G}_{A.2}$, $\mathcal{B}_1$ uses its signing oracle to compute signatures under sk_U^{DVS} (and for any peer semi-static public key $sspk$). Since $\mathsf{clean}_{\mathsf{LTSS}}(\pi^*) = \mathsf{true}$, $\mathcal{B}_1$ never has to answer the query $\textsc{CorruptLTKey}(U)$ or $\textsc{CorruptSSKey}(V, \mathsf{ssid})$. Hence $\mathcal{B}_1$ can provide a perfect simulation of $\mathcal{G}_{A.2}$, and if π^* as a responder receives a signature σ on a session-identifier message sid that no session of U has issued, $\mathcal{B}_1$ can output this as its forgery and wins. Thus,

$$\mathsf{Adv}^{\mathcal{G}_{A.1}}_{\mathsf{SPQR}}(\mathcal{A}) \le \epsilon_{\mathsf{DVS}} + \mathsf{Adv}^{\mathcal{G}_{A.2}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game A.3 (Guess partnered initiator session). By Game $\mathcal{G}_{A.2}$, we are now ensured that a responder test session does not accept unless an honest session π^*_{p} has sent the ciphertext c_2 that π^* received, as c_2 is signed under σ. We now guess this initiator session π^*_{p} (if π^* is a responder), aborting and setting $\mathcal{A}$'s output to 0, if the test session is a responder and the guess was incorrect. This reduces the adversary's advantage by a factor of at most the number of sessions n_s:

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

Game A.4 (Twisted PRF). Next, we replace all tPRF evaluations involving U's long-term secret tk_U by the evaluation of a randomly chosen function. This, in particular, replaces the value r_2 in π^* (if π^* is an initiator) or in π^*_{p} (if π^* is a responder) with an independent random value $\widetilde{r_2}$ (recall that the randomness value r is unique per session as of Game $\mathcal{G}_1$).

We bound the advantage difference introduced by this step based on the $(t, \epsilon_{\mathsf{tPRF}}, n_s)$-twisted pseudorandomness of tPRF via the following reduction $\mathcal{B}_2$. The reduction $\mathcal{B}_2$ receives a sequence of n_s tuples (x_i, y_i) (and a tuple (K', z) but this is not relevant for our purposes here), which is either $((x_1, \mathsf{tPRF}(K, x_1)), \ldots, (x_q, \mathsf{tPRF}(K, x_q)))$ or $((x_1, g(x_1)), \ldots, (x_q, g(x_q)))$ for random values $K, x_1, \ldots, x_q$ and a randomly chosen function g.

During the reduction, instead of sampling the tPRF key tk_U itself, $\mathcal{B}_2$ will simply use y_i as the expanded randomness in the i-th initiator session of U (there are at most n_s such sessions), setting $r_1 \| r_2 \| r_3 \| r_4 \leftarrow y_i$. ($\mathcal{B}_2$ simulates the rest of the game as usual, in particular generating the tPRF keys for all other users itself.) As its bit guess, $\mathcal{B}_2$ outputs 1 if $\mathcal{A}$ wins the game and 0 otherwise. Depending on which sequence $\mathcal{B}_2$ is given, it either simulates $\mathcal{G}_{A.3}$ or $\mathcal{G}_{A.4}$, thus

$$\mathsf{Adv}^{\mathcal{G}_{A.3}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{tPRF}} + \mathsf{Adv}^{\mathcal{G}_{A.4}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game A.5 (Semi-static KEM). In the following, let

$$(c_2, K_2) \leftarrow\!\!\$\; \mathsf{KEM}_2.\mathsf{Encaps}(sspk_V^{\mathsf{ssid}^*}; \widetilde{r_2})$$

be the encapsulation computed in the initiator session between π^* and π^*_{p} under the semi-static key identified by ssid^* of V. Recall that by the previous game, $\widetilde{r_2}$ is an independent random value, unknown to the adversary. This allows us to now replace the key K_2 encapsulated in c_2 with a randomly sampled key $\widetilde{K_2}$ in π^* and its partnered session(s), if existent. Furthermore, we replace K_2 with $\widetilde{K_2}$ in any session of V using ssid^* that has received the same encapsulating ciphertext c_2.[9]

We can now bound $\mathcal{A}$'s difference in advantage by the advantage of a reduction $\mathcal{B}_3$ in winning the $(t, \epsilon_{\mathsf{KEM}_2}, n_s)$-$\mathsf{IND\text{-}CCA}$ security game for KEM_2. The reduction $\mathcal{B}_3$ obtains the $\mathsf{IND\text{-}CCA}$ challenge (pk, c^*, K^*_b) and simulates the game for $\mathcal{A}$ as follows: It samples the test bit b_{test} itself and generates all long-term, semi-static, and ephemeral pre-keys itself, except for the key pair $(sspk_V^{\mathsf{ssid}^*}, sssk_V^{\mathsf{ssid}^*})$ of the previously guessed responder identity V and identifier ssid^*. The reduction embeds its received challenge public key pk by setting $sspk_V^{\mathsf{ssid}^*} = pk$. As predicate $\mathsf{clean}_{\mathsf{LTSS}}$ holds, $\mathcal{A}$ never asks the query $\textsc{CorruptSSKey}(V, \mathsf{ssid}^*)$ and the reduction thus never needs to output the secret key sk corresponding to pk.

Whenever a decapsulation of some ciphertext $c \neq c^*$ using sk is necessary to faithfully simulate the game for $\mathcal{A}$, $\mathcal{B}_3$ simply forwards this ciphertext to its decapsulation oracle $\textsc{Decaps}$ (making at most n_s queries as claimed). In both π^* and its partnered session(s) π^*_{p} (if existent), $\mathcal{B}_3$ embeds K^*_b wherever K_2 would be used and c^* wherever c_2 would be used. The same replacement is employed in responder sessions of party V that receive c_2 as an encapsulation under $sspk_V^{\mathsf{ssid}^*} = pk$. When $\mathcal{A}$ stop with output b', the reduction $\mathcal{B}_3$ returns $[\![b_{\mathsf{test}} = b']\!]$.

Observe that $\mathcal{B}_3$ perfectly simulates $\mathcal{G}_{A.4}$ if $b = 0$ in $\mathcal{G}^{\mathsf{indcca}}_{\mathsf{KEM}_2}(\mathcal{B}_3)$ and $\mathcal{G}_{A.5}$ otherwise. Hence, any difference in $\mathcal{A}$'s advantage in the two games is bounded by the distinguishing advantage of $\mathcal{B}_3$ against the $\mathsf{IND\text{-}CCA}$ security of KEM_2:

$$\mathsf{Adv}^{\mathcal{G}_{A.4}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_2} + \mathsf{Adv}^{\mathcal{G}_{A.5}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game A.6 (Session key KDF). Lastly, we replace the output of the session key derivation $K \leftarrow \mathsf{KDF}(K_1 \| \widetilde{K_2} \| K_3, sid)$ in the test session and its partnered session(s), as well as in any other session using $\widetilde{K_2}$, by the output of a random function; in particular replacing K with a uniformly random key $\widetilde{K}$. We show that any adversary that

[9]Note that we know the involved initiator session of U (it is either the test session π^* itself or its partnered session π^*_{p}) and the identity V of the owner of the involved semi-static key pair with id ssid^*. This allows us to precompute c_2 at the outset of the game and thus easily identify responder sessions that receive c_2.

can efficiently distinguish Game $\mathcal{G}_{A.6}$ from Game $\mathcal{G}_{A.5}$ can be turned into an efficient adversary $\mathcal{B}_4$ against the $(t, \epsilon_{\mathsf{KDF}}, n_s)$-pseudorandomness of the key derivation function KDF, treated as a PRF keyed through the second key component K_2 and taking (K_1, K_3, sid) as label.

The reduction $\mathcal{B}_4$ generates all key pairs itself and initializes $\mathcal{A}$ as usual. In particular, $\mathcal{B}_4$ samples the test bit b_{test} itself and can answer all CorruptSSKey, CorruptLTKey queries truthfully. Similarly, the reduction $\mathcal{B}_4$ can execute all Send queries. Furthermore, $\mathcal{B}_4$ can reveal the randomness and the session keys of sessions, with the exception of session keys in the test session and its partnered session(s) (which is unproblematic since these queries would trigger an immediate loss for the adversary when checking $\mathsf{fresh}(\pi^*)$).

In any session using $\widetilde{K_2}$ as of Game $\mathcal{G}_{A.5}$, and in particular in the test session π^* and its partner(s), $\mathcal{B}_4$ queries (K_1, K_3, sid) to its PRFChallenge oracle to compute the session key, where $sid = (U, V, pk_U, pk_V, sspk_V, epk_V, n, c_1, c_2, c_3)$; this amount to at most n_s oracle queries, as claimed. The returned values are either $\mathsf{KDF}(K_1 \| \widetilde{K_2} \| K_3, sid)$ for a uniformly random key $\widetilde{K_2}$ if $b = 0$, or the outputs of a uniformly random function g if $b = 1$. When $\mathcal{A}$ terminates with output b', $\mathcal{B}_4$ returns $[\![b_{\mathsf{test}} = b']\!]$.

Note that $\mathcal{B}_4$ perfectly simulates $\mathcal{G}_{A.5}$ if $b = 0$ and $\mathcal{G}_{A.6}$ if $b = 1$. Hence, if $\mathcal{A}$ can distinguish the two games, the reduction can win the PRF security game against KDF with the same advantage and we have

$$\mathsf{Adv}^{\mathcal{G}_{A.5}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}^{\mathcal{G}_{A.6}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Finalize To conclude this proof case, observe that in Game $\mathcal{G}_{A.6}$ the challenge K_{test} for π^* is now a uniformly random key, independent of b_{test}. Furthermore, $\mathcal{A}$ cannot reveal K_{test} via a RevealSessKey query on π^* or any partnered session which might hold the same key. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess:

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

Case B ($\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$). In this proof case, we are guaranteed that either

1. π^* is an initiator session owned by U whose randomness is unrevealed and whose intended peer V's long-term key is uncorrupted or
2. π^* is a responder session owned by V and (via $\mathsf{clean}_{\mathsf{peerE}}$) there exists a unique partnered initiator session π^*_{p} whose randomness is unrevealed and which is unique (via sound). We further know that π^*_{p} is owned by U, as the matching session identifiers include the initiator identity guessed in Game $\mathcal{G}_4$.

Game B.0. We now condition on $\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$:

$$\mathsf{Adv}^{\mathcal{G}_{B.0}}_{\mathsf{SPQR}}(\mathcal{A}) = \mathsf{Adv}^{\mathcal{G}_5[\mathsf{clean}_{\mathsf{ELT}}(\pi^*)]}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game B.1 (Guess unique partnered initiator session). As mentioned above, if the test session π^* is a responder session, by $\mathsf{clean}_{\mathsf{peerE}}$ there exists an initiator partner session π^*_{p} to π^* which furthermore is unique by sound. We now guess this partnered initiator session π^*_{p} owned by party U; if π^* is an initiator session we simply ignore the guess. The game is changed to overwrite $\mathcal{A}$'s output to 0 if the test session is a responder and the guess was incorrect. This reduces the adversary's advantage by a factor of at most the number of sessions n_s:

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

Game B.2 (Twisted PRF). Next, we replace the tPRF evaluation of the initiator session π owned by U by the evaluation of a randomly chosen function (here $\pi = \pi^*$, if $\pi^*.\mathsf{role} = \mathsf{initiator}$, and $\pi = \pi^*_{\mathsf{p}}$, if $\pi^*.\mathsf{role} = \mathsf{responder}$). In particular, we replace the value r_1 in π with an independent random value $\widetilde{r_1}$ (recall the randomness value r is unique per session as of Game $\mathcal{G}_1$).

We bound the advantage difference introduced by this step by the $(t, \epsilon_{\mathsf{tPRF}}, 0)$–twisted pseudorandomness of tPRF via the following reduction $\mathcal{B}_5$. The reduction receives (K', z) which is either $(K', \mathsf{tPRF}(K', x))$ if $b = 0$, or $(K', g'(K'))$ if $b = 1$, where K', x are random values and g' is a random function.

The reduction $\mathcal{B}_5$ then generates all keys and parameters for the key exchange games itself, but sets $tk_U \leftarrow K'$. It uses tk_U in all sessions of U except for π, where instead of evaluating $\mathsf{tPRF}(tk_U, r)$, $\mathcal{B}_5$ sets $r_1 \| r_2 \| r_3 \| r_4 \leftarrow z$. Upon a potential CorruptLTKey(U) query, $\mathcal{B}_5$ can hand out tk_U as part of U's secret key (note that r remains hidden as RevealRandom(π) is never called). As its bit guess, $\mathcal{B}_5$ outputs 1 if $\mathcal{A}$ wins the game and 0 otherwise. Depending on which sequence $\mathcal{B}_5$ is given, it either simulates $\mathcal{G}_{B.1}$ or $\mathcal{G}_{B.2}$, and thus:

$$\mathsf{Adv}^{\mathcal{G}_{B.1}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{tPRF}} \cdot \mathsf{Adv}^{\mathcal{G}_{B.2}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game B.3 (Long-term KEM). In the following, let

$$(c_1, K_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(pk_V^{\mathsf{KEM}}; \widetilde{r_1})$$

be the encapsulation computed at session π, where again $\pi = \pi^*$, if $\pi^*.\mathsf{role} = \mathsf{initiator}$, and $\pi = \pi^*_{\mathsf{p}}$, if $\pi^*.\mathsf{role} = \mathsf{responder}$. Recall that by the previous game, $\widetilde{r_1}$ is an independent random value, unknown to the adversary. In Game $\mathcal{G}_{B.3}$, we now replace the encapsulated key K_1 with a randomly sampled key $\widetilde{K_1}$ in the test session π^* and its partnered session(s), if existent. Furthermore, in any responder session of V that receives the same encapsulating ciphertext c_1, we replace K_1 with $\widetilde{K_1}$, too. Observe that, knowing the involved initiator session π as well as the long-term key identity V in advance, we can precompute c_1 at the outset of the game and then simply check when c_1 is received by responder sessions owned by V.

We bound the difference in $\mathcal{A}$'s advantage by the advantage of a reduction $\mathcal{B}_6$ against the $(t, \epsilon_{\mathsf{KEM}_1}, n_s)$-$\mathsf{IND\text{-}CCA}$ security of KEM_1 as follows. $\mathcal{B}_6$ obtains a challenge (pk, c^*, K_b^*) and simulates the game for $\mathcal{A}$ as follows: It samples the test bit b_{test} itself and generates all key pairs to initialize $\mathcal{A}$ itself, except for the long-term KEM public key of V, for which it only sets $pk_V^{\mathsf{KEM}} = pk$.

Note that $\mathsf{clean}_{\mathsf{ELT}}$ ensures that $\mathcal{A}$ never calls $\textsc{CorruptLTKey}(V)$, so $\mathcal{B}_6$ never has to output sk_V^{KEM}. Whenever $\mathcal{B}_6$ would have to use sk_V^{KEM} to decapsulate some ciphertext $c \neq c^*$ in some responder session of V, it does so via its $\textsc{Decaps}$ oracle (quering the oracle at most n_s times as claimed). In the test session π^* and its potential initiator partner π^*_{p}, $\mathcal{B}_6$ embeds K_b^* in the place of K_1 and c^* in the place of c_1. Also, in responder sessions of V receiving c_1 (recall, $\mathcal{B}_6$ knows c_1 from the start of the game), $\mathcal{B}_6$ uses K_b^* in the place of K_1 and c^* in the place of c_1. When $\mathcal{A}$ stops and outputs its bit guess b', $\mathcal{B}_6$ returns $[\![b_{\mathsf{test}} = b']\!]$.

The simulation $\mathcal{B}_6$ provides for $\mathcal{A}$ perfectly represents Game $\mathcal{G}_{B.2}$ if $b = 0$ in the $\mathsf{IND\text{-}CCA}$ game for KEM_1, and Game $\mathcal{G}_{B.3}$ otherwise. Any difference in $\mathcal{A}$'s advantage between the two games hence translates into a distinguishing advantage of $\mathcal{B}_6$ in the $\mathsf{IND\text{-}CCA}$ game against KEM_1:

$$\mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{B.2}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_1} + \mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{B.3}}(\mathcal{A}).$$

Game B.4 (Session key KDF). As the final step in this proof case, we replace in Game $\mathcal{G}_{B.4}$ the session key derived in the test and partnered session as $K \leftarrow \mathsf{KDF}(K_1 \| K_2 \| K_3, sid)$, as well as in any other session using $\widetilde{K_1}$, by the output of a random function; in particular replacing K with a randomly sampled key $\widetilde{K}$. As in the previous case in Game $\mathcal{G}_{A.6}$ we can bound the advantage introduced by this change by the advantage of an adversary $\mathcal{B}_7$ against the $(t, \epsilon_{\mathsf{KDF}}, n_s)$–pseudorandomness property of KDF, treated as a PRF keyed through the first key component K_1 and taking (K_2, K_3, sid) as label:

$$\mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{B.3}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{B.4}}(\mathcal{A}).$$

Finalize To conclude the proof, we observe that in Game $\mathcal{G}_{B.4}$, the challenge session key is uniformly random independent of b_{test} and cannot be revealed by $\mathcal{A}$, hence $\mathcal{A}$ cannot do better than guessing:

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

Case C ($\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$). In this proof case, we are guaranteed that either

1. π^* is an initiator session owned by U whose session randomness is unrevealed and whose intended peer V's semi-static key in question is uncorrupted or
2. π^* is a responder session owned by V whose semi-static key in question is uncorrupted and (via $\mathsf{clean}_{\mathsf{peerE}}$ and sound) there exists a unique partnered session π^*_{p} owned by U whose randomness is unrevealed .

Similarly to the cases before, we leverage this to show that the KEM ciphertext c_2 associated with the test session π^* was generated using an uncorrupted KEM key with good randomness, yielding key secrecy for the corresponding encapsulated key K_2.

Game C.0. We now condition on $\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$:

$$\mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{C.0}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_5[\mathsf{clean}_{\mathsf{ESS}}(\pi^*)]}(\mathcal{A}).$$

Game C.1 (Guess unique partnered initiator session). We guess the unique existing partner session π^*_{p} of π^*, if the test session is a responder session; if π^* is an initiator session, we simply ignore the guess. As before we set $\mathcal{A}$'s output to 0 if the guess was incorrect. We thus reduce the adversary's advantage by a factor of at most the number of sessions n_s:

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

Game C.2 (Guess semi-static key of V). Next, we guess the identifier ssid of the (uncorrupted) semi-static key $sspk_V^{\mathsf{ssid}}$ of party V, which, depending on the role of π^*, is either the test session's own key (if $\pi^*.\mathsf{role} = \mathsf{responder}$) or that of its intended peer (if $\pi^*.\mathsf{role} = \mathsf{initiator}$). As before, we denote the guessed identifier by ssid^*, and abort with 0 if this guess is incorrect, losing at most a factor of the number of semi-static keys per user n_{ss}:

$$\mathsf{Adv}^{\mathcal{G}_{C.1}}_{\mathsf{SPQR}}(\mathcal{A}) \leq n_{ss} \cdot \mathsf{Adv}^{\mathcal{G}_{C.2}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game C.3 (Twisted PRF). Next, we replace the tPRF evaluation of the initiator session π owned by U by the evaluation of a randomly chosen function (here $\pi = \pi^*$, if $\pi^*.\mathsf{role} = \mathsf{initiator}$, and $\pi = \pi^*_{\mathsf{p}}$, if $\pi^*.\mathsf{role} = \mathsf{responder}$).

In particular, we replace the value r_2 in π with an independent random value $\widetilde{r_2}$ (recall the randomness value r is unique per session as of Game $\mathcal{G}_1$). This change thus assures that the randomness involved in the ensuing encapsulation under the semi-static public key of V is unknown to the adversary.

We bound the advantage difference induced by this step by the twisted $(t, \epsilon_{\mathsf{tPRF}}, 0)$-twisted pseudorandomness of tPRF via the following reduction $\mathcal{B}_8$. The reduction receives (K', z) which is either $(K', \mathsf{tPRF}(K', x))$ if $b = 0$, or $(K', g'(K'))$ if $b = 1$, where K', x are random values and g' is a random function.

The reduction $\mathcal{B}_8$ then generates all keys and parameters for the key exchange games itself, in particular it sets $tk_U \leftarrow K'$. Instead of evaluating $\mathsf{tPRF}(tk_U, r)$ for $(n, r) \leftarrow \pi.\mathsf{coins}$, $\mathcal{B}_8$ sets $r_1 \| r_2 \| r_3 \| r_4 \leftarrow z$. Upon a potential $\textsc{CorruptLTKey}(U)$ query, $\mathcal{B}_8$ can hand out tk_U as part of U's secret key (while r remains hidden as $\textsc{RevealRandom}(\pi)$ is never called). As its bit guess, $\mathcal{B}_8$ outputs 1 if $\mathcal{A}$ wins the game and 0 otherwise. Depending on which sequence $\mathcal{B}_8$ is given, it either simulates $\mathcal{G}_{C.2}$ or $\mathcal{G}_{C.3}$, and thus:

$$\mathsf{Adv}^{\mathcal{G}_{C.2}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{tPRF}} + \mathsf{Adv}^{\mathcal{G}_{C.3}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game C.4 (Semi-static KEM). In the following, let

$$(c_2, K_2) \leftarrow_{\$} \mathsf{KEM}_2.\mathsf{Encaps}(sspk_V^{\mathsf{ssid}^*}; r_2)$$

be the encapsulation computed at session π, where $\pi = \pi^*$, if $\pi^*.\mathsf{role} = \mathsf{initiator}$, and $\pi = \pi^*_{\mathsf{p}}$, if $\pi^*.\mathsf{role} = \mathsf{responder}$.

We can now replace the key K_2 encapsulated in c_2 under the semi-static key of V with identifier ssid^* with a randomly sampled key $\widetilde{K_2}$ in $\pi^*$and its partnered session(s), if existent. Furthermore, we replace K_2 with $\widetilde{K_2}$ in any session of V that has received the same encapsulating ciphertext c_2.

As in previous cases, we can bound $\mathcal{A}$'s difference in advantage that was introduced by this change by the advantage of a reduction $\mathcal{B}_9$ in winning the $(t, \epsilon_{\mathsf{KEM}_2}, n_s)$-$\mathsf{IND}$-$\mathsf{CCA}$ security game for KEM_2, where the reduction $\mathcal{B}_9$ obtains its IND-CCA challenge (pk, c^*, K_b^*) and simulates the game $\mathcal{A}$ by generating all parameters of the game itself, except for embedding its received challenge public key pk by setting $sspk_U^{\mathsf{ssid}^*} = pk$. The predicate $\mathsf{clean}_{\mathsf{ESS}}$ holds, thus we know that $\mathcal{A}$ never asks a $\textsc{CorruptSSKey}(V, \mathsf{ssid}^*)$ query and the reduction need never output the secret key sk corresponding to pk. Whenever a decapsulation of some ciphertext $c \neq c^*$ using sk is necessary to faithfully simulate the game for $\mathcal{A}$, $\mathcal{B}_9$ simply forwards this ciphertext to its decapsulation oracle $\textsc{Decaps}$ (querying its oracle at most n_s times as claimed). In both π^* and its partnered session(s) π^*_{p} (if existent), $\mathcal{B}_9$ embeds K_b^* wherever K_2 would be used and c^*, wherever c_2 would be used. The same replacement is employed in any responder sessions of party V that receive c_2. At some point, $\mathcal{A}$ will stop with output b', and the reduction $\mathcal{B}_9$ returns 0 if $b' = b_{\mathsf{test}}$ and 1 otherwise.

Observe that $\mathcal{B}_9$ perfectly simulates $\mathcal{G}_{C.3}$ if $b = 0$ in $\mathcal{G}^{\mathsf{indcca}}_{\mathsf{KEM}_2}(\mathcal{B}_9)$ and $\mathcal{G}_{C.4}$ otherwise. Hence, any difference in $\mathcal{A}$'s advantage in the two games is bounded by the distinguishing advantage of $\mathcal{B}_9$ against the IND-CCA security of KEM_2:

$$\mathsf{Adv}^{\mathcal{G}_{C.3}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_2} + \mathsf{Adv}^{\mathcal{G}_{C.4}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game C.5 (Session key KDF). As the final step in this proof case, we replace in Game $\mathcal{G}_{C.5}$ the session key derived in the test and partnered session as $K \leftarrow \mathsf{KDF}(K_1 \| K_2 \| K_3, sid)$ by a randomly sampled key $\widetilde{K}$. As in the previous cases we can bound the advantage introduced by this change by the advantage of an adversary $\mathcal{B}_{10}$ against $(t, \epsilon_{\mathsf{KDF}}, n_s)$-$\mathsf{PRFSEC}$ property of KDF, this time keyed via K_2:

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

Finalize To conclude the proof, we observe that the adversary expects the challenge K_{test} to be the output of the key derivation function KDF applied to the master secret ms and session identifier sid if $b_{\mathsf{test}} = 0$ or a uniformly random string if $b_{\mathsf{test}} = 1$. In all of the above cases, this distinction cannot be made by $\mathcal{A}$ anymore as both keys are now uniformly random. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess, causing us to arrive at the final bound

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

Case D (π^*.type = full and clean$_{\mathsf{EE}}(\pi^*)$). In this proof case, we are guaranteed

1. π^* is an initiator session owned by U whose session randomness is unrevealed and that has received an ephemeral pre-key that was generated using unrevealed randomness in a session of intended partner V, or
2. π^* is a responder session owned by V whose ephemeral pre-key generation was executed with unrevealed randomness and there exists a partnered initiator session π^*_{p} owned by U whose session randomness is unrevealed.

Similarly to the cases before, we leverage this to show that the KEM ciphertext c_3 associated with the test session π^* was generated using an uncorrupted KEM key with good randomness, yielding key indistinguishability for the corresponding encapsulated key K_3.

Game D.0. We now condition on the test session running in full mode and $\mathsf{clean}_{\mathsf{EE}}(\pi^*)$ being satisfied:

$$\mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{D.0}}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_5[\pi^*.\mathsf{type}=\mathsf{full} \wedge \mathsf{clean}_{\mathsf{EE}}(\pi^*)]}(\mathcal{A}).$$

Game D.1 (Guess unique (contributive) partner session). We first guess the unique existing (contributive) partner session π^*_{p} of π^*: If π^* is a responder session, π^*_{p} is its sid-partner, if π^* is an initiator session, π^*_{p} is its contributively partnered session via cid. (Recall that this unique contributive partner exists since we ruled out collisions in the ephemeral pre-keys, and $\pi^*.\mathsf{type} = \mathsf{full}$, so π^* received such ephemeral pre-key contained in its contributive identifier.) The game sets $\mathcal{A}$'s output bit to 0 if the guess was incorrect. We thus reduce the adversary's advantage by a factor of at most the number of sessions n_s:

$$\mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{D.0}}(\mathcal{A}) \leq n_s \cdot \mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{D.1}}(\mathcal{A}).$$

Game D.2 (Twisted PRF). Next, we replace the tPRF evaluation of the initiator session π owned by U by the evaluation of a randomly chosen function (here $\pi = \pi^*$, if $\pi^*.\mathsf{role} = \mathsf{initiator}$, and $\pi = \pi^*_{\mathsf{p}}$, if $\pi^*.\mathsf{role} = \mathsf{responder}$).

In particular, we replace the value r_3 in π with an independent random value $\widetilde{r_3}$ (recall the randomness value r is unique per session as of Game $\mathcal{G}_1$). This change thus assures that the randomness involved in the ensuing encapsulation under the ephemeral pre-key of V is unknown to the adversary.

As in previous cases, we bound the advantage difference induced by this step by the $(t, \epsilon_{\mathsf{tPRF}}, 0)$-twisted pseudorandomness of tPRF via a reduction $\mathcal{B}_{11}$. The reduction receives (K', z) which is either $(K', \mathsf{tPRF}(K', x))$ if $b = 0$, or $(K', g'(K'))$ if $b = 1$, where K', x are random values and g' is a random function.

Instead of evaluating $\mathsf{tPRF}(tk_U, r)$ for $(n, r) \leftarrow \pi.\mathsf{coins}$, $\mathcal{B}_{11}$ sets $r_1 \| r_2 \| r_3 \| r_4 \leftarrow z$; as in prior cases, $\mathcal{B}_{11}$ can still answer a potential CorruptLTKey(U) query, but r remains hidden since RevealRandom(π) is never called. As its bit guess, $\mathcal{B}_{11}$ outputs 1 if $\mathcal{A}$ wins the game and 0 otherwise. Depending on which sequence $\mathcal{B}_{11}$ is given, it either simulates $\mathcal{G}_{D.1}$ or $\mathcal{G}_{D.2}$, and thus:

$$\mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{D.1}}(\mathcal{A}) \leq \epsilon_{\mathsf{tPRF}} + \mathsf{Adv}_{\mathsf{SPQR}}^{\mathcal{G}_{D.2}}(\mathcal{A}).$$

Game D.3 (Ephemeral pre-key KEM). In the following, let

$$(c_3, K_3) \leftarrow_{\$} \mathsf{KEM}_3.\mathsf{Encaps}(epk_V; r_3)$$

be the encapsulation computed at session π, where $\pi = \pi^*$, if $\pi^*.\mathsf{role} = \mathsf{initiator}$, and $\pi = \pi^*_{\mathsf{p}}$, if $\pi^*.\mathsf{role} = \mathsf{responder}$.

We can now replace the key K_3 encapsulated in c_3 under the ephemeral pre-key of V with a randomly sampled key $\widetilde{K_3}$ in $\pi^*$and its partnered session(s), if existent. Furthermore, we replace K_3 with $\widetilde{K_3}$ in any session of V that has received the same encapsulating ciphertext c_3.

Similar to previous cases, we can bound $\mathcal{A}$'s difference in advantage that was introduced by this change by the advantage of a reduction $\mathcal{B}_{12}$ in winning the $(t, \epsilon_{\mathsf{KEM}_3}, 1)$-$\mathsf{IND\text{-}CCA}$ security game for KEM_3, which embeds the received challenge (pk, c^*, K_b^*) by setting $epk_V = pk$, $c_3 = c^*$, and $K_3 = K_b^*$. The predicate $\mathsf{clean}_{\mathsf{EE}}$ holds, thus we know that $\mathcal{A}$ never asks a RevealRandom($\tilde{\pi}$) query, where $\tilde{\pi} = \pi^*$ if the π^* is the responder, and $\tilde{\pi} = \pi^*_{\mathsf{p}}$, if π^* is the initiator; hence $\mathcal{B}_{12}$ need never output the secret key sk corresponding to pk. If session $\tilde{\pi}$ receives a different ciphertext than c_3, $\mathcal{B}_{12}$ uses (once) its Decaps oracle to obtain the resulting key. At some point, $\mathcal{A}$ will stop with output b', and the reduction $\mathcal{B}_{12}$ returns 0 if $b' = b_{\mathsf{test}}$ and 1 otherwise.

Observe that $\mathcal{B}_{12}$ perfectly simulates $\mathcal{G}_{D.2}$ if $b = 0$ in $\mathcal{G}^{\mathsf{indcca}}_{\mathsf{KEM}_3}(\mathcal{B}_{12})$ and $\mathcal{G}_{D.3}$ otherwise. Hence, any difference in $\mathcal{A}$'s advantage in the two games is bounded by the distinguishing advantage of $\mathcal{B}_{12}$ against the IND-CCA security of KEM_3:

$$\mathsf{Adv}^{\mathcal{G}_{D.2}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_3} + \mathsf{Adv}^{\mathcal{G}_{D.3}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Game D.4 (Session key KDF). As the final step in this proof case, we replace in Game $\mathcal{G}_{D.4}$ the session key derived in the test and partnered session as $K \leftarrow \mathsf{KDF}(K_1 \| K_2 \| K_3, sid)$ by a randomly sampled key $\widetilde{K}$. As in the previous cases we can bound the advantage introduced by this change by the advantage of an adversary $\mathcal{B}_{13}$ against $(t, \epsilon_{\mathsf{KDF}}, 2)$-PRFSEC property of KDF (note that here, $\widetilde{K}$ is used at most in two sessions, π^* and π^*_{p}):

$$\mathsf{Adv}^{\mathcal{G}_{D.3}}_{\mathsf{SPQR}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}^{\mathcal{G}_{D.4}}_{\mathsf{SPQR}}(\mathcal{A}).$$

Finalize To conclude the proof, we observe that the adversary expects the challenge K_{test} to be the output of the key derivation function KDF applied to the master secret ms and session identifier sid if $b_{\mathsf{test}} = 0$ or a uniformly random string if $b_{\mathsf{test}} = 1$. In all of the above cases, this distinction cannot be made by $\mathcal{A}$ anymore as both keys are now uniformly random. Thus, $\mathcal{A}$ cannot gain any information about the test bit b_{test} and can do no better than to guess, causing us to arrive at the final bound

$$\mathsf{Adv}^{\mathcal{G}_{D.4}}_{\mathsf{SPQR}}(\mathcal{A}) \leq 0. \qquad \square$$

8.2 Proof of Deniability

Theorem 6 (Deniability of SPQR). *If DVS is a $(t, \epsilon_{\mathsf{srchid}}, Q_{Ch})$-source hiding designated verifier signature and tPRF is a $(t, \epsilon_{\mathsf{tPRF}}, Q_{Ch})$-pseudorandom function, then the SPQR protocol as shown in Figure 8 and formalized in Figure 11 is (t', ϵ', Q'_{Ch})-deniable, where $t' \approx t$, $\epsilon' \leq n_p Q_{Ch} \cdot \epsilon_{\mathsf{tPRF}} + n_p^2 n_{ss} \cdot \epsilon_{\mathsf{srchid}}$, where n_p is the number of parties and n_{ss} the number of semi-static keys per party, and $Q'_{Ch} = Q_{Ch}$.*

Proof. We proceed via a sequence of game hops starting from $\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{SPQR}}(\mathcal{A})$ with the secret bit $b = 0$. In the first game hop we replace the use of the tPRF with sampling the randomness uniformly at random. In the second game hop we replace the use of DVS.Sign with DVS.Sim, which is the same as $\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{SPQR}}(\mathcal{A})$ with the secret bit $b = 1$.

Game 0. The initial game, Game $\mathcal{G}_0$, is the deniability game $\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{SPQR}}(\mathcal{A})$ for SPQR played by $\mathcal{A}$ with the secret bit $b = 0$, i.e., the Chall oracle always returns real transcripts.

Game 1 (Twisted PRF). First, we replace all tPRF evaluations inside the Chall oracle with the evaluation of a randomly chosen function. This replaces the values $r_1 \| r_2 \| r_3 \| r_4 \leftarrow \mathsf{tPRF}(tk, r)$ (where tk is part of the initiator's long term secret key and r is a randomly sampled value) with independent random values $\widetilde{r_1} \| \widetilde{r_2} \| \widetilde{r_3} \| \widetilde{r_4}$.

We bound the advantage difference introduced by this step by the $(t, \epsilon_{\mathsf{tPRF}}, 0)$-twisted pseudorandomness of tPRF via the following reduction $\mathcal{B}_1$. The reduction $\mathcal{B}_1$ receives (K', z) which is either $(K', \mathsf{tPRF}(K', x))$ if $b = 0$, or $(K', g'(K'))$ if $b = 1$, where K', x are random values and g' is a random function. The reduction then simulates the game $\mathcal{G}_0$ or $\mathcal{G}_1$ (depending on the secret bit of the challenger) for $\mathcal{A}$ as follows.

For each of the n_p parties $\mathcal{B}_1$ generates a long-term key pair and n_{ss} many semi-static keys. It randomly guesses the identifier of a party $\mathsf{iid}^* \in [n_p]$ and the challenge query $i^* \in [Q_{Ch}]$ (i.e., the i^*th challenge query with iid^* as initiator) for which the deniability adversary can distinguish between the two games. Let a number $i \in [n_p Q_{Ch}]$ uniquely denote two independent values iid, q in a query (e.g., as $(\mathsf{iid} - 1) \cdot Q_{Ch} + q$) and let $i^* \in [n_p Q_{Ch}]$ denote the specific guess iid^*, q^* of the reduction. For the party iid^*, $\mathcal{B}_1$ replaces the tPRF key in the long-term secret key with K' from its own challenge. It starts $\mathcal{A}$ with all key pairs.

$\mathcal{B}_1$ answers the queries of $\mathcal{A}$ to the Chall oracle as in $\mathcal{G}_0$ except for sampling the randomness: Here $\mathcal{B}_1$ distinguishes between three cases: The first case is that the query is for $1 \leq i < i^*$ (i.e., for $\mathsf{iid} < \mathsf{iid}^*$ or the at most $q - 1$th query for $\mathsf{iid} = \mathsf{iid}^*$). Here, the reduction samples independent random values $\widetilde{r_1} \| \widetilde{r_2} \| \widetilde{r_3} \| \widetilde{r_4}$. The second case is that the query is for $i = i^*$ (i.e., the qth query for $\mathsf{iid} = \mathsf{iid}^*$). In this case, the reduction uses the randomness z from its own challenge. The third case is that the query is for $i^* < i \leq n_p Q_{Ch}$ (i.e., for $\mathsf{iid} > \mathsf{iid}^*$ or the at least $q+1$th query for $\mathsf{iid} = \mathsf{iid}^*$). In that case, the reduction derives randomness using the tPRF key from its own challenge: $r_1 \| r_2 \| r_3 \| r_4 \leftarrow \mathsf{tPRF}(K', r)$ for a randomly sampled r. Finally, when $\mathcal{A}$ returns its guess bit b', $\mathcal{B}_1$ returns b' as its guess.

Observe that $\mathcal{B}_1$ faithfully simulates either the game $\mathcal{G}_0$ or $\mathcal{G}_1$ for $\mathcal{A}$. Moreover, the runtime of $\mathcal{B}_1$ is essentially the runtime of $\mathcal{A}$ plus the runtime to generate the keys and answer the oracle queries.

Now let us analyze the probability of $\mathcal{A}$ distinguishing between the two games $\mathcal{G}_0$ and $\mathcal{G}_1$. For this, we define the hybrids $H_0, \ldots, H_{n_p Q_{Ch}}$ with H_i being the hybrid that answers all challenge queries for indices $1, \ldots, i$ using randomness sampled uniformly at random and all other challenge queries for indices $i+1, \ldots, n_p Q_{Ch}$ are answered using randomness from $\mathsf{tPRF}(K', \cdot)$. The extreme hybrids are $H_{n_p Q_{Ch}}$ which answers all the challenge queries using randomness from $\mathsf{tPRF}(K', \cdot)$ and H_0 which answers all queries using uniformly sampled randomness. Hence, H_0 is equivalent to $\mathcal{G}_1$ and $H_{n_p Q_{Ch}}$ to $\mathcal{G}_0$. Observe that H_{i-1} and H_i only differ on how the randomness is sampled for one query depending on the reduction $\mathcal{B}_1$'s challenge oracle. Hence, it is easy to see that the probability of distinguishing between H_{i-1} and H_i is bounded by $\epsilon_{\mathsf{tprfsec}}$.

Summing the individual hybrid difference over all $n_p Q_{Ch}$ hybrid steps, we get

$$\left|\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{SPQR}}(\mathcal{A})\right| = \left|\mathsf{Adv}^{H_0}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{H_{n_p Q_{Ch}}}_{\mathsf{SPQR}}(\mathcal{A})\right|$$
$$\leq \sum_{i=1}^{n_p Q_{Ch}} \left|\mathsf{Adv}^{H_{i-1}}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{H_i}_{\mathsf{SPQR}}(\mathcal{A})\right| \leq \sum_{i=1}^{n_p Q_{Ch}} \epsilon_{\mathsf{tprfsec}} \leq n_p Q_{Ch} \cdot \epsilon_{\mathsf{tprfsec}}.$$

Game 2 (DVS simulation). In this game we replace the call to $\mathsf{DVS.Sign}$ with a call to $\mathsf{DVS.Sim}$.

We bound the advantage difference introduced by this step by the $(t, \epsilon_{\mathsf{srchid}}, Q_{Ch})$-source hiding of DVS via the following reduction $\mathcal{B}_2$.

The challenger starts $\mathcal{B}_2$ with two DVS key pairs who then simulates the game $\mathcal{G}_1$ or $\mathcal{G}_2$ (depending on the secret bit of the challenger) for $\mathcal{A}$ as follows. For each of the n_p parties $\mathcal{B}_2$ generates a long-term key pair and n_{ss} many semi-static keys. It randomly guesses the identifiers of two parties $\mathsf{iid}^*, \mathsf{rid}^* \in [n_p]$ and the identifier of a semi-static key $\mathsf{ssid}^* \in [n_{ss}]$ for which the deniability adversary can distinguish between the two games. Let a number $i \in [n_p^2 n_{ss}]$ uniquely denote three independent values $\mathsf{iid}, \mathsf{rid}, \mathsf{ssid}$ in a query (e.g., as $(\mathsf{iid}-1) \cdot n_p \cdot n_{ss} + (\mathsf{rid}-1) \cdot n_{ss} + \mathsf{ssid}$) and let $i^* \in [n_p^2 n_{ss}]$ denote the specific guess $\mathsf{iid}^*, \mathsf{rid}^*, \mathsf{ssid}^*$ of the reduction. For the party iid^*, $\mathcal{B}_2$ replaces the DVS sender key pair in the long-term key with its own challenge key pair (pk_S, sk_S). For the party rid^*, $\mathcal{B}_2$ replaces the DVS verifier key pair in the semi-static key with id ssid^* with its own challenge key pair (pk_D, sk_D). It starts $\mathcal{A}$ with all key pairs.

$\mathcal{B}_2$ answers the queries of $\mathcal{A}$ to the CHALL oracle as follows: First, it runs the responder ephemeral key generation. Then, it randomly samples the nonce n and the randomness used to compute the three KEM ciphertexts and the DVS signature. Using this randomness, $\mathcal{B}_2$ computes the KEM ciphertexts (c_1, c_2, c_3). It sets the master secret ms to the concatenation of the KEM encapsulations. In the next step, the reduction computes the DVS signature on the session identifier sid. Here $\mathcal{B}_2$ distinguishes between three cases: The first case is that the query is for $1 \leq i < i^*$. Then the reduction executes $\mathsf{DVS.Sign}$ (using the randomness sampled previously). The second case is that the query is for $i = i^*$. In this case the reduction forwards the query to its own oracle to obtain a DVS signature or a simulated one depending on the outside challenge bit. The third case is that the query is for $i^* < i \leq n_p^2 n_{ss}$. Then the reduction executes $\mathsf{DVS.Sim}$ (using the randomness sampled previously). In all cases the reduction then proceeds to compute the session key K from the master secret and the session id. Finally, the reduction returns the transcript and the session key K to $\mathcal{A}$. Hence, the transcript and session key were computed either as specified by Run (using uniformly sampled randomness instead) or as specified by Fake, depending on the query index i and the secret bit of the DVS challenger. Finally, when $\mathcal{A}$ returns its guess bit b', $\mathcal{B}_2$ returns b' as its guess.

Observe that $\mathcal{B}_2$ faithfully simulates either the game $\mathcal{G}_1$ or $\mathcal{G}_2$ for $\mathcal{A}$. Moreover, the runtime of $\mathcal{B}_2$ is essentially the runtime of $\mathcal{A}$ plus the runtime to generate the keys and answer the oracle queries.

Now let us analyze the probability of $\mathcal{A}$ distinguishing between the two games $\mathcal{G}_1$ and $\mathcal{G}_2$. For this, we define the hybrids $H_0, \ldots, H_{n_p^2 n_{ss}}$ with H_i being the hybrid that answers all challenge queries for indices $1, \ldots, i$ by Run (with uniformly sampled randomness) and all other challenge queries for indices $i+1, \ldots, n_p^2 n_{ss}$ are answered with Fake. The extreme hybrids are $H_{n_p^2 n_{ss}}$ which answers all the challenge queries with Run using uniformly sampled randomness and H_0 which answers all queries by Fake. Hence, H_0 is equivalent to $\mathcal{G}_1$ and $H_{n_p^2 n_{ss}}$ to $\mathcal{G}_2$. Observe that H_{i-1} and H_i only differ in an execution of $\mathsf{DVS.Sign}$ or $\mathsf{DVS.Sim}$ depending on the reduction $\mathcal{B}_2$'s challenge oracle. Hence, it is easy to see that the probability of distinguishing between H_{i-1} and H_i is bounded by $\epsilon_{\mathsf{srchid}}$.

Summing the individual hybrid difference over all $n_p Q_{Ch}$ hybrid steps, we get

$$\left|\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{SPQR}}(\mathcal{A})\right| = \left|\mathsf{Adv}^{H_0}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{H_{n_p^2 n_{ss}}}_{\mathsf{SPQR}}(\mathcal{A})\right|$$
$$\leq \sum_{i=1}^{n_p^2 n_{ss}} \left|\mathsf{Adv}^{H_{i-1}}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{H_i}_{\mathsf{SPQR}}(\mathcal{A})\right| \leq \sum_{i=1}^{n_p^2 n_{ss}} \epsilon_{\mathsf{srchid}} \leq n_p^2 n_{ss} \cdot \epsilon_{\mathsf{srchid}}.$$

Finalize To conclude the proof, we observe the initial game $\mathcal{G}_0$ is the game $\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{SPQR}}(\mathcal{A})$ with the secret bit $b = 0$, and the final game $\mathcal{G}_2$ is the game $\mathcal{G}^{\mathsf{adake\text{-}den}}_{\mathsf{SPQR}}(\mathcal{A})$ with the secret bit $b = 1$. Collecting the bounds, we get

$$\epsilon' = \mathsf{Adv}^{\mathsf{adake\text{-}den}}_{\mathsf{SPQR}}(\mathcal{A}) = \left|\mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{SPQR}}(\mathcal{A})\right| + \left|\mathsf{Adv}^{\mathcal{G}_1}_{\mathsf{SPQR}}(\mathcal{A}) - \mathsf{Adv}^{\mathcal{G}_2}_{\mathsf{SPQR}}(\mathcal{A})\right|$$
$$\leq n_p Q_{Ch} \cdot \epsilon_{\mathsf{tPRF}} + n_p^2 n_{ss} \cdot \epsilon_{\mathsf{srchid}}. \qquad \square$$

9 Discussion and Limitations

Our protocols demonstrate that designated verifier signatures are helpful for constructing practical AKE protocols with constraints on the message flow (asynchronicity) and with specialized security properties (deniability).

The key ingredient in our approach for achieving post-quantum asynchronous DAKE is a post-quantum designated verifier signature scheme. While there are several lattice-based DVS schemes in the literature as described in Section 3.1, we believe that their security merits further scrutiny before adoption. In the meantime, we propose instantiations via 2-user ring signatures, for which we discussed post-quantum candidates in Section 3.2.

We believe SPQR is a good start as a PQ replacement for the Signal X3DH handshake, but in any real-world protocol deployment there are many subtleties, some of which we now highlight.

The way Signal is used in practice has the semi-static keys signed under the long-term key. In SPQR the long-term key is not suitable for this purpose, so an additional long-term signing key might have to be introduced solely for the purposes of signing the other keys; note this could be done without undermining deniability. This characteristic was likewise not considered in the provable security analysis of Signal of [21].

SPQR is solely a replacement for the initial handshake (X3DH). A fully post-quantum Signal would require quantum-resistance in the ratcheting and message encryption; fortunately there are several generic treatments of ratcheting [6,76,1].

As Signal does not use certificates or a PKI, long-term public keys must be manually authenticated out-of-band, and that remains the case with SPQR.

Our analysis of SPQR considers randomness exposure, but not malicious randomness. The latter has been captured for ratcheting [1], but not yet in the initial handshake. Our security analysis shows that SPQR, as an authenticated key exchange protocol, has offline deniability. As discussed in Appendix A, we think that our deniability notion is the best one can hope for if the adversary has access to the secret keys. We leave formally proving this as future work.

Cryptographic deniability should be treated with caution. How cryptographers understand deniability may be different from how a judge in a legal system understands it [85]. Additionally, there are stronger notions of deniability [32] that SPQR (and the Signal handshake) does not achieve, such as if one party maliciously generates messages or colludes in real-time with the judge. One should further confirm deniability at all protocol levels, and that deniability of individual components composes appropriately. Despite all these subtleties, steps toward deniability are helpful, as Unger and Goldberg write [85]: "we should strive to design deniable protocols to avoid unintentionally incriminating users."

Acknowledgements

We thank Shuichi Katsumata and the anonymous reviewers of PKC 2022 for the helpful comments. Furthermore, we also thank anonymous reviewers who pointed out a flaw in our DVS constructions in an earlier version of this paper.

R.F. was supported by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. F.G. was supported in part by German Research Foundation (DFG) Research Fellowship grant GU 1859/1-1. C.J. was (partially) funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297. D.S. was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant RGPIN-2016-05146.

References

1. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 129–158. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_5
2. Azarderakhsh, R., Jao, D., Leonardi, C.: Post-quantum static-static key agreement using multiple protocol instances. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 45–63. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-72565-9_3
3. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_21
4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A.D. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (May 1995). https://doi.org/10.1007/BFb0053428
5. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_34
6. Bellare, M., Singh, A.C., Jaeger, J., Nyayapati, M., Stepanovs, I.: Ratcheted encryption and key exchange: The security of messaging. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 619–650. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63697-9_21
7. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. Journal of Cryptology **22**(1), 114–138 (Jan 2009). https://doi.org/10.1007/s00145-007-9011-9
8. Beullens, W., Katsumata, S., Pintore, F.: Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 464–492. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_16
9. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) 6th IMA International Conference on Cryptography and Coding. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (Dec 1997)
10. Boneh, D., Glass, D., Krashen, D., Lauter, K., Sharif, S., Silverberg, A., Tibouchi, M., Zhandry, M.: Multiparty non-interactive key exchange and more from isogenies on elliptic curves. Journal of Mathematical Cryptology **14**(1), 5–14 (2020)
11. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 493–522. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_17
12. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018. pp. 353–367. IEEE (2018), https://cryptojedi.org/papers/#kyber
13. Boyd, C., Cliff, Y., Gonzalez Nieto, J.M., Paterson, K.G.: One-round key exchange in the standard model. IJACT **1**, 181–199 (2009)
14. Boyd, C., Gellert, K.: A Modern View on Forward Security. The Computer Journal **64**(4), 639–652 (08 2020). https://doi.org/10.1093/comjnl/bxaa104, https://doi.org/10.1093/comjnl/bxaa104
15. Brendel, J., Fischlin, M., Günther, F., Janson, C., Stebila, D.: Towards post-quantum security for Signal's X3DH handshake. In: 27th Conference on Selected Areas in Cryptography (SAC). Springer (Oct 2020)
16. Cai, J., Jiang, H., Zhang, P., Zheng, Z., Wang, H., Lü, G., Xu, Q.: Id-based strong designated verifier signature over $\mathcal{R}$-SIS assumption. Secur. Commun. Networks **2019**, 9678095:1–9678095:8 (2019). https://doi.org/10.1155/2019/9678095, https://doi.org/10.1155/2019/9678095
17. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_28
18. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_10, https://eprint.iacr.org/2002/120/
19. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. Journal of Cryptology **25**(4), 601–639 (Oct 2012). https://doi.org/10.1007/s00145-011-9105-2
20. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
21. Cohn-Gordon, K., Cremers, C.J.F., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the Signal messaging protocol. In: IEEE European Symposium on Security and Privacy, EuroS&P 2017. pp. 451–466 (2017). https://doi.org/10.1109/EuroSP.2017.27
22. Cohn-Gordon, K., Cremers, C.J.F., Garratt, L.: On post-compromise security. In: Hicks, M., Köpf, B. (eds.) CSF 2016 Computer Security Foundations Symposium. pp. 164–178. IEEE Computer Society Press (2016). https://doi.org/10.1109/CSF.2016.19
23. Cremers, C., Feltz, M.: One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300 (2011), https://eprint.iacr.org/2011/300

24. Dagdelen, Ö., Fischlin, M., Gagliardoni, T., Marson, G.A., Mittelbach, A., Onete, C.: A cryptographic analysis of OPACITY - (extended abstract). In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 345–362. Springer, Heidelberg (Sep 2013). https://doi.org/10.1007/978-3-642-40203-6_20
25. Damgård, I., Haagh, H., Mercer, R., Nitulescu, A., Orlandi, C., Yakoubov, S.: Stronger security and constructions of multi-designated verifier signatures. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 229–260. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_9
26. de Saint Guilhem, C.D., Smart, N.P., Warinschi, B.: Generic forward-secure key agreement without signatures. In: Nguyen, P.Q., Zhou, J. (eds.) ISC 2017. LNCS, vol. 10599, pp. 114–133. Springer, Heidelberg (Nov 2017)
27. Derler, D., Ramacher, S., Slamanig, D.: Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018. pp. 419–440. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-79063-3_20
28. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 400–409. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180454
29. Dobson, S., Galbraith, S.D.: Post-quantum signal key agreement with SIDH. Cryptology ePrint Archive, Report 2021/1187 (2021), https://eprint.iacr.org/2021/1187
30. Dobson, S., Galbraith, S.D., LeGrow, J.T., Ti, Y.B., Zobernig, L.: An adaptive attack on 2-sidh. Int. J. Comput. Math. Comput. Syst. Theory **5**(4), 282–299 (2020). https://doi.org/10.1080/23799927.2020.1822446, https://doi.org/10.1080/23799927.2020.1822446
31. Dobson, S., Li, T., Zobernig, L.: A note on a static SIDH protocol. Cryptology ePrint Archive, Report 2019/1244 (2019), https://eprint.iacr.org/2019/1244
32. Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and on-line deniability of authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_10
33. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 1197–1210. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813653
34. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 335–352. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_19
35. Duits, I.: The Post-Quantum Signal Protocol: Secure Chat in a Quantum World. Master's thesis, University of Twente (2019), http://essay.utwente.nl/77239/
36. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: 30th ACM STOC. pp. 409–418. ACM Press (May 1998). https://doi.org/10.1145/276698.276853
37. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 115–146. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_5
38. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19. LNCS, vol. 11464, pp. 67–88. Springer, Heidelberg (Jun 2019). https://doi.org/10.1007/978-3-030-21568-2_4
39. Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 567–584. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3354200
40. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
41. Fischlin, M., Mazaheri, S.: Notions of deniable message authentication. In: Ray, I., Hopper, N., Jansen, R. (eds.) Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, WPES 2015, Denver, Colorado, USA, October 12, 2015. pp. 55–64. ACM (2015). https://doi.org/10.1145/2808138.2808143, https://doi.org/10.1145/2808138.2808143
42. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_17
43. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_28
44. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
45. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 63–91. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_3

46. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407
47. Harn, L., Lee, C., Lin, C., Chang, C.: Fully deniable message authentication protocols preserving confidentiality. Comput. J. **54**(10), 1688–1699 (2011). https://doi.org/10.1093/comjnl/bxr081, https://doi.org/10.1093/comjnl/bxr081
48. Hashimoto, K., Katsumata, S., Kwiatkowski, K., Prest, T.: An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 410–440. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75248-4_15
49. Hashimoto, K., Katsumata, S., Kwiatkowski, K., Prest, T.: An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. Cryptology ePrint Archive, Report 2021/616 (2021), https://eprint.iacr.org/2021/616
50. Herranz, J.: Some digital signature schemes with collective signers. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona (2005), https://upcommons.upc.edu/bitstream/handle/2117/94334/01Jhs01de01.pdf
51. Hülsing, A., Weber, F.: Epochal signatures for deniable group chats. In: 2021 IEEE Symposium on Security and Privacy. pp. 1677–1695. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00058
52. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_13
53. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011. pp. 19–34. Springer, Heidelberg (Nov / Dec 2011). https://doi.org/10.1007/978-3-642-25405-5_2
54. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 525–537. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243805
55. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. In: Hong, D. (ed.) ICISC 20. LNCS, vol. 12593, pp. 58–84. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-68890-5_4
56. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: 27th Conference on Selected Areas in Cryptography (SAC). Springer (Oct 2020)
57. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_33
58. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000. The Internet Society (Feb 2000)
59. Kurosawa, K., Furukawa, J.: 2-pass key exchange protocols from CPA-secure KEM. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 385–401. Springer, Heidelberg (Feb 2014). https://doi.org/10.1007/978-3-319-04852-9_20
60. Laguillaumie, F., Vergnaud, D.: Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In: Blundo, C., Cimato, S. (eds.) SCN 04. LNCS, vol. 3352, pp. 105–119. Springer, Heidelberg (Sep 2005). https://doi.org/10.1007/978-3-540-30598-9_8
61. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (Nov 2007)
62. Li, B., Liu, Y., Yang, S.: Lattice-based universal designated verifier signatures. In: 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE). pp. 329–334. IEEE (2018). https://doi.org/10.1109/ICEBE.2018.00062
63. Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1343–1360. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134006
64. Li, Y., Susilo, W., Mu, Y., Pei, D.: Designated verifier signature: Definition, framework and new constructions. In: Indulska, J., Ma, J., Yang, L.T., Ungerer, T., Cao, J. (eds.) Ubiquitous Intelligence and Computing, 4th International Conference, UIC 2007, Hong Kong, China, July 11-13, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4611, pp. 1191–1200. Springer (2007). https://doi.org/10.1007/978-3-540-73549-6_116, https://doi.org/10.1007/978-3-540-73549-6_116
65. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 1–31. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_1
66. Lu, X., Au, M.H., Zhang, Z.: Raptor: A practical lattice-based (linkable) ring signature. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 19. LNCS, vol. 11464, pp. 110–130. Springer, Heidelberg (Jun 2019). https://doi.org/10.1007/978-3-030-21568-2_6
67. Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_35
68. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
69. Lyubashevsky, V., Neven, G.: One-shot verifiable encryption from lattices. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 293–323. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_11

70. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: SMILE: Set membership from ideal lattices with applications to ring signatures and confidential transactions. Cryptology ePrint Archive, Report 2021/564 (2021), https://eprint.iacr.org/2021/564
71. Marlinspike, M., Perrin, T.: The double ratchet algorithm (November 2016), https://www.signal.org/docs/specifications/doubleratchet/
72. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol (November 2016), https://signal.org/docs/specifications/x3dh/
73. Naor, M.: Deniable ring authentication. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_31
74. Noh, G., Jeong, I.R.: Strong designated verifier signature scheme from lattices in the standard model. Security Comm. Networks **9**, 6202–6214 (Feb 2017). https://doi.org/10.1002/sec.1766
75. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 463–492. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_16
76. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 3–32. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_1
77. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
78. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
79. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_32
80. Saeednia, S., Kremer, S., Markowitch, O.: An efficient strong designated verifier signature scheme. In: Lim, J.I., Lee, D.H. (eds.) ICISC 03. LNCS, vol. 2971, pp. 40–54. Springer, Heidelberg (Nov 2004)
81. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1461–1480. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3423350
82. Sheikhi-Garjan, M., Kiliç, N.G.O., Cenk, M.: A supersingular isogeny-based ring signature. Cryptology ePrint Archive, Report 2021/1318 (2021), https://eprint.iacr.org/2021/1318
83. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (Nov / Dec 2003). https://doi.org/10.1007/978-3-540-40061-5_33
84. Sun, X., Tian, H., Wang, Y.: Toward quantum-resistant strong designated verifier signature from isogenies. In: 4th International Conference on Intelligent Networking and Collaborative Systems. pp. 292–296. IEEE (2012). https://doi.org/10.1109/iNCoS.2012.70
85. Unger, N., Goldberg, I.: Deniable key exchanges for secure messaging. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 1211–1223. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813616
86. Unger, N., Goldberg, I.: Improved strongly deniable authenticated key exchanges for secure messaging. PoPETs **2018**(1), 21–66 (Jan 2018). https://doi.org/10.1515/popets-2018-0003
87. Vatandas, N., Gennaro, R., Ithurburn, B., Krawczyk, H.: On the cryptographic deniability of the Signal protocol. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 20, Part II. LNCS, vol. 12147, pp. 188–209. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-57878-7_10
88. Wang, F., Hu, Y., Wang, B.: Lattice-based strong designate verifier signature and its applications. Malaysian Journal of Computer Science **25**, 11–22 (2012)
89. Wang, F., Hu, Y., Wang, B.: Identity-based strong designate verifier signature over lattices. The Journal of China Universities of Post and Telecommunications **21**, 52–60 (2014). https://doi.org/10.1016/S1005-8885(14)60345-9
90. Yang, B., Yu, Y., Sun, Y.: A novel construction of SDVS with secure disavowability. Clust. Comput. **16**(4), 807–815 (2013). https://doi.org/10.1007/s10586-013-0254-y, https://doi.org/10.1007/s10586-013-0254-y
91. Yuen, T.H., Esgin, M.F., Liu, J.K., Au, M.H., Ding, Z.: DualRing: Generic construction of ring signatures with efficient instantiations. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 251–281. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_10
92. Zaverucha, G., Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Katz, J., Wang, X., Kolesnikov, V., Kales, D.: Picnic. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
93. Zhang, Y., Liu, Q., Tang, C., Tian, H.: A lattice-based designated verifier signature for cloud computing. International Journal of High Performance Computing and Networking **8**, 135–143 (Jun 2015). https://doi.org/10.1504/IJHPCN.2015.070013

Supplementary Material

A Related Work on Deniability

Deniability allows a party to deny having interacted with a peer. In particular, the peer cannot convince a judge of the first party having interacted with itself. *Online deniability* is concerned with the scenario of the judge interacting with the peer during the protocol execution with the first party. This notion is not achievable in the asynchronous setting. [85] Hence, we address *offline deniability*, where the peer presents data to the judge *after* the protocol execution has taken place.

Prior work defined several notions of offline deniability for authenticated key exchange [28,23,24,85,86]. Based on the work of Dwork, Naor, and Sahai [36] on deniable authentication, Di Raimondo, Gennaro, and Krawczyk defined *concurrently deniable (or fully deniable) authenticated key exchange* using the simulation paradigm in [28]. Given the list of all public keys and some auxiliary information (e.g., some honestly generated transcripts), the adversary may freely interact with honest parties as either initiator or as responder, interleaving between executions at will. The view of the adversary then consists of the transcripts, session keys, and random coins of the protocol executions it took part in. The session key is included in the view as it may be used as part of another protocol for which deniability is desirable. This view needs to be indistinguishable from the output of a simulator running on the same inputs as the adversary.

Di Raimondo, Gennaro, and Krawczyk [28] also proposed a weaker notion called *partial deniability*, which formalizes the intuition that it is indistinguishable whether an (honest) user interacted with party A or party B. Based on the definition of partial deniability, Cremers and Feltz [23] proposed *peer deniability* and *peer-and-time deniability*. For either notion the simulator does not have to output the session key and gets access to the secret key of corrupted parties. Peer-deniability intuitively allows a user to deny its communication peer, while peer-and-time deniability allows a party to deny that it was alive during a certain time frame and interacted with a certain peer.

Dagdelen, Fischlin, Gagliardoni, Marson, Mittelbach, and Onete [24] proposed a game-based definition called *outsider deniability*. Here, the adversary has access to Init, Exec, and Send oracles (identical to the key secrecy game) and a modified challenge oracle. Depending on the secret bit, the challenge oracle returns either a real transcript and session key or a transcript and session key simulated based on public data. Intuitively, this allows parties to deny having engaged in a protocol run against an eavesdropper that frames a party.

In [85,86], Unger and Goldberg have given deniability definitions in the UC model. For this they define an ideal functionality called post-specified peer key exchange with incrementing abort that unifies the model of contributiveness, deniability with abort, and their model [85] of post-specified peers.

In [87], Vatandas, Gennaro, Ithurburn, and Krawczyk provided an analysis that Signal's X3DH is deniable (wrt. full deniability of [28]) under a general extractability assumption.

Recently, Hülsing and Weber have defined deniability for group chats (and not just key exchange) in [51]. They formalize a stronger notion than ours that allows an unbounded judge to choose all long-term key pairs and learn all short-term keys, and the simulator does not get access to any secret key. Furthermore, the judge chooses the instructions (i.e., messages and group actions) to be executed. However, the group setting requires a restriction: Informally, they need one message that authentically reaches all other group members.

We propose a game-based definition, Definition 11, where the adversary interacts with a real-or-random challenge oracle. Intuitively, the "real" part relates to the view of the adversary and the "random" part to the simulated view. However this simulated view cannot make use of features like re-winding and is a plain probabilistic classical algorithm. At the same time, the distinguisher (or judge) of simulation-based definitions relates to the adversary in our game-based definition. Note that the "real" part of the challenge oracle is generated according to the protocol specification. Hence, our notion captures semi-honest adversaries, not malicious adversaries.

We further take into account the informal requirement on deniability for asynchronous DAKE in [72, §4.4]: *In some cases a third party that has compromised legitimate private keys from Alice or Bob could be provided a communication transcript that appears to be between Alice and Bob and that can only have been created by some other party that also has access to legitimate private keys from Alice or Bob.* This informal description implies both a relaxation and a strengthening compared to previous definitions: Firstly, one has to make use of the secret key of either party to simulate a transcript. Secondly, the judge may have access to all secret keys.

Observe that we only consider asynchronous DAKEs that consist of only one message with a specified peer (for our protocols it is the the initiator's message). Note that the responder's message is independent of the peer and, therefore, cannot serve as incriminating evidence. It remains to be shown that the initiator's message could have been produced by either using the initiator's secret key and the Run algorithm, or by using the responder's secret key and the Fake algorithm. Since only one message is tied to a specific peer, we do not need to take any special precautions to achieve deniability for concurrent executions.

We give the adversary (i.e., the distinguisher) access to all secret keys. This models the scenario where, e.g., a party is framed in court and the judge (in a legal sense) learns the secret keys of all involved parties through a subpoena. Hence, our distinguisher is significantly stronger than previous distinguishers. As the distinguisher has access to all secret keys, the challenge oracle does not return the random coins used. Otherwise, the distinguisher could compute both the real and simulated execution of the protocol and check which result is identical to the return value of the oracle. One could prevent this by requiring identical outputs instead of indistinguishably distributed outputs. We deem this impractical, though.

Harn, Lee, Lin, and Chang [47] have introduced the terms *1-out-of-*2 *deniable*, *1-out-of-n deniable*, and *1-out-of-*∞ *deniable* to differentiate flavors of deniability (in the context of message authentication). A DVS signature is 1-out-of-2 deniable since either party could have generated the signature (due to the source hiding property defined in Definition 6). A ring signature is 1-out-of-n deniable for a ring of size n since any ring member could have generated the signature (due to the anonymity property defined in Definition 9). An unauthenticated DH key exchange is 1-out-of-∞ deniable since the DH key shares are not bound to anybody. Our deniability definition is 1-out-of-2 deniable since the Fake algorithm has access to the receiver's secret key. Similarly, Hülsing and Weber [51] distinguish between *universal deniability* (i.e., 1-out-of-∞ deniability) and *non-universal deniability* (i.e., 1-out-of-2 and 1-out-of-n deniability). The deniability definitions of prior works [28,23,24,51] provide 1-out-of-∞ deniability since the simulator does not get access to any secret key.

In the context of deniable ring authentication, Naor [73] uses the terms source hiding and deniability to differentiate between interactive and non-interactive adversaries (judges). Here the term source hiding corresponds to online deniability while the term deniability corresponds to offline deniability in the key exchange literature. Naor also considers both notions in the big brother setting, where the attacker (i.e., the judge in the simulation-based setting) has access to all secret keys.

Using these terms, we provide the first definition of offline 1-out-of-2 deniability in the big brother setting for key exchange. Previous definitions (in particular [28]) provide offline 1-out-of-∞ deniability outside of the big brother setting.

Intuitively, it seems impossible to achieve 1-out-of-∞ deniability in the big brother setting: Assume a judge in the big brother setting cannot distinguish between a real transcript and one that was simulated using only public data. Then, the initiator-to-responder authentication is based on a proof that can be simulated using public data only (e.g., a DVS signature with a Sim algorithm that does not need access to a secret key). Note that the responder may leverage its own secret keys to verify the initiator-to-responder authenticator. However, the judge also has access to the responder's secret keys and can therefore use the same strategy to verify the authenticator. It seems implausible that an authentication scheme is correct (i.e., convinces an honest responder) and at the same time an authenticator that is simulated without knowledge to secrets is indistinguishable from a valid authenticator if one has access to the secret keys. In consequence, it seems that 1-out-of-2 deniability is the best we can hope for in the big brother setting.

B Designated Verifier Signatures from Chameleon Hash Functions

As mentioned in Section 3.1, in an earlier version we attempted to build post-quantum designated verifier signatures in a direct manner, following full-domain-hash and Fiat–Shamir-type approaches and involving chameleon hash functions. For transparency and educational purposes, we comment on these attempts a bit further in the following.

We first recall the definition of a chameleon hash function, more concretely, the formalization of Cash, Hofheinz, Kiltz, and Peikert [19], where the trapdoor enables preimage sampling (unlike [58], where the trapdoor enables collision sampling).

Definition 12. *A* chameleon hash function *(CHF) is a tuple of algorithms* $\mathsf{CHF} = (\mathsf{KGen}, \mathsf{Hash}, \mathsf{Inv})$ *with public key space* $\mathcal{P}$*, message space* $\mathcal{M}$*, digest space* $\mathcal{D}$*, randomness space* $\mathcal{R}$*, and a (not necessarily uniform) distribution* $\mathcal{R}_{dist}$ *over* $\mathcal{R}$*:*

- $\mathsf{KGen}() \xrightarrow{\$} (pk, sk)$*: A probabilistic key generation algorithm.*
- $\mathsf{Hash}(pk, m; r) \to h$*: A hashing algorithm that takes as input a public key pk and a message* $m \in \mathcal{M}$ *along with randomness* $r \in \mathcal{R}$*, and outputs a digest* $h \in \mathcal{D}$*.*
- $\mathsf{Inv}(sk, h, m) \xrightarrow{\$} r$*: A probabilistic hash inversion algorithm that takes as input a secret key sk, digest* $h \in \mathcal{D}$*, and message* $m \in \mathcal{M}$*, and outputs randomness* $r \in \mathcal{R}$*.*

Chameleon hash functions provide the same security properties as standard hash functions, with the addition of the *chameleon properties* introduced by the trapdoor:

Definition 13. *A CHF is (t, ϵ)-secure if it satisfies:*

Uniformity *For $(pk, sk) \leftarrow_\$ \mathsf{KGen}()$, $m \in \mathcal{M}$, and $r \leftarrow_\$ \mathcal{R}_{dist}$, we have that $(pk, \mathsf{Hash}(pk, m; r))$ is ϵ-close to uniform over $\mathcal{P} \times \mathcal{D}$.*

Chameleon *For $(pk, sk) \leftarrow_\$ \mathsf{KGen}(), h \in \mathcal{D}, m \in \mathcal{M}$, we have that $h = \mathsf{Hash}(pk, m; \mathsf{Inv}(sk, h, m))$.*

Collision resistance *Given $pk \in \mathcal{P}$, no time-t-bounded adversary can find distinct $(m, r), (m', r')$ with $\mathsf{Hash}(pk, m; r) = \mathsf{Hash}(pk, m'; r')$ with probability greater than ϵ.*

Chameleon indistinguishability *For all $(pk, sk) \leftarrow_\$ \mathsf{KGen}()$, $m \in \mathcal{M}$, and $h \in \mathcal{D}$, $\mathsf{Inv}(sk, h, m)$ is ϵ-close to the distribution of $r \leftarrow_\$ \mathcal{R}_{dist}$ conditioned on $\mathsf{Hash}(pk, m; r) = h$.*

In the initial version of this paper, we used such chameleon hash functions within full-domain-hash and Fiat-Shamir-style signatures to build designated verifier signatures. This required the modeling of the chameleon hash function as a random oracle. However, as was kindly pointed out to us, chameleon hash functions have strong algebraic properties, which makes them inappropriate for instantiating random oracles.

Since random oracles are crucial in the proofs of the DVS constructions in question, we tried to salvage the situation by additionally introducing a standard hash function that could either be applied before or after chameleon hashing. This newly-introduced hash function can be modeled as a random oracle RO, while the chameleon hash function CHF is assumed to only provide the above-defined security properties. Unfortunately, neither order of application (*CHF-then-RO* or *RO-then-CHF*) lead to a secure construction. We will briefly outline the reasons next, using Fiat-Shamir-style signatures as illustrative example; the obstacles are analogous for our attempted full-domain-hash based GPV [46] construction.

Recall that Fiat-Shamir signatures are built on top of a passively secure canonical identification scheme $\mathsf{CID} = (\mathsf{KGen}, \mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$. Signing in our failed DVS construction then was defined as in Figure 12, with the chameleon hash function replacing the regular hash function in the Fiat-Shamir transform:

```
$\underline{\mathsf{FSDVS.Sign}(sk_S, pk_D, m):}$
$(\mathsf{com}, \mathsf{st}) \leftarrow_\$ \mathsf{CID.P}_1(sk_S)$
$r \leftarrow_\$ \mathsf{CHF}.\mathcal{R}_{\text{dist}}$
$\mathsf{ch} \leftarrow \mathsf{CHF.Hash}(pk_D, \mathsf{com} \| m; r)$
$\mathsf{rsp} \leftarrow_\$ \mathsf{CID.P}_2(\mathsf{ch}, \mathsf{st})$
$\sigma \leftarrow (r, (\mathsf{com}, \mathsf{ch}, \mathsf{rsp}))$
return $\sigma$
```

Fig. 12. Fiat-Shamir-style signature using a chameleon hash function.

B.1 CHF-then-RO

As CHF cannot be modeled as a random oracle, one loses the required programmability for the security proofs of the FSDVS construction in Figure 12. One could try to salvage this construction by first applying the chameleon hash function, and then a hash function modeled as a random oracle within the signature. Figure 13 shows this approach, with the changes marked in boxed code.

```
$\underline{\mathsf{FSDVS'.Sign}(sk_S, pk_D, m):}$
$(\mathsf{com}, \mathsf{st}) \leftarrow_\$ \mathsf{CID.P}_1(sk_S)$
$r \leftarrow_\$ \mathsf{CHF}.\mathcal{R}_{\text{dist}}$
$\boxed{h \leftarrow \mathsf{CHF.Hash}(pk_D, m; r)}$
$\boxed{\mathsf{ch} \leftarrow \mathrm{RO}(\mathsf{com} \| h)}$
$\mathsf{rsp} \leftarrow_\$ \mathsf{CID.P}_2(\mathsf{ch}, \mathsf{st})$
$\sigma \leftarrow (r, (\mathsf{com}, \mathsf{ch}, \mathsf{rsp}))$
return $\sigma$
```

Fig. 13. Fiat-Shamir-style signature with RO applied to a CHF digest; changes from Figure 12 in $\boxed{\text{boxes}}$.

Unfortunately, this construction cannot provide source-hiding anymore. Recall that source-hiding requires the existence of an efficient simulator Sim which, given the secret key of the designated verifier and a signer's public key, can output signatures that are indistinguishable from signatures by the sender for the designated verifier.

The original construction given in Figure 12 achieved source hiding with a simulator that first retrieved an accepting conversation $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$, and then utilized the trapdoor sk_D of the chameleon hash function to find randomness r such that $\mathsf{ch} = \mathsf{Hash}(pk_D, \mathsf{com}\|m; r)$.

However, such a simulator does not exist anymore for signatures generated as in Figure 13. Starting from an accepting conversation $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$, the simulator cannot utilize the trapdoor in the chameleon hash function, as it does not know the value h that led to ch, i.e., finding this value h would imply an efficient algorithm that can find preimages of RO. The same obstacle arises in full-domain-hash style signatures based on GPV.

B.2 RO-then-CHF

The above issue clearly stemmed from the fact that the random oracle hid the relevant information for the simulator. But what if we first used the random oracle to generate a uniformly random value and then applied the chameleon hash function to that? Figure 14 depicts this approach, again marking changes in boxed code:

```
$\underline{\mathsf{FSDVS}''.\mathsf{Sign}(sk_S, pk_D, m)}$:
1 $(\mathsf{com}, \mathsf{st}) \leftarrow_\$ \mathsf{CID.P}_1(sk_S)$
2 $\boxed{h \leftarrow \mathrm{RO}(\mathsf{com}\|m)}$
3 $r \leftarrow_\$ \mathsf{CHF}.\mathcal{R}_{\mathrm{dist}}$
4 $\boxed{\mathsf{ch} \leftarrow \mathsf{CHF.Hash}(pk_D, h; r)}$
5 $\mathsf{rsp} \leftarrow_\$ \mathsf{CID.P}_2(\mathsf{ch}, \mathsf{st})$
6 $\sigma \leftarrow (r, (\mathsf{com}, \mathsf{ch}, \mathsf{rsp}))$
7 **return** $\sigma$
```

Fig. 14. Fiat-Shamir-style signatures with CHF applied to a RO digest; changes from Figure 12 in $\boxed{\text{boxes}}$.

Here, unforgeability fails to hold. Recall that the Fiat-Shamir transform crucially relies on the programming of the random oracle that outputs the challenge ch for the prover P_2 to achieve unforgeability of the resulting signature scheme. This is not possible here, as the object that outputs the challenge for the prover is the chameleon hash function and not the random oracle.

For our GPV construction this approach also fails. In the original proof, unforgeability hinges on a reduction from a successful forger to the collision-resistance of the employed preimage-sampleable function. However, since chameleon hashing is a probabilistic process (as opposed to regular hashing) and the forger controls the randomness leading to the eventual signature forgery, no efficient collision-finder can be constructed.

C Summary of Major Changes

- **version 1.0 and 1.0.1 - June 2021**: Initial release and minor editorial changes
- **version 1.1 - August 2021**:
  - removed direct DVS constructions using full-domain-hash GPV and Fiat–Shamir-type approaches leveraging chameleon hash functions due to flawed security proofs; added a discussion of the obstacles in Appendix B
  - added $\mathsf{RingDVS}$ construction in Section 3.2
- **version 1.2 - March 2022**:
  - emphasized differences in the definition of deniability to prior work in Section 1 and Section 4
  - extended comparison with concurrent work ([48] and [29]), especially in light of ring signatures and DVS being equivalent for our case
  - ensured consistent syntax for semi-static keys in Section 7
  - fixed bound for deniability reduction of SPQR to account for pseudorandomness of tPRF in Theorem 6