Post-quantum asynchronous deniable key exchange and the Signal handshake

Jacqueline Brendel
CISPA Helmholtz Center for Information Security
jacqueline.brendel@cispa.de

Rune Fiedler
Technische Universität Darmstadt
rune.fiedler@cryptoplexity.de

Felix Günther
ETH Zürich
mail@felixguenther.info

Christian Janson
Technische Universität Darmstadt
christian.janson@cryptoplexity.de

Douglas Stebila
University of Waterloo
dstebila@uwaterloo.ca

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 KEM-based replacement for the Signal handshake 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 schemes. Furthermore, we show how designated verifier signatures can be built by using chameleon hash functions in both full-domain-hash and Fiat–Shamir-style signature schemes, enabling efficient post-quantum instantiations. This provides the first 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 [62, 63], 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" [63], 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 [62]), 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 [29, 51] 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). See [75] for a detailed analysis of the deniability of X3DH.
- *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. The restrictions on communication flow in an asynchronous protocol are weaker than those of non-interactive key exchange [40].
- *Forward secrecy and post-compromise security [23] 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.

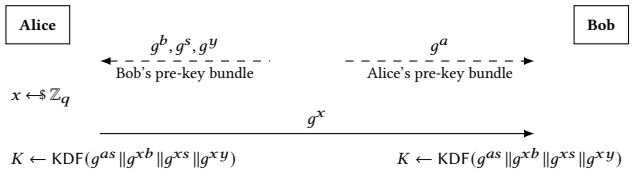

Figure 1: Simplified version of Signal's X3DH handshake. Long-term keys a and b; semi-static key s; ephemeral keys x and y.

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 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).

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 [62] and analyzed by Cohn-Gordon, Cremers, Dowling, Garratt, and Stebila [22] 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 thus on the handshake. 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 Diffie–Hellman 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.

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 [11, 12, 28, 41, 70] but do not have the desired asynchronicity property for reasons similar to the discussion above.

One post-quantum option that avoids the problem with KEMs described above is to use CSIDH [19], 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 [27, 50]. Unfortunately, there are several reasons CSIDH may not be a fully satisfactory solution: it is not part of the current NIST standardization process; it is much more computationally expensive than most other forms of post-quantum cryptography; there is ongoing debate about the security of its concrete parameters [10, 66]; and the decisional form of a related problem [20] is not hard.

Most other post-quantum assumptions used in KEMs, including SIDH [49] and learning-with-errors (LWE) [68], are insecure against key reuse attacks without additional protection such as the Fujisaki–Okamoto transform [42] 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, 31, 32, 35], and one attempt relying on an additional novel assumption [9] the security of which is unknown.

Brendel, Fischlin, Günther, Janson, and Stebila [14] 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 high security notion. They showed how CSIDH and LWE could be used to build split KEMs, but these constructions did not achieve the suitably high security notion required for the Signal-like handshake, effectively leaving the overall problem unsolved.

Unger and Goldberg [73, 74] also consider deniable authenticated key exchange (DAKE) protocols for secure messaging. Their construction relies on primitives for which post-quantum versions are not under consideration for standardization by NIST, ring signatures and dual receiver encryption, the latter of which does not yet appear to have a PQ instantiation in the literature. Their protocol does permit the use of a PQ KEM for ephemeral key exchange.

The recent work by Hashimoto, Katsumata, Kwiatkowski, and Prest [45] is closest to ours. Their core protocol is meant to replace the Signal handshake based on (post-quantum) KEMs and signatures. It achieves strong security against exposure of long-term keys and session state (though not against randomness exposure) similar to Signal and a weaker deniability level by encrypting the (regular) signature exchanged. They provide an implementation for their weakly-deniable protocol and further discuss a theoretical variant achieving stronger deniability based on ring signatures, strong knowledge-type assumptions for plaintext-aware [4] KEMs, and non-interactive zero-knowledge arguments; assessing the post-quantum security of these components is left as an open problem.

1.3 Our Contributions

We show how to construct an asynchronous deniable authenticated key exchange protocol from generic building blocks that can be efficiently instantiated in the post-quantum setting and that can be constructed with small modifications to algorithms in the NIST post-quantum standardization project.

The main tool that allows us to achieve these goals is a *designated verifier signature (DVS) scheme*. Introduced by Jakobsson, Sako, and Impagliazzo [48], DVS schemes allow 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

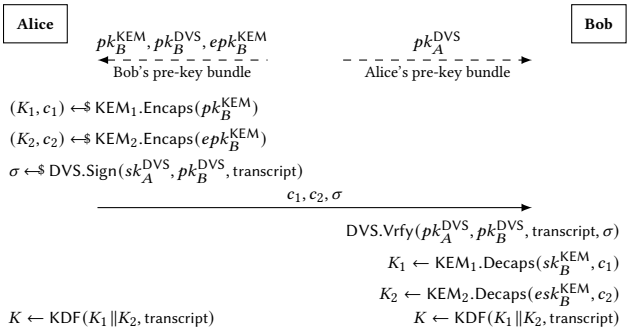

Figure 2: Our core asynchronous DAKE protocol, combining static and ephemeral key encapsulation schemes KEM_1 and KEM_2, and a designated verifier signature DVS.

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.

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 (including [26, 48, 55, 58, 69, 71, 78]), comparatively little is available in the literature on post-quantum DVS schemes. An isogeny-based DVS scheme was proposed in [72] but is insecure due to key reuse attacks identified in [43]. There are several lattice-based DVS schemes which may fit the bill [57, 64, 76, 77, 79], but these have not received much scrutiny in the mainstream cryptographic literature; we summarize this literature and differences to our constructions in Section 3.5. These lattice-based DVS schemes are direct constructions not based on any NIST candidates, so they would require their own thorough analysis.

In contrast, we give two generic constructions of DVS schemes, both of which can be instantiated from post-quantum building blocks much closer to schemes involved in NIST standardization.

- Our first DVS construction is based on the full-domain-hash signature scheme [5], although following the variant by Gentry, Peikert, and Vaikuntanathan [44] which uses a trapdoor function rather than a trapdoor permutation as in [5].
- Our second DVS construction is based on the method of Fiat and Shamir [37] for constructing a signature scheme from an honest-verifier zero-knowledge canonical identification protocol.

In both of these signature schemes, signatures are 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 is to replace the standard hash function with a *chameleon hash function* [18, 53], which allows pre-images of the hash function to be found with knowledge of a trapdoor, which will be held by the verifier. This enables valid-looking signatures to also be constructed in the "backward" direction by the verifier generating a signature for an arbitrary hash then inverting the hash function using the CHF's trapdoor, yielding the source hiding property we need.

Gentry, Peikert, and Vaikuntanathan [44] do provide a lattice-based instantiation of their signature scheme. Moreover, the NIST Round 3 finalist Falcon [39] is an efficient instantiation of the GPV signature scheme relying on the NTRU lattice structure [46]. There are many examples of Fiat–Shamir-based signature schemes. Picnic [21] is a NIST Round 3 alternate candidate, some versions of which are obtained by applying the Fiat–Shamir transform to the ZKB++ proof system which can be viewed as a canonical identification protocol. As for the chameleon hash function, Cash, Hofheinz, Kiltz, and Peikert [18] show how to construct a lattice-based chameleon hash function from the short integer solutions (SIS) problem; and Ducas and Micciancio [34] give a ring-SIS-based version. These CHF constructions are admittedly not under consideration for standardization by NIST; however, their designs are actually quite simple and the connection of their security to the underlying SIS problem is clearly stated, so the parameter analysis of NIST candidates relying on SIS can inform security parameter choices for these chameleon hash functions. In particular, the ring-SIS CHF of [34] can be instantiated with modulus and ring dimension compatible with Falcon [39] at level 1 (128-bit) security, with CHF public key around 12.5 KiB and signature nonce around 25 KiB.

Application to 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 [22] 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. In Section 3 we show how to construct designated verifier signature schemes by using a chameleon hash function (Section 3.2) in the

GPV signature scheme (Section 3.3) and the Fiat–Shamir transform (Section 3.4), showing both the existential unforgeability and source hiding properties for these DVS schemes. 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 on 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 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 2.1 (Key Encapsulation Mechanisms). A *key encapsulation mechanism* KEM is a triple of algorithms KEM = (KGen, Encaps, Decaps). In more detail:

- $\mathsf{KGen}() \xrightarrow{\$} (pk, sk)$: A probabilistic algorithm taking that outputs a public-key/secret-key pair with $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$.
- $\mathsf{Encaps}(pk) \xrightarrow{\$} (c, K)$: 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 KEM = (KGen, Encaps, 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

$$\Pr\left[K' \neq K \;\middle|\; K' \leftarrow \mathsf{Decaps}(sk, c)\right] \leq \delta.$$

$\underline{\mathcal{G}^{\mathsf{indcpa}}_{\mathsf{KEM}}(\mathcal{A})}$:

1 $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}()$
2 $(c^*, K_0^*) \leftarrow\!\!\$\ \mathsf{Encaps}(pk)$
3 $K_1^* \leftarrow\!\!\$\ \mathcal{K}$
4 $b \leftarrow\!\!\$\ \{0,1\}$
5 $b' \leftarrow\!\!\$\ \mathcal{A}(pk, c^*, K_b^*)$
6 **return** $[\![b' = b]\!]$

$\underline{\mathcal{G}^{\mathsf{indcca}}_{\mathsf{KEM}}(\mathcal{A})}$:

1 $(pk, sk) \leftarrow\!\!\$\ \mathsf{KGen}()$
2 $(c^*, K_0^*) \leftarrow\!\!\$\ \mathsf{Encaps}(pk)$
3 $K_1^* \leftarrow\!\!\$\ \mathcal{K}$
4 $b \leftarrow\!\!\$\ \{0,1\}$
5 $b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Decaps}}(pk, c^*, K_b^*)$
6 **return** $[\![b' = b]\!]$

$\underline{\textsc{Decaps}(c)}$:

7 **if** $c = c^*$
8 **return** $\perp$
9 **else**
10 **return** $\mathsf{Decaps}(sk, c)$

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

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:

*Definition 2.2 (*IND-ATK *Security of KEMs).* Let KEM = (KGen, Encaps, 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, resp. and making at most Q_D queries to the decapsulation 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 atk = cpa, resp. atk = 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* [41, 54]. In the following we recall the respective definitions and security games.

Definition 2.3. Let $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^\iota \rightarrow \{0,1\}^\omega$ 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

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

$\underline{\mathcal{G}^{\mathrm{prfsec}}_{\mathsf{F}}(\mathcal{A})}$:

$K \leftarrow\$ \{0,1\}^{\kappa}$
$g \leftarrow\$ \{\text{functions } f : \{0,1\}^{\iota} \to \{0,1\}^{\omega}\}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\text{PRFCHALLENGE}}()$
return $[\![b' = b]\!]$

$\underline{\text{PRFCHALLENGE}(x)}$:

if $b = 0$
  return $\mathsf{F}(K, x)$
else
  return $g(x)$

$\underline{\mathcal{G}^{\mathrm{tprfsec}}_{\mathsf{F},q}(\mathcal{A})}$:

$g \leftarrow\$ \{\text{functions } f : \{0,1\}^{\iota} \to \{0,1\}^{\omega}\}$
$g' \leftarrow\$ \{\text{functions } f : \{0,1\}^{\kappa} \to \{0,1\}^{\omega}\}$
$K, K' \leftarrow\$ \{0,1\}^{2\kappa}$
$b \leftarrow\$ \{0,1\}$
$x, x_1, x_2, \ldots, x_q \leftarrow\$ \{0,1\}^{(q+1)\iota}$
$s_0 \leftarrow \{(x_1, \mathsf{F}(K, x_1)), (x_2, \mathsf{F}(K, x_2)), \ldots, (x_q, \mathsf{F}(K, x_q)), (K', \mathsf{F}(K', x))\}$
$s_1 \leftarrow \{(x_1, g(x_1)), (x_2, g(x_2)), \ldots, (x_q, g(x_q)), (K', g'(K'))\}$
$b' \leftarrow\$ \mathcal{A}(s_b)$
return $[\![b' = b]\!]$

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

Note that one can easily build a twisted PRF tPRF from a PRF F in the standard model. Following Kurosawa and Furukawa [54], 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 [48]. 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. This property is typically modeled by requiring that the designated verifier can efficiently simulate signatures that are indistinguishable from signatures produced by the signer.

In this section we give two generic constructions of DVS schemes, both of which can be instantiated in the post-quantum setting. A key ingredient in our construction is a chameleon hash function. Previously, Yang, Yu, and Sun [78] used chameleon hash functions with collision trapdoors to build a variant of a DVS scheme; in contrast, we use CHF with a trapdoor that enables preimage sampling.

3.1 DVS Definitions

Definition 3.1. A *designated verifier signature scheme* (DVS) is a tuple of algorithms DVS = (SKGen, VKGen, Sign, Vrfy, Sim) along with a message space $\mathcal{M}$.

- $\mathsf{SKGen}() \xrightarrow{\$} (pk_S, sk_S)$: 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.

$\underline{\mathcal{G}^{\mathrm{eufcma}}_{\mathrm{DVS}}(\mathcal{A})}$:

$Q \leftarrow \emptyset$
$(pk_S, sk_S) \leftarrow\$ \mathrm{DVS.SKGen}()$
$(pk_D, sk_D) \leftarrow\$ \mathrm{DVS.VKGen}()$
$(m^{\star}, \sigma^{\star}) \leftarrow\$ \mathcal{A}^{\text{SIGN}}(pk_S, pk_D)$
$d \leftarrow \mathrm{DVS.Vrfy}(pk_S, pk_D, m^{*}, \sigma^{*})$
return $[\![d = \mathsf{true} \ \wedge \ m^{*} \notin Q]\!]$

$\underline{\text{SIGN}(pk, m)}$:

if $pk = pk_D$
  $Q \leftarrow Q \cup \{m\}$
$\sigma \leftarrow\$ \mathrm{DVS.Sign}(sk_S, pk, m)$
return σ

$\underline{\mathcal{G}^{\mathrm{srchid}}_{\mathrm{DVS}}(\mathcal{A})}$:

$(pk_S, sk_S) \leftarrow\$ \mathrm{DVS.SKGen}()$
$(pk_D, sk_D) \leftarrow\$ \mathrm{DVS.VKGen}()$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\text{CHALL}}(pk_S, sk_S, pk_D, sk_D)$
return $[\![b' = b]\!]$

$\underline{\text{CHALL}(m)}$:

if $b = 0$
  $\sigma \leftarrow\$ \mathrm{DVS.Sign}(sk_S, pk_D, m)$
else
  $\sigma \leftarrow\$ \mathrm{DVS.Sim}(pk_S, sk_D, m)$
return σ

Figure 5: Existential unforgeability under chosen-message attacks (top) and source hiding (bottom) of a designated verifier signature scheme DVS**.**

- $\mathsf{Sign}(sk_S, pk_D, m) \xrightarrow{\$} \sigma$: 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.
- $\mathsf{Vrfy}(pk_S, pk_D, m, \sigma) \to \mathsf{true/false}$: 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

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

We follow [55] in defining separate key generation algorithms for signer and designated verifier. While in some cases these two algorithms may be identical, they differ for our constructions. Some DVS schemes, called *strong* DVS, are written with a verification algorithm that requires the designated verifier's secret key instead of the public key.

A long line of research has scrutinized the security of DVS schemes [26, 48, 55, 58, 69, 71, 78] in different settings. For the purpose of this paper, it suffices to define the security notions of *unforgeability* and *source hiding*. Unforgeability for DVS is like that of standard signature schemes, except the signing oracle permits the adversary to supply the public key of the designated verifier.

Definition 3.2. A designated verifier signature scheme DVS is (t, ϵ, Q_S)-*existentially unforgeable under chosen-message attacks* if, for any adversary $\mathcal{A}$ with running time at most t and making at most Q_S queries to the signing oracle, we have that

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

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

The second property we consider is called source hiding [55], where it should be infeasible for an attacker to determine whether a given signature has been generated by the signer or by the designated verifier, even if the attacker learns the secret keys.

Definition 3.3. A designated verifier signature scheme DVS is (t, ϵ, Q_C)-*source hiding* if, for any adversary $\mathcal{A}$ with running time at most t and making at most Q_C to the challenge 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.

This property of source hiding also appears under different terms in the literature such as the designated verifier property [48, 69], non-transferability [71], source deniable [38], untransferability [15], and recently off-the-record [26]. While all these definitions share the intuition that the sender can blame another party (in particular, the designated receiver) as originator of the signature, they vary in the attacker capabilities, i.e., whether the attacker is unbounded or whether it gets access to the secret keys.

3.2 Chameleon Hash Functions

For our constructions of DVS we need a chameleon hash function [53]. We use the formalization of Cash, Hofheinz, Kiltz, and Peikert [18], where the trapdoor enables preimage sampling (unlike [53], where the trapdoor enables collision sampling).

Definition 3.4. 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 $\mathcal{R}_{\text{dist}}$ 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 $r \in \mathcal{R}$.

Definition 3.5. A CHF is (t, ϵ)-secure if it satisfies:

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

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

Chameleon hash functions as random oracles. In proofs later in this section we model $\mathsf{CHF} = (\mathsf{KGen}, \mathsf{Hash}, \mathsf{Inv})$ as a random oracle using lazy sampling. This is achieved by providing the adversary with two oracles: RO and InvRO.

- $\mathrm{RO}(pk, m, r) \xrightarrow{\$} (r', h)$: Models the idealized computation of Hash. It takes as input a public key pk, the data m to be hashed, and randomness $r \in \mathcal{R} \cup \{\varepsilon\}$. Provision of the hashing randomness is required when recomputing an already existing digest, e.g., during signature verification. If no randomness is given, i.e., $r = \varepsilon$, the random oracle will choose the randomness according to the distribution $\mathcal{R}_{\text{dist}}$. It then outputs the randomness r' and the lazily-sampled digest h.
- $\textsc{InvRO}(sk, h, m) \xrightarrow{\$} r$: Provides the idealized execution of the Inv algorithm. It takes as input a secret key sk, the desired digest h, and the data m to be hashed to this digest and outputs randomness r, which was lazily-sampled from $\mathcal{R}_{\text{dist}}$.

Note that the chameleon property of CHF requires that $h = \mathsf{Hash}(pk, m; \mathsf{Inv}(sk, h, m))$. Therefore, consistency between RO and InvRO also must be ensured. We achieve this via standard bookkeeping techniques under the assumption that given a secret key sk, we can compute the corresponding public key pk.[1]

In more detail, Q_{RO} many queries (pk, m, r) will be stored along with the corresponding response (r', h) as tuples (pk, m, r', h) in the list $\mathcal{L}_{\mathrm{RO}}$. For the proof of Theorem 3.8 we will save an additional element in the list, i.e., entries of the form (pk, m, r', s, h). Note that if $r \neq \varepsilon$ in the query, then $r' = r$. For queries to InvRO which are of the form (sk, m, h) with response r, the secret key sk is first converted into the corresponding public key pk and then the tuple (pk, m, r, h) is also stored in $\mathcal{L}_{\mathrm{RO}}$. Hence, $\mathcal{L}_{\mathrm{RO}}$ contains up to $Q_{\mathrm{RO}} + Q_{\textsc{InvRO}}$ many entries (less entries in case a RO queries asks for an input set by InvRO). In the proofs, the adversary cannot use the InvRO oracle on keys that are relevant for the reduction since it does not know the corresponding secret keys.

3.3 DVS from Chameleon Hashing in GPV Signatures

Gentry, Peikert, and Vaikuntanathan [44] showed how to construct a hash-then-sign signature scheme analogous to the full-domain hash signature scheme [5], but using a preimage sampleable trapdoor function rather than a permutation. In this section we show that using a chameleon hash function in the GPV construction results in a designated verifier signature scheme.

3.3.1 Preimage sampleable functions and GPV signatures.

Definition 3.6. A *preimage sampleable function (PSF)* [44] is defined by four algorithms $(\mathsf{TrapGen}, \mathsf{SampleDom}, \mathsf{SamplePre}, f)$ and associated spaces $\mathcal{D}, \mathcal{R}$ (for domain and range of f):

- $\mathsf{TrapGen}() \xrightarrow{\$} (pk, sk)$: A probabilistic trapdoor generation algorithm that outputs a public-key/secret-key pair.
- $\mathsf{SampleDom}() \xrightarrow{\$} x$: A probabilistic domain sampling algorithm that outputs a sample $x \in \mathcal{D}$.
- $\mathsf{SamplePre}(sk, y) \xrightarrow{\$} x$: A probabilistic preimage sampling algorithm that takes as input a secret key from a key pair (pk, sk) output by $\mathsf{TrapGen}()$ and $y \in \mathcal{R}$, and outputs $x \in \mathcal{D}$.
- $f(pk, x) \to y$: A deterministic function taking as input a public key and a value $x \in \mathcal{D}$ and outputting $y \in \mathcal{R}$.

Definition 3.7. A PSF is (t, ϵ)-secure if, for $(pk, sk) \xleftarrow{\$} \mathsf{TrapGen}()$, we have:

Uniformity $\mathsf{SampleDom}()$ yields x from a (possibly non-uniform) distribution over $\mathcal{D}$ such that $f(pk, x)$ is ϵ-close to uniform over $\mathcal{R}$.

[1] This is a reasonable assumption to make since public keys pk can often be considered to be computed as $pk \leftarrow \mathsf{KGen}(\mathsf{params}; sk)$, where sk is simply the randomness within the execution of KGen. Thus, given sk and the public parameters of the scheme, one can deterministically recompute the public key.

GPV.KGen():

1 $(pk, sk) \leftarrow\$ \mathsf{PSF.TrapGen}()$
2 **return** (pk, sk)

GPV.Sign(sk, m):

3 $r \leftarrow\$ \mathcal{R}$
4 $c \leftarrow \mathsf{H}(m \| r)$
5 $s \leftarrow\$ \mathsf{PSF.SamplePre}(sk, c)$
6 $\sigma \leftarrow (r, s)$
7 **return** σ

GPV.Vrfy(pk, m, σ):

8 $(r, s) \leftarrow \sigma$
9 **if** $s \notin \mathsf{PSF}.\mathcal{D}$ **then return** false
10 **if** $r \notin \mathcal{R}$ **then return** false
11 $c \leftarrow \mathsf{H}(m \| r)$
12 **if** $\mathsf{PSF}.f(pk, s) = c$ **then return** true
13 **else return** false

Figure 6: Signature scheme $\mathsf{GPV} = \mathsf{GPV}[\mathsf{H}, \mathsf{PSF}, \mathcal{R}]$ **of [44] constructed from a hash function** H**, a preimage sampleable function** PSF**, and signature randomness space** $\mathcal{R}$**.**

Preimage sampling with trapdoor For $y \in \mathcal{R}$, $\mathsf{SamplePre}(sk, y)$ outputs x ϵ-close to the distribution of $x \leftarrow\$ \mathsf{SampleDom}()$ conditioned on $f(pk, x) = y$.

One-way without trapdoor For any time-t-bounded adversary $\mathcal{A}$, the probability that $y = f(pk, \mathcal{A}(pk, y))$ is at most ϵ, where the probability is taken over $(pk, sk) \leftarrow\$ \mathsf{TrapGen}()$, $y \leftarrow\$ \mathcal{R}$, and $\mathcal{A}$'s random coins.

Preimage min-entropy For every $y \in \mathcal{R}$, the conditional min-entropy of $x \leftarrow\$ \mathsf{SampleDom}()$ given $f(pk, x) = y$ is at least $\omega(\log\log|\mathcal{D}|)$.

Collision-resistant without trapdoor For any time-t-bounded adversary $\mathcal{A}$, the probability that $\mathcal{A}(pk)$ outputs distinct $x, x' \in \mathcal{D}$ such that $f(pk, x) = f(pk, x')$ is at most ϵ, where the probability is taken over $(pk, sk) \leftarrow\$ \mathsf{TrapGen}()$ and $\mathcal{A}$'s coins.

A PSF that has preimage min-entropy and is collision-resistant without trapdoor is also one-way without trapdoor.

The GPV signature scheme is shown in Figure 6. It is constructed from a preimage sampleable function PSF, a signature randomness space $\mathcal{R}$, and a hash function $\mathsf{H} : \{0,1\}^* \times \mathcal{R} \to \mathsf{PSF}.\mathcal{R}$.

Gentry, Peikert, and Vaikuntanathan [44] show that the GPV signature scheme in Figure 6 is (strongly) existentially unforgeable under chosen message attack when the preimage sampleable function has preimage min-entropy and is collision-resistant without trapdoor, and the hash function is modeled as a random oracle.

Our construction. We show that if we use a chameleon hash function in the GPV signature scheme, where the digest space of the CHF is the same as the range of the PSF, then we can obtain a designated verifier signature scheme that is unforgeable and source hiding. The construction is shown in Figure 7. The signature randomness space $\mathcal{R}$ in the GPV scheme is set to be $\mathsf{CHF}.\mathcal{R}$, the randomness space of the chameleon hash function.

Theorem 3.8 (Unforgeability of GPVDVS). *If* PSF *is a* $(t, \epsilon_{\mathsf{PSF}})$*-secure preimage sampleable function and* CHF *is modeled as a random oracle, then* GPVDVS *as shown in Figure 7 is* $(t', \epsilon', Q_S + Q_{RO} + Q_{InvRO})$ *existentially unforgeable under chosen message attacks, with* $t' \approx t$ *and* $\epsilon' \leq \epsilon_{\mathsf{PSF}} \cdot (1 - 2^{-\omega(\log\log|\mathsf{PSF}.\mathcal{D}|)})$.

Proof. The proof idea follows [44, Prop. 6.2]. We reduce existential unforgeability of GPVDVS to collision resistance without trapdoor of PSF, while [44] reduces from strong unforgeability of the signature scheme. Furthermore, the treatment of the hash function differs.

GPVDVS.SKGen():

1 $(pk_S, sk_S) \leftarrow\$ \mathsf{PSF.TrapGen}()$
2 **return** (pk_S, sk_S)

GPVDVS.VKGen():

3 $(pk_D, sk_D) \leftarrow\$ \mathsf{CHF.KGen}()$
4 **return** (pk_D, sk_D)

GPVDVS.Sign(sk_S, pk_D, m):

5 $r \leftarrow\$ \mathsf{CHF}.\mathcal{R}_{\mathsf{dist}}$
6 $c \leftarrow \mathsf{CHF.Hash}(pk_D, m; r)$
7 $s \leftarrow\$ \mathsf{PSF.SamplePre}(sk_S, c)$
8 $\sigma \leftarrow (r, s)$
9 **return** σ

GPVDVS.Vrfy(pk_S, pk_D, m, σ):

10 $(r, s) \leftarrow \sigma$
11 **if** $s \notin \mathsf{PSF}.\mathcal{D}$ **then return** false
12 **if** $r \notin \mathsf{CHF}.\mathcal{R}$ **then return** false
13 $c \leftarrow \mathsf{CHF.Hash}(pk_D, m; r)$
14 **if** $\mathsf{PSF}.f(pk_S, s) = c$ **then return** true
15 **else return** false

GPVDVS.Sim(pk_S, sk_D, m):

16 $s \leftarrow\$ \mathsf{PSF.SampleDom}()$
17 $c \leftarrow \mathsf{PSF}.f(pk_S, s)$
18 $r \leftarrow\$ \mathsf{CHF.Inv}(sk_D, c, m)$
19 $\sigma \leftarrow (r, s)$
20 **return** σ

Figure 7: Designated-verifier signature scheme $\mathsf{GPVDVS} = \mathsf{GPVDVS}[\mathsf{CHF}, \mathsf{PSF}]$ **constructed from a chameleon hash function** CHF **and a preimage sampleable function** PSF **satisfying** $\mathsf{CHF}.\mathcal{D} = \mathsf{PSF}.\mathcal{R}$**.**

In our setting, the chameleon hash function (modeled according to Section 3.2) is called on the designated verifier's public key pk_D, the message m, and the randomness r.

We can further assume that queries to the RO and the signing oracle are not repeatedly asked by $\mathcal{A}$ as the consistent random oracle response yields no new information.

Initialization of $\mathcal{A}$ The adversary $\mathcal{B}$ against collision resistance without trapdoor of PSF receives as input a public key pk_{PSF}. Next, $\mathcal{B}$ sets $pk_S \leftarrow pk_{\mathsf{PSF}}$ and generates a key pair $(pk_D, sk_D) \leftarrow\$ \mathsf{CHF.KGen}()$. The reduction then initializes the EUF-CMA adversary $\mathcal{A}$ on input (pk_S, pk_D).

Queries to RO For any query (pk, m, r), the reduction first checks if an entry $(pk, m, r, \cdot, h)$ exists in $\mathcal{L}_{\mathrm{RO}}$. If yes, it returns (r, h). Else, it sets $r' \leftarrow r$ if $r \neq \epsilon$ and $r' \leftarrow\$ \mathsf{CHF}.\mathcal{R}_{\mathsf{dist}}$ otherwise. It then samples $s \leftarrow\$ \mathsf{PSF.SampleDom}()$, sets $h \leftarrow \mathsf{PSF}.f(pk_S, s)$, saves (pk, m, r', s, h) in $\mathcal{L}_{\mathrm{RO}}$, and returns (r', h).

Queries to InvRO For any query (sk, h, m) the reduction computes pk from sk, samples $r \leftarrow\$ \mathsf{CHF}.\mathcal{R}_{\mathsf{dist}}$, saves $(pk, m, r, \perp, h)$ to $\mathcal{L}_{\mathrm{RO}}$, and returns r.

Queries to Sign $\mathcal{A}$'s queries to the Sign oracle are of the form (pk, m). For each of the Q_S queries, the reduction $\mathcal{B}$ first samples $r \leftarrow\$ \mathsf{CHF}.\mathcal{R}_{\mathsf{dist}}$ and queries the random oracle on (pk, m, r). It finds an entry (pk, m, r, s, h) for any s, h in $\mathcal{L}_{\mathrm{RO}}$ and returns s. If $s = \perp$, $\mathcal{B}$ restarts with sampling new randomness r. Note that this case only occurs if the sampled randomness is identical to the randomness that was sampled for a previous InvRO query on the same public key and message. For signing queries (pk_D, m) including the challenge public key, $\mathcal{B}$ additionally records m in the list $\mathcal{L}_{\mathrm{Sign}}$.

Existential Forgery. At some point, $\mathcal{A}$ outputs a valid DVS forgery $(m^*, \sigma^* = (r^*, s^*))$ wrt. pk_S and pk_D. Recall that the validity of the forgery implies:

- $m^* \notin \mathcal{L}_{\mathrm{Sign}}$, i.e., m^* has not been queried to the signing oracle for the designated verifier's public key,
- $s^* \in \mathsf{PSF}.\mathcal{D}$, i.e., s^* is a valid domain element of PSF,
- $r^* \in \mathsf{CHF}.\mathcal{R}$, i.e., r^* is a valid randomness of the CHF, and
- $\mathsf{CHF.Hash}(pk_D, m^*, r^*) = \mathsf{PSF}.f(pk_S, s^*)$, i.e., the signature contains a preimage for the chameleon hash of the message.

We assume that the attacker has queried RO on $(pk_D, m^*, \cdot)$ before returning the forgery. The reduction checks $\mathcal{L}_{\text{RO}}$ for an entry (pk_D, m^*, r^*, s, h^*) with $h^* = \text{CHF.Hash}(pk_D, m^*, r^*)$ and outputs (s, s^*) as a collision in PSF under pk_S if $s \neq s^*$.

The reduction soundly simulates the unforgeability game against GPVDVS. The random oracle provides answers with randomness sampled according to the distribution $\mathcal{R}_{\text{dist}}$ for both RO and InvRO. For RO the digests are computed by sampling a domain element of PSF and applying $\text{PSF}.f$ to it. According to the uniformity property of the PSF these values are distributed uniformly. Thus, the signing oracle answers consistently with the random oracle by construction.

Since the attacker has queried RO on (pk_D, m^*), there is an entry (pk_D, m^*, r^*, s, h^*) in $\mathcal{L}_{\text{RO}}$ with $h^* = \text{CHF.Hash}(pk_D, m^*, r^*)$. The min-entropy of s^* conditioned on $\text{PSF}.f(pk_S, s^*) = h^*$ is $\omega(\log\log|\text{PSF}.\mathcal{D}|)$ due to the preimage min-entropy property of PSF. Hence, $s^* \neq s$ except with small probability $2^{-\omega(\log\log|\text{PSF}.\mathcal{D}|)}$ and (s, s^*) is a collision wrt. PSF and pk_S.

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 random oracle is not expensive. If $\mathcal{A}$ outputs a successful forgery with probability ϵ', then $\mathcal{B}$ is able to produce a collision without trapdoor wrt. PSF and pk_S with probability $\epsilon' \leq \epsilon_{\text{PSF}} \cdot (1 - 2^{-\omega(\log\log|\text{PSF}.\mathcal{D}|)})$. □

Theorem 3.9 (Source hiding of GPVDVS). *If* PSF *is a* $(t, \epsilon_{\text{PSF}})$*-secure preimage sampleable function and* CHF *is a* $(t, \epsilon_{\text{CHF}})$*-secure chameleon hash function, then* GPVDVS *is* $(t', 2\epsilon_{\text{PSF}} + 2\epsilon_{\text{CHF}}, Q_C)$*-source hiding with* $t \approx t'$.

Proof. We need to show that GPVDVS.Sign and GPVDVS.Sim output essentially the same distribution of signatures, for fixed public-key/secret-key pairs and any fixed message m. We do so by arguing that both algorithms generate values (r, c, s) close to the distribution, call it Δ, generated by the following procedure: sample $c \leftarrow\!\!\$\ \text{CHF}.\mathcal{D}$, then sample (r, s) from the conditional distribution over the product of the distribution $\text{CHF}.\mathcal{R}_{\text{dist}}$ on $\text{CHF}.\mathcal{R}$ and the distribution on $\text{PSF}.\mathcal{D}$ output by PSF.SampleDom, conditioned on $c = \text{CHF.Hash}(pk_D, m; r) = \text{PSF}.f(pk_S, s)$. Note $\text{CHF}.\mathcal{D} = \text{PSF}.\mathcal{R}$.

First consider (r, c, s) in GPVDVS.Sign. By the uniformity property of CHF, lines 5 and 6 of Figure 7 result in c ϵ_{CHF}-close to uniformly distributed over $\text{CHF}.\mathcal{D}$, with r sampled from the distribution $\text{CHF}.\mathcal{R}_{\text{dist}}$ conditioned on $c = \text{CHF.Hash}(pk_D, m; r)$. By the preimage sampling with trapdoor property of PSF, line 7 of Figure 7 outputs s that is ϵ_{PSF}-close to the distribution of PSF.SampleDom conditioned on $c = \text{PSF}.f(pk_S, s)$. Thus, (r, c, s) in GPVDVS.Sign are $\epsilon_{\text{CHF}} + \epsilon_{\text{PSF}}$-close to Δ.

Now consider (r, c, s) in GPVDVS.Sim. By the uniformity property of PSF, lines 16 and 17 of Figure 7 result in c ϵ_{PSF}-close to uniformly distributed over $\text{PSF}.\mathcal{R} = \text{CHF}.\mathcal{D}$, with s sampled from the distribution output by PSF.SampleDom conditioned on $c = \text{PSF}.f(pk_S, s)$. By the chameleon indistinguishability property of CHF, line 18 of Figure 7 outputs r that is ϵ_{CHF}-close to the distribution $r \leftarrow\!\!\$\ \text{CHF}.\mathcal{R}_{\text{dist}}$ conditioned on $\text{CHF.Hash}(pk_D, m; r) = c$. Thus, (r, c, s) in GPVDVS.Sim are $\epsilon_{\text{PSF}} + \epsilon_{\text{CHF}}$-close to Δ.

Hence, (r, c, s) in GPVDVS.Sign and GPVDVS.Sim are $(2\epsilon_{\text{PSF}} + 2\epsilon_{\text{CHF}})$-close, so GPVDVS is source hiding as required. □

Instantiation. The GPVDVS scheme can be instantiated by using the existing preimage sampleable function at the heart of Falcon [39] (which is a signature scheme in the GPV framework) and either the plain SIS-based chameleon hash function of [18, §4.1] or the ring-SIS based CHF of [34, §B.3]. Either way, we must ensure that the range of the CHF and the range of the PSF are the same (or have an efficient mapping between them). For Falcon, the range of the PSF are polynomials in $\mathcal{R}_q$ where $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$; Falcon's level 1 parameters have $n = 512$ and $q = 12289$. The ring-SIS-based CHF of [34] can be instantiated at the 128-bit security level with the same n and q, yielding a CHF public key of 12.5 KiB and CHF randomness of 25 KiB.[2] For the plain-SIS-based CHF of [18], we must use a much larger modulus in the CHF ($q \approx 2^{47}$), and then need to adapt the Falcon parameters accordingly; at the 128-bit security level this yields a CHF public key of 287 MiB, CHF randomness of 0.5 MiB, Falcon public key of 3 KiB, and Falcon signature of 0.7 KiB.

3.4 DVS from Chameleon Hashing in Fiat–Shamir Signatures

In this section, we give the full construction of designated verifier signatures via the Fiat-Shamir transform with chameleon hashing. We first recall some of the basic definitions and formalize the construction before re-stating the theorem and proving it in full.

3.4.1 Definitions. Fiat-Shamir signatures are traditionally built from canonical identification protocols CID. These are three-move challenge-response protocols between a prover P and a verifier V. In order to build existentially unforgeable signatures from canonical identification protocols, we need them to have a sufficient level of min-entropy, to be honest-verifier zero-knowledge, and to be secure against impersonation attacks. These properties are defined formally as follows:

Definition 3.10. A canonical identification protocol CID is defined by three algorithms (KGen, P, V):

- $\text{KGen}() \xrightarrow{\$} (pk, sk)$: A probabilistic key generation algorithm that outputs a public-key/secret-key pair.
- The prover $\text{P} = (\text{P}_1, \text{P}_2)$ is a two-stage algorithm that takes as input the secret key sk. P_1 starts taking sk as input and outputs a commitment com along with some state st. P_2 takes as input the challenge ch (provided by V_1) and state st outputting a response rsp.
- The verifier $\text{V} = (\text{V}_1, \text{V}_2)$ is a two-stage algorithm that takes as input the public key. V_1 samples a random challenge ch and sends it to the prover. V_2 takes as input the public key as well as the tuple (com, ch, rsp) and outputs true if it accepts the conversation or false otherwise.

For all $(pk, sk) \leftarrow \text{KGen}()$, we require that any honest interaction between $\text{P}(sk)$ and $\text{V}(pk)$ within an instance of the protocol results into the verifier accepting. That is, for $(\text{com}, \text{st}) \leftarrow\!\!\$\ \text{P}_1(sk)$, $\text{ch} \leftarrow\!\!\$\ \text{V}_1()$, and $\text{rsp} \leftarrow\!\!\$\ \text{P}_2(sk, \text{com}, \text{ch}, \text{st})$, we have that $\Pr[\text{V}_2(pk, \text{com}, \text{ch}, \text{rsp}) = \text{true}] = 1$.

We denote the interactive execution of the canonical identification protocol between the prover and the verifier by $\text{P}(sk) \rightleftarrows \text{V}(pk)$. Additionally, we write $\text{Trans}[\text{P}(sk) \rightleftarrows \text{V}(pk)]$ to simply denote a transcript (com, ch, rsp) generated from the interaction between P and V.

[2] We used the SIS security estimator from PQ-Crystals: https://github.com/pq-crystals/security-estimates and the Falcon parameters script from the NIST Round 3 submission.

$\underline{\mathcal{G}^{\text{impkoa}}_{\text{CID}}(\mathcal{A})}$:

1 $(pk, sk) \leftarrow\!\$$ CID.KGen()
2 $(\text{com}, \text{st}) \leftarrow\!\$\ \mathcal{A}(pk)$
3 $\text{ch} \leftarrow\!\$$ CID.$\text{V}_1()$
4 $\text{rsp} \leftarrow\!\$\ \mathcal{A}(\text{ch}, \text{st})$
5 **return** $[\![\text{CID.V}_2(pk, \text{com}, \text{ch}, \text{rsp})]\!]$

$\underline{\mathcal{G}^{\text{imppa}}_{\text{CID}}(\mathcal{A})}$:

1 $(pk, sk) \leftarrow\!\$$ CID.KGen()
2 $(\text{com}, st) \leftarrow\!\$\ \mathcal{A}(pk)$
3 $\text{ch} \leftarrow\!\$$ CID.$\text{V}_1()$
4 $\text{rsp} \leftarrow\!\$\ \mathcal{A}^{\text{Transcript}}(\text{ch}, st)$
5 **return** $[\![\text{CID.V}_2(pk, \text{com}, \text{ch}, \text{rsp})]\!]$

Transcript:

6 **return** $\text{Trans}[\text{CID.P}(sk) \rightleftarrows \text{CID.V}(pk)]$

Figure 8: Security experiments for IMP-KOA and IMP-PA of CID against impersonating adversaries $\mathcal{A}$.

Definition 3.11. A canonical identification protocol CID = (KGen, P, V) has α bits of *min-entropy* if the probability over the random choice $(pk, sk) \leftarrow\!\$\ \text{KGen}()$ that the commitment com generated by $\text{P}_1(sk)$ is from a distribution with at least α bits of min-entropy is at least $1 - 2^{\alpha}$.

Definition 3.12. Let CID = (KGen, P, V) be a canonical identification protocol. We say that CID is ϵ_{ZK}*-honest-verifier zero-knowledge*, or ϵ_{ZK}-HVZK, if there exists an algorithm Sim, called the simulator, such that for all $(pk, sk) \leftarrow\!\$\ \text{KGen}()$, the outputs of $\text{Sim}(pk)$ can only be distinguished from real conversations between P and V with probability at most ϵ_{ZK}.

Definition 3.13. Let CID = (KGen, P, V) be a canonical identification protocol and let $\mathcal{A}$ be an algorithm. First consider the security game $\mathcal{G}^{\text{impkoa}}_{\text{CID}}(\mathcal{A})$ as provided on the left of Figure 8. We say that CID is (t, ϵ)-secure against impersonation attacks under key-only attacks, (t, ϵ)-IMP-KOA-secure, if for any adversary $\mathcal{A}$ running in time at most t, we have $\Pr[\mathcal{G}^{\text{impkoa}}_{\text{CID}}(\mathcal{A}) = 1] \leq \epsilon$.

Similarly, consider the security game $\mathcal{G}^{\text{imppa}}_{\text{CID}}(\mathcal{A})$ as provided on the right of Figure 8. We say that CID is (t, ϵ, Q_T)-secure against impersonation attacks under passive attacks, (t, ϵ, Q_T)-IMP-PA-secure, if for any adversary $\mathcal{A}$ running in time at most t and with at most Q_T queries to the oracle Transcript, we have $\Pr[\mathcal{G}^{\text{imppa}}_{\text{CID}}(\mathcal{A}) = 1] \leq \epsilon$.

3.4.2 Construction and security. The construction of the DVS scheme denoted by FSDVS[CID, CHF] from a canonical identification scheme CID and a chameleon hash function CHF is given in Figure 9. The key idea is that using a chameleon hash function in the Fiat–Shamir transform enables the designated-verifier property. As usual for the Fiat–Shamir transform we require that the digest space $\mathcal{D}$ of the (chameleon) hash function equals the challenge set of the canonical identification protocol.

To show that FSDVS[CID, CHF] is a secure DVS scheme, we need to show existential unforgeability as well as source hiding:

Theorem 3.14 (Unforgeability of FSDVS). *Let* CID = (KGen, P, V) *be a* (t, ϵ, Q_T)-IMP-PA*-secure canonical identification protocol with commitments that have* α *bits min-entropy. Let further* CHF = (KGen, Hash, Inv) *be a secure chameleon hash function, modeled as a programmable random oracle.*

Then the Fiat–Shamir transformed designated verifier signature scheme FSDVS[CID, CHF] *as defined in Figure 9 is* $(t', \epsilon', (Q_S + Q_{RO} + Q_{InvRO}))$-EUF-CMA *secure according to Definition 3.2, where* $t \approx t' \quad \text{and} \quad \epsilon' \leq Q_{RO} \cdot (\epsilon + Q_S(Q_{RO} + Q_{InvRO}) \cdot 2^{-\alpha})$.

FSDVS.SKGen():

1 $(pk_S, sk_S) \leftarrow\!\$$ CID.KGen()
2 **return** (pk_S, sk_S)

FSDVS.VKGen():

3 $(pk_D, sk_D) \leftarrow\!\$$ CHF.KGen()
4 **return** (pk_D, sk_D)

FSDVS.Sign(sk_S, pk_D, m):

5 $(\text{com}, \text{st}) \leftarrow\!\$$ CID.$\text{P}_1(sk_S)$
6 $r \leftarrow\!\$$ CHF.$\mathcal{R}_{\text{dist}}$
7 $\text{ch} \leftarrow \text{CHF.Hash}(pk_D, \text{com} \| m; r)$
8 $\text{rsp} \leftarrow\!\$$ CID.$\text{P}_2(\text{ch}, \text{st})$
9 $s \leftarrow (\text{com}, \text{ch}, \text{rsp})$
10 $\sigma \leftarrow (r, s)$
11 **return** σ

FSDVS.Vrfy(pk_S, pk_D, m, σ):

12 $(r, (\text{com}, \text{ch}, \text{rsp})) \leftarrow \sigma$
13 $\text{ch}' \leftarrow \text{CHF.Hash}(pk_D, \text{com} \| m; r)$
14 **if** $\text{ch} \neq \text{ch}'$ **then return** false
15 $d \leftarrow \text{CID.V}_2(pk_S, \text{com}, \text{ch}, \text{rsp})$
16 **return** d

FSDVS.Sim(pk_S, sk_D, m):

17 $(\text{com}, \text{ch}, \text{rsp}) \leftarrow\!\$$ CID.Sim(pk_S)
18 $r \leftarrow\!\$$ CHF.Inv$(sk_D, \text{ch}, \text{com} \| m)$
19 $s \leftarrow (\text{com}, \text{ch}, \text{rsp})$
20 **return** (r, s)

Figure 9: Designated-verifier signature scheme FSDVS[CID, CHF] constructed from an HVZK and IMP-PA-secure identification protocol CID = (KGen, P, V) and a chameleon hash function CHF = (KGen, Hash, Inv).

Q_T denotes the number of $\mathcal{B}$'s queries to its Transcript oracle which is equal to Q_S, the number of $\mathcal{A}$'s signing queries to Sign. Q_{RO} denotes the number of $\mathcal{A}$'s queries to the random oracle RO, and Q_{InvRO} the number of $\mathcal{A}$'s queries to the random oracle InvRO.

Proof. Assume there exists a $(t', \epsilon', (Q_S + Q_{\text{RO}} + Q_{\text{InvRO}}))$-adversary $\mathcal{A}$ against the EUF-CMA security of the FSDVS[CID, CHF] designated verifier signature scheme. We give a (t, ϵ, Q_T)-adversary $\mathcal{B}$ against the IMP-PA security of the underlying identification scheme CID.

Without loss of generality, assume that $\mathcal{A}$ only asks "signature-relevant" RO queries, i.e., queries of the form (pk, m, r), where pk is a designated verifier's public key, m a message, and $r \in \mathcal{R} \cup \{\varepsilon\}$ the randomness input for hashing.

We can further assume that queries (pk, m, r) with $r \neq \varepsilon$ are not repeatedly asked by $\mathcal{A}$ as the consistent random oracle response yields no new information.

Initialization of $\mathcal{A}$ The IMP-PA adversary $\mathcal{B}$ receives as input a public key, say, pk_{CID}. $\mathcal{B}$ sets $pk_S \leftarrow pk_{\text{CID}}$ and generates a key pair $(pk_D, sk_D) \leftarrow\!\$\ \text{CHF.KGen}()$. The reduction then initializes the EUF-CMA adversary $\mathcal{A}$ on input (pk_S, pk_D).

Queries to RO Let Q_{RO} be the number of queries that $\mathcal{A}$ makes to RO. The reduction $\mathcal{B}$ guesses the query $i^* \in [Q_{\text{RO}}]$ to RO that belongs to the existential signature forgery that $\mathcal{A}$ will output, yielding a loss of a factor Q_{RO}.

Simulation of query i^*: Let (pk, m', r^*) be the i^*-th query of $\mathcal{A}$ to RO. Note that in order to yield a winning signature forgery it must be that $pk = pk_D$ and m' is of the form $\text{com}^* \| m^*$ for some commitment com^* and message m^*. The randomness r^* may be chosen by the adversary or the empty string ε in which case the reduction will sample the randomness $r^* \leftarrow\!\$\ \mathcal{R}_{\text{dist}}$ to be used. $\mathcal{B}$ then submits com^* to its own challenger, receiving a challenge ch^*. $\mathcal{B}$ stores $(pk, m', r^*, \text{ch}^*)$ in $\mathcal{L}_{\text{RO}}$ and returns (r^*, ch^*) to $\mathcal{A}$.

Simulation of queries $j \neq i^*$: For any query (pk, m, r) where $r \neq \varepsilon$, the reduction consults the list of recorded random oracle queries $\mathcal{L}_{\text{RO}}$. If there exists a tuple $(pk, m, r, h) \in \mathcal{L}_{\text{RO}}$, then

the reduction returns (r, h) to the adversary. Note that, in particular, this already ensures consistency with previous InvRO queries (sk, m, h) with response r, in case pk is the public key corresponding to sk. If there is no such value in $\mathcal{L}_{\text{RO}}$, then the reduction samples a digest h at random, stores (pk, m, r, h) in $\mathcal{L}_{\text{RO}}$, and returns (r, h) to the adversary. For queries (pk, m, ε), the reduction samples a value $r \leftarrow\!\!\$\ \mathcal{R}_{\text{dist}}$ and ensures that there is no $(pk, m, r, \cdot)$ entry in $\mathcal{L}_{\text{RO}}$ (the probability of such an entry already existing is negligible). If there is, the reduction simply resamples r until no such list entry is present. It then samples a uniformly random digest h and, after storing (pk, m, r, h) in $\mathcal{L}_{\text{RO}}$, returns (r, h) to the adversary. Note that the probability that there exist (pk, m, r) and (pk, m', r') such that both get assigned the same digest h by this procedure is negligible, due to the random sampling from the digest space; moreover this can be avoided by simply resampling the digest.

Queries to InvRO For any query (sk, h, m), the reduction samples $r \leftarrow\!\!\$\ \mathcal{R}_{\text{dist}}$, computes pk from sk, stores (pk, m, r, h) in $\mathcal{L}_{\text{RO}}$ and returns r to $\mathcal{A}$. Consistency with subsequent RO queries on these values is ensured via the entry in $\mathcal{L}_{\text{RO}}$; no further bookkeeping is required.

Queries to Sign $\mathcal{A}$'s queries to Sign are of the form (pk, m), where pk is the designated verifier's public key and m the message to be signed. For each of the Q_S queries of $\mathcal{A}$, $\mathcal{B}$ uses its Transcript oracle to receive an accepting conversation $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ with respect to signing public key pk_S. In order to ensure the validity of the signature, the reduction samples $r \leftarrow\!\!\$\ \mathcal{R}_{\text{dist}}$ and must ensure that $\mathsf{ch} = \text{RO}(pk, \mathsf{com} \| m, r)$. With high probability, $(pk, \mathsf{com} \| m, r, \cdot)$ has not been set in $\mathcal{L}_{\text{RO}}$ due to an adversary's previous RO or InvRO query; the probability of $\mathcal{A}$ detecting the simulation of the random oracle due to such a collision is upper bounded by $(Q_{\text{RO}} + Q_{\text{InvRO}}) \cdot 2^{-\alpha}$, where α is the min-entropy of commitments in CID. Thus, $\mathcal{A}$ cannot detect the programming of the random oracle by $\mathcal{B}$ on these values.

Existential Forgery At some point, $\mathcal{A}$ outputs a valid DVS forgery $(m^*, \sigma^* = (r^*, (\mathsf{com}^*, \mathsf{ch}^*, \mathsf{rsp}^*)))$ with respect to pk_S, pk_D and corresponding to the i^*-th random oracle query, otherwise $\mathcal{B}$ aborts. Recall that the validity of the forgery implies:

- $m^* \notin \mathcal{L}_{\text{Sign}}$, i.e., m^* has not been queried to the signing oracle for the designated verifier's public key, and thus the RO has not been patched by $\mathcal{B}$ on this value;
- $\mathsf{ch}^* = \text{RO}(pk_D, \mathsf{com}^* \| m^*, r^*)$, i.e., ch^* is the appropriate digest set in the i^*-th query to RO; and
- $\mathsf{V}_2(pk_S, (\mathsf{com}^*, \mathsf{ch}^*, \mathsf{rsp}^*)) = 1$, i.e., $(\mathsf{com}^*, \mathsf{ch}^*, \mathsf{rsp}^*)$ is an accepting conversation under pk_S.

Thus, $\mathcal{B}$ can return rsp as output in its own IMP-PA game and win.

The running time t of $\mathcal{B}$ is dominated by the running time t' of $\mathcal{A}$ and we write $t \approx t'$, omitting the explicit mention of the time it takes to simulate the random oracle, to query the IMP-PA challenger once and to query Transcript a total of $Q_T = Q_S$ times. If $\mathcal{A}$ outputs a successful forgery with probability ϵ', then $\mathcal{B}$ is able to win the IMP-PA game with probability $\epsilon' \leq Q_{\text{RO}} \cdot (\epsilon + Q_S(Q_{\text{RO}} + Q_{\text{InvRO}}) \cdot 2^{-\alpha})$. □

Theorem 3.15 (Source Hiding of FSDVS). *Let* $\mathsf{CID} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ *be a* ϵ_{ZK}*-HVZK and* (t, ϵ)*-IMP-KOA-secure canonical identification protocol. Let* $\mathsf{CHF} = (\mathsf{KGen}, \mathsf{Hash}, \mathsf{Inv})$ *be a* $(t, \epsilon_{\mathsf{CHF}})$*-secure chameleon hash function. Then the Fiat–Shamir transformed signature scheme* $\mathsf{FSDVS}[\mathsf{CID}, \mathsf{CHF}]$ *as defined in Figure 9 is* (t', ϵ', Q_C)*-source hiding according to Definition 3.3, where* $t' \approx t$ *and* $\epsilon' \leq 2\epsilon_{\mathsf{CHF}} + \epsilon_{\mathsf{ZK}}$.

Proof. The proof proceeds similar to the one from Theorem 3.9. We need to show that FSDVS.Sign and FSDVS.Sim output essentially the same distribution of signatures, for fixed public-key/secret-key pairs and any fixed message m.

Let us first consider (r, s) in FSDVS.Sign where s is of the form $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$. By the uniformity property of CHF, we have that ch is ϵ_{CHF}-close to uniformly distributed over $\mathsf{CHF}.\mathcal{D}$ (which is equal to the challenge set of the canonical identification protocol), with r sampled from the distribution $\mathsf{CHF}.\mathcal{R}_{\text{dist}}$ conditioned on $\mathsf{ch} = \mathsf{CHF.Hash}(pk_D, m; r)$. Since the canonical identification protocol is honestly executed this results in generating a commitment com and response rsp such that the overall transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$ will correctly verify.

Now consider (r, s) in FSDVS.Sim. By the honest-verifier zero-knowledge property of CID, we have that the simulated transcript is ϵ_{ZK}-close distributed to real transcripts. By the chameleon indistinguishability property of CHF, the call to CHF.Inv on line 18 of Figure 9 outputs r that is ϵ_{CHF}-close to the distribution $r \leftarrow\!\!\$\ \mathsf{CHF}.\mathcal{R}_{\text{dist}}$ conditioned on $\mathsf{CHF.Hash}(pk_D, m; r) = \mathsf{ch}$.

Hence, (r, s) in FSDVS.Sign and FSDVS.Sim are $(2\epsilon_{\mathsf{CHF}} + \epsilon_{\mathsf{ZK}})$-close, so FSDVS is source hiding as required. □

3.5 Related Work: Lattice-based DVS

There are several lattice-based DVS schemes in the literature. In this section we review this related work and highlight the differences to our two constructions.

Wang, Hu, and Wang [76] construct a DVS scheme directly from lattice assumptions (LWE and SIS) by combining the Bonsai tree lattice trapdoor of [18] with the GPV lattice-based signature scheme [44]; a subsequent paper of theirs [77] extends this to the identity-based setting. While [76] does use ingredients from the same two papers that our first construction relies on ([44] and [18]), their construction is different as it uses the Bonsai lattice trapdoor directly in the arithmetic of the signing and verifying operations, rather than in the hashing as we do with the chameleon hash function. [76] also requires the designated verifier to use their secret key in the verification algorithm (i.e., they require a *strong* DVS). Finally, our construction is generic and can be instantiated by any preimage sampleable trapdoor function and chameleon hash function (as long as the digest space of the CHF matches the range of the PSF), whereas theirs is written for specific lattice operations.

Noh and Jeong [64] improve on [76, 77] 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. Whereas [64] uses a CHF as a standard-model replacement of a programmable random oracle for maintaining consistency of their simulation in the proof of unforgeability, we use the CHF for the source hiding property of the DVS.

Li, Liu, and Yang [57] construct a DVS scheme directly from ideal lattice assumptions (ring-SIS) by combining a ring version of the GPV signature scheme [61] with a ring chameleon hash function [34] and adding a Fiat–Shamir-with-aborts technique [59, 60]. The use of the chameleon hash function in the LLY scheme is closest to ours, in the sense that it permits the hash digest to be inverted by the designated verifier for simulating signatures. However, the exact way the CHF is used is different, as is their overall construction, and their formulation is non-generic.

Zhang, Loiu, Tang, and Tian also give a DVS constructed directly from SIS by adapting the Lyubashevsky signature scheme [60].

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 [17, 52]. 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, indicating cryptographically relevant material for key derivation.
- $\mathsf{K} \in \mathcal{K}_{\mathsf{KE}} \cup \{\bot\}$: 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 other 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:

- $\mathsf{Fake}(pk_U, sk_V) \xrightarrow{\$}$ transcript: A probabilistic transcript simulation algorithm that takes as input one party's public key and the other party's secret key and generates a transcript 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)[3] 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 ($\pi_U^i.\mathsf{sid} = \pi_V^j.\mathsf{sid}$).

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 Test query has been called before or session π_U^i has not accepted, then return $\bot$. 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

[3] 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., [22, 73, 74]).

(a) there exists an honest partner session π^*_{p} (π^*_{p}.sid = π^*.sid if π^* is a responder, and π^*_{p}.cid = π^*.cid if π^* is an initiator), covering weak forward secrecy, or
(b) CorruptLTKey(π^*.oid) and CorruptLTKey(π^*.pid) was never called, covering implicit authentication.

At the end of the experiment, the adversary outputs a bit b'. The adversary is said to win if $b' = b_{\text{test}}$ and the test session π^* is fresh. Formally, if the test session is fresh, the experiment outputs 1 if $b' = b_{\text{test}}$ 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 Chall oracle which takes as input two party identifiers U and V. If b is 0, then 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 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 [24, 25, 30, 73, 74]. Our definition differs from previous ones in that it gives access to secret keys to the Fake algorithm (corresponding to the simulator in simulation-based definitions) and to the adversary (i.e., the judge). This models the informal deniability requirement from the Signal specification [63, §4.4]. See Appendix A for a more detailed discussion.

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 in IND-CCA security of the two KEMs, unforgeability of the DVS, and security of the KDF; deniability of Π depends on 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 initiator role, Bob the 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:

$$\begin{aligned}\text{transcript} = \text{sid} &= (A, B, pk_A^{\mathsf{DVS}}, pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}, epk_B^{\mathsf{KEM}}, c_1, c_2)\\ \text{cid} &= (B, pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}, epk_B^{\mathsf{KEM}})\end{aligned}$$

5.1 Key Indistinguishability

Theorem 5.1 (Key indistinguishability of Π). *Let* DVS *be a* $(t, \epsilon_{\mathsf{DVS}}, Q_S)$*–*EUF-CMA*-secure DVS scheme,* KEM_1 *be a* $(t, \epsilon_{\mathsf{KEM}_1}, n_s)$*–*IND-CCA*-secure KEM,* KEM_2 *be a* $(t, \epsilon_{\mathsf{KEM}_2}, 1)$*–*IND-CCA*-secure KEM, and* KDF *be a* $(t, \epsilon_{\mathsf{KDF}}, n_s)$*–*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*

$$\epsilon' \leq n_s \cdot \left(\begin{array}{l} n_s \cdot \left(\epsilon_{\mathsf{KEM}_2} + \epsilon_{\mathsf{KDF}}\right) \\ + n_p \cdot \left(\epsilon_{\mathsf{KEM}_1} + \epsilon_{\mathsf{KDF}}\right) \\ + n_p^2 \cdot \left(\epsilon_{\mathsf{DVS}} + n_s \cdot \left(\epsilon_{\mathsf{KEM}_2} + \epsilon_{\mathsf{KDF}}\right)\right) \end{array} \right),$$

where $n_s \leq Q_{Snd}$ *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, denoted $\mathcal{G}_0$, letting $\epsilon' := \mathsf{Adv}^{\mathcal{G}_0}_{\mathsf{ADAKE}}(\mathcal{A}) = |\Pr[\mathcal{G}_0 = 1] - \frac{1}{2}|$.

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),

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

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} (π^*_{p}.sid = π^*.sid if π^* is a responder, and π^*_{p}.cid = π^*.cid if π^* is an initiator).
B. The tested session is an initiator ("Alice") session and CorruptLTKey(π^*.pid) was never called.
C. The tested session is a responder ("Bob") session and neither CorruptLTKey(π^*.oid) nor CorruptLTKey(π^*.pid) was ever called.[4]

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:

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

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)

[4] 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 [7] against Bob, as in the Signal handshake [63, §4.6].

one partner session exists and is guessed with probability at least $1/n_s$, hence

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

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 IND-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 IND-CCA KEM challenge bit, the reduction perfectly simulates $\mathcal{G}_{A.1}$ or $\mathcal{G}_{A.2}$, hence

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

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

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

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 $\mathsf{Adv}^{\mathcal{G}_{A.3}}_{\mathsf{ADAKE}}(\mathcal{A}) = 0$.

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:

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

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 IND-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^{KEM}_V 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 IND-CCA KEM challenge bit, the reduction perfectly simulates $\mathcal{G}_{B.1}$ or $\mathcal{G}_{B.2}$, hence

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

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 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

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

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

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:

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

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 existential unforgeability of DVS, running in time $t \approx t'$ and simulating $\mathcal{G}_{C.1}$

except that it embeds the EUF-CMA game's challenge public keys as $pk_U = pk_S$ and $pk_V = pk_D$. 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 (transcript, σ) that no session of U sent (and hence transcript was not queried to the DVS Sign oracle), the reduction outputs this pair as its forgery and wins. Therefore,

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

Game C.3 (Guess partnered session). As of $\mathcal{G}_{C.2}$, we know that π^* receives a DVS signature on a transcript value transcript = π^*.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

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

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 IND-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 IND-CPA security of KEM_2 here.) The reduction simulates the difference between $\mathcal{G}_{C.3}$ and $\mathcal{G}_{C.4}$, so

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

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, transcript). Due to π^* and π^*_{p} agreeing on the 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

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

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 $\mathsf{Adv}^{\mathcal{G}_{C.5}}_{\mathsf{ADAKE}}(\mathcal{A}) = 0$. □

5.2 Deniability

Theorem 5.2 (Deniability of Π). *Let* DVS = (SKGen, VKGen, Sign, Vrfy, Sim) *be a* $(t, \epsilon_{\mathsf{srchid}}, Q_C)$*-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_C *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 attacker 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 iid, 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 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 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 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 an 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 10.

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 [13]. We capture this similarly in SPQR by encapsulating a

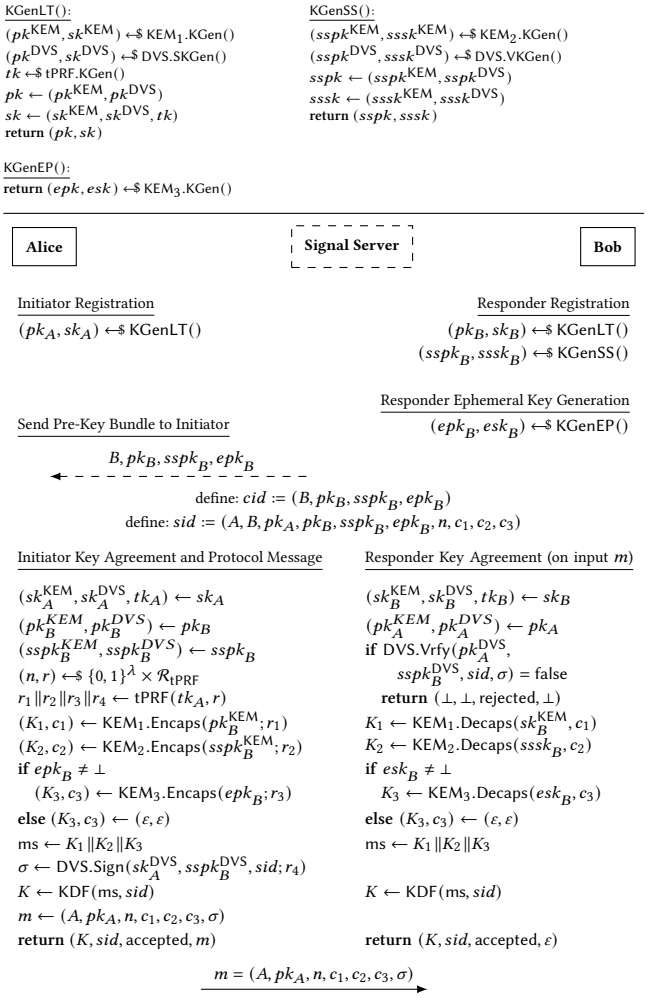

Figure 10: The SPQR **protocol (top: key generation, middle: protocol flow, bottom: fake transcript generation), combining static, semi-static and ephemeral key encapsulation schemes** KEM_1**,** KEM_2**, and** KEM_3**, a designated verifier signature** DVS**, and a twisted pseudorandom function** tPRF**.**

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) [16, 22, 56]. 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 [56] trick to extract randomness from Alice's long-term secrets via a twisted PRF [41, 54] (generically instantiable from regular PRFs, see Section 2.3).

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 ADAKE model that we will use to prove our post-quantum Signal construction SPQR depicted in Figure 10 secure. The key-indistinguishability game is fully specified in Figure 11 and the deniability game in Figure 12. 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 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 CorruptSSKey(U, ssid) oracle, similar to the corruption of long-term keys via 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 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 RevealRandom(U, i) oracle that returns the coins sampled in session π_U^i, and marks them as revealed via a flag revrand.

7.1 Key Indistinguishability

Definition 7.1. An asynchronous DAKE key exchange protocol ADAKE 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

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

where $\mathcal{G}^{\text{ke-kind}}_{\mathsf{ADAKE}}(\mathcal{A})$ is defined in Figure 11 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:

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

In order to fully describe the security game $\mathcal{G}_{\mathsf{KE}}^{\text{ke-kind}}(\mathcal{A})$ 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:

- $\text{corrltk}_U \in$ {true, 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.
- $\text{corrssk}_U^{\text{ssid}} \in$ {true, 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.

7.1.1 Session Partnering and Correctness. As in the core model, (full) session partnering is defined via session identifiers: We say that a session π_U^i owned by U is *partnered* with a session π_V^j owned by V if they agree on the session identifier, i.e., $\pi_U^i.\text{sid} = \pi_V^j.\text{sid} \neq \perp$. 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_U^i owned by U is *contributively partnered* with a session π_V^j owned by V, if they agree on their contributive session identifier, i.e., whenever $\pi_U^i.\text{cid} = \pi_V^j.\text{cid} \neq \perp$.

We say that an asynchronous authenticated key exchange protocol KE = (KGenLT, KGenSS, KGenEP, Run) 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.

7.1.2 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. reduced), or do not agree on their contributive identifiers (Fig. 11, Line 18).
(ii) Two initiator sessions accept with the same session identifier (Fig. 11, Line 19).
(iii) Three sessions accept with the same session identifier in full handshake type (Fig. 11, Line 20).

7.1.3 Freshness. Granting the adversary $\mathcal{A}$ access to the oracles described in Figure 11 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 11) 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. 11, 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. 11, 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. 11, Line 16).

Clean keys. Following the terminology of Cohn-Gordon, Cremers, Dowling, Garratt, and Stebila [22] 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 11. 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 $\text{clean}_{\text{full}}$ or the $\text{clean}_{\text{reduced}}$ predicate to π^*.

Since $\text{clean}_{\text{reduced}}$ is part of the description of $\text{clean}_{\text{full}}$, we first assume that π^*.type = 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^*:

$\text{clean}_{\text{LTSS}}$: 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.

$\text{clean}_{\text{ELT}}$: This predicate indicates whether the combination of the ephemeral contribution[5] of the initiator and the long-term key of the responder is unknown to the adversary.

$\text{clean}_{\text{ESS}}$: 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 $\text{clean}_{\text{ELT}}$ and $\text{clean}_{\text{ESS}}$ necessitates a further predicate called $\text{clean}_{\text{peerE}}$ (in all other cases, it is sufficient to consider the flags corrltk, corrssk, and revrand, respectively). For responder test sessions, $\text{clean}_{\text{peerE}}$ 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 π^*.type = full, it must either hold that $\text{clean}_{\text{reduced}}$ is true or that the additional input to the master secret computation is clean. The latter is captured by the following predicate:

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

$\underline{\mathcal{G}^{\text{ke-kind}}_{\mathsf{KE}}(\mathcal{A})}$:

$b_{\text{test}} \leftarrow\$ \{0,1\}$ //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 ssid $\in [n_{ss}]$
    $(sspk_U^{\text{ssid}}, sssk_U^{\text{ssid}}) \leftarrow\$ \mathsf{KGenSS}()$ //semi-static key generation
$\vec{pk} \leftarrow \{pk_U\}_{U\in[n_p]}$; $\vec{sspk} \leftarrow \{sspk_U^{\text{ssid}}\}_{U\in[n_p]}^{\text{ssid}\in[n_{ss}]}$
$b' \leftarrow\$ \mathcal{A}^{\textsc{Send},\textsc{Test},\textsc{CorruptLTKey},\textsc{CorruptSSKey},\textsc{RevealRandom},\textsc{RevealSessKey}}(\vec{pk}, \vec{sspk})$ //run adversary
if sound() = false //adversary wins if it breaks soundness
  return 1
if fresh(π^*) = false //attack invalid if test session is not fresh
  $b' \leftarrow 0$
return $[\![b' = b_{\text{test}}]\!]$ //determine win or loss

$\underline{\text{fresh}(\pi^*)}$:

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

$\underline{\text{sound}()}$:

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

$\underline{\textsc{Send}(U, i, m)}$:

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

$\underline{\textsc{Test}(U, i)}$:

if $\pi_U^i = \perp$ **or** $\pi_U^i.\text{st}_{\text{exec}} \neq$ 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 $b_{\text{test}} = 0$
  $K_{\text{test}} \leftarrow \pi_U^i.\text{K}$ //real session key
else
  $K_{\text{test}} \leftarrow\$ \mathcal{K}_{\mathsf{KE}}$ //random key from key space
return K_{test} //return challenge key

$\underline{\textsc{CorruptLTKey}(U)}$:

$\text{corrltk}_U \leftarrow$ true //mark long-term key as corrupted
return sk_U //return long-term secret key

$\underline{\textsc{CorruptSSKey}(U, \text{ssid})}$:

$\text{corrssk}_U^{\text{ssid}} \leftarrow$ true //mark semi-static key as corrupted
return $sssk_U^{\text{ssid}}$ //return semi-static secret key

$\underline{\textsc{RevealRandom}(U, i)}$:

if $\pi_U^i = \perp$ **then return** $\perp$ //session does not exist
$\pi_U^i.\text{revrand} \leftarrow$ true //mark randomness as revealed
return $\pi_U^i.\text{coins}$ //return session's random coins

$\underline{\textsc{RevealSessKey}(U, i)}$:

if $\pi_U^i = \perp$ **or** $\pi_U^i.\text{st}_{\text{exec}} \neq$ accepted **then return** $\perp$ //session does not exist or has not derived key yet
$\pi_U^i.\text{revealed} \leftarrow$ true //mark session key as revealed
return $\pi_U^i.\text{K}$ //return session key

$\underline{\text{clean}_{\text{full}}(\pi^*)}$:

return $\text{clean}_{\text{reduced}}(\pi^*)$ **or** $\text{clean}_{\text{EE}}(\pi^*)$

$\underline{\text{clean}_{\text{reduced}}(\pi^*)}$:

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

$\underline{\text{clean}_{\text{EE}}(\pi^*)}$:

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

$\underline{\text{clean}_{\text{peerE}}(\pi^*)}$:

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

$\underline{\text{clean}_{\text{LTSS}}(\pi^*)}$:

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

$\underline{\text{clean}_{\text{ELT}}(\pi^*)}$:

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

$\underline{\text{clean}_{\text{ESS}}(\pi^*)}$:

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

Figure 11: Key indistinguishability game $\mathcal{G}^{\text{ke-kind}}_{\mathsf{KE}}(\mathcal{A})$ for key exchange protocol KE against adversary $\mathcal{A}$; 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.

$\mathsf{clean}_{\mathsf{EE}}$: 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.

7.1.4 Main differences to the model in [22]. Our authenticated key exchange model closely follows the one used in the original Signal analysis of Cohn-Gordon et al. [22]. 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.

7.2 Deniability

Definition 7.2. An asynchronous DAKE protocol ADAKE is (t, ϵ, Q_C)*-deniable* if for any adversary $\mathcal{A}$ with running time at most t and making at most Q_C many queries to its Chall oracle, we have that

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

where $\mathcal{G}^{\text{adake-den}}_{\mathsf{ADAKE}}(\mathcal{A})$ is defined in Figure 12.

The main difference between the textual description of the deniability game in Section 5.2 and Figure 12 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 attacker. When querying the challenge oracle, the attacker 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 ADAKE.

$\mathcal{G}^{\text{adake-den}}_{\mathsf{ADAKE}}(\mathcal{A})$:

$\mathcal{L} \leftarrow \emptyset$ //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 ssid $\in [n_{ss}]$
    $(sspk_U^{\text{ssid}}, sssk_U^{\text{ssid}}) \leftarrow\$ \mathsf{KGenSS}()$ //semi-static key generation
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(sspk_U^{\text{ssid}}, sssk_U^{\text{ssid}})\}$
$\vec{pk} \leftarrow \{pk_U\}_{U \in [n_p]}$; $\vec{sspk} \leftarrow \{sspk_U^{\text{ssid}}\}_{U \in [n_p]}^{\text{ssid} \in [n_{ss}]}$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{A}^{\textsc{Chall}}(\mathcal{L})$
return $[\![b' = b]\!]$

Chall(iid, rid, ssid):

if $b = 0$
  $\pi_{\text{rid}}.\mathsf{role} \leftarrow \text{responder}$; $\pi_{\text{rid}}.\mathsf{st}_{\text{exec}} \leftarrow \text{running}$ //initialize session variables
  $\pi_{\text{iid}}.\mathsf{role} \leftarrow \text{initiator}$; $\pi_{\text{iid}}.\mathsf{st}_{\text{exec}} \leftarrow \text{running}$
  $(\pi_{\text{rid}}, m) \leftarrow\$ \mathsf{Run}(sk_{\text{rid}}, \vec{pk}, \vec{sspk}, \pi_{\text{rid}}, (\text{create}, \text{ssid}))$ //build pre-key bundle
  $(\pi_{\text{iid}}, m') \leftarrow\$ \mathsf{Run}(sk_{\text{iid}}, \vec{pk}, \vec{sspk}, \pi_{\text{iid}}, m)$ //initiator sends message
  $(K, \text{transcript}) \leftarrow (\pi_{\text{iid}}.K, (m, m'))$ //save session key and transcript
else
  $(K, \text{transcript}) \leftarrow\$ \mathsf{Fake}(pk_{\text{iid}}, sk_{\text{rid}}, \vec{sspk}, \text{ssid})$
return $(K, \text{transcript})$

Figure 12: Security game for deniability of an asynchronous DAKE protocol ADAKE against an adversary $\mathcal{A}$.

8 SPQR SECURITY PROOFS

In this section we present the security results for our SPQR protocol (Figure 10) 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 10 into the syntax of our model. The resulting protocol flow is depicted in Figure 13.

8.1 Key Indistinguishability

Theorem 8.1 (Key indistinguishability of SPQR). *Let* DVS *be a* $(t, \epsilon_{\mathsf{DVS}}, Q_S)$–EUF-CMA*-secure DVS scheme.*

Let KEM_1 *be a* $(t, \epsilon_{\mathsf{KEM}_1}, n_s)$–IND-CCA*-secure KEM,* KEM_2 *be a* $(t, \epsilon_{\mathsf{KEM}_2}, n_s)$–IND-CCA*-secure KEM,* KEM_3 *be a* $(t, \epsilon_{\mathsf{KEM}_3}, 1)$–IND-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* $(t, \epsilon_{\mathsf{KDF}}, n_s)$–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 10 and formalized in Figure 13 provides* $(t', \epsilon', (Q_{Snd}, Q_{CorrLT}, Q_{CorrSS}, Q_{RevR}, Q_{RevSK}))$*–key indistinguishability 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* Send *queries),* n_p *the number of parties, and* n_{ss} *the number of semi-static keys per party.*

Proof. We proceed via a sequence of game hops starting from $\mathcal{G}^{\text{ke-kind}}_{\mathsf{SPQR}}(\mathcal{A})$ (cf. Figure 11), 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 $\mathcal{G}^{\text{ke-kind}}_{\mathsf{SPQR}}(\mathcal{A})$ for SPQR played by $\mathcal{A}$. By definition,

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

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

Figure 13: 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 $\mathcal{G}_{\mathsf{KE}}^{\text{ke-kind}}(\mathcal{A})$ and the Signal Server is abstracted away. Thus these elements are not included explicitly in the above description.**

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 11, 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 type = 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 ($epk_B = \perp$ if and only if type = 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 11, 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 11, 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:

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

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:

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

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:

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

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 $(t, \epsilon_{\mathsf{DVS}}, Q_S)$-existential unforgeability of DVS.

Reduction $\mathcal{B}_1$ samples all key components itself except for the long-term DVS key $(pk_U^{\mathsf{DVS}}, sk_U^{\mathsf{DVS}})$ of U and the semi-static DVS key $(sspk_V^{\mathsf{DVS}}, sssk_V^{\mathsf{DVS}})$ of V, for which instead it uses the public keys pk_S and pk_D, respectively, obtained in its EUF-CMA game. 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 a CorruptLTKey(U) or a CorruptSSKey(V, ssid) query. 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,

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

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 randomness). 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.[6]

We can now bound $\mathcal{A}$'s difference in advantage by the advantage of a reduction $\mathcal{B}_3$ in winning the $(t, \epsilon_{\mathsf{KEM}_2}, n_s)$-IND-CCA security game for KEM_2. The reduction $\mathcal{B}_3$ obtains the IND-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 CorruptSSKey(V, 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 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 IND-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 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

[6] 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.

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 $\text{fresh}(\pi^*)$).

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 $[\![b_{\text{test}} = b']\!]$.

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

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

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 who 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:

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

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 $\mathsf{clean}_{\mathsf{ELT}}(\pi^*)$:

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

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:

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

Game B.2 (Twisted PRF randomness). 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 π^*.role = initiator, and $\pi = \pi^*_{\mathrm{p}}$, if π^*.role = 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:

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

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 π^*.role = initiator, and $\pi = \pi^*_{\mathrm{p}}$, if π^*.role = 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 $(t, \epsilon_{\mathsf{KEM}_1}, n_s)$-IND-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 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 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 $[\![b_{\text{test}} = b']\!]$.

The simulation $\mathcal{B}_6$ provides for $\mathcal{A}$ perfectly represents Game $\mathcal{G}_{B.2}$ if $b = 0$ in the IND-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 IND-CCA game against KEM_1:

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

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:

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

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:

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

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 $\mathsf{clean}_{\mathsf{ESS}}(\pi^*)$:

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

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 π^*.role = responder) or that of its intended peer (if π^*.role = 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 randomness). 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 π^*.role = initiator, and $\pi = \pi^*_{\mathsf{p}}$, if π^*.role = 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 π^*.role = initiator, and $\pi = \pi^*_{\mathsf{p}}$, if π^*.role = 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 $(t, \epsilon_{\mathsf{KEM}_2}, n_s)$-IND-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 $b' = b_{\text{test}}$ and 1 otherwise.

Observe that $\mathcal{B}_9$ perfectly simulates $\mathcal{G}_{C.3}$ if $b = 0$ in $\mathcal{G}^{\text{indcca}}_{\mathsf{KEM}_2}(\mathcal{B}_9)$ 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:

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

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 $(t, \epsilon_{\mathsf{KDF}}, n_s)$-PRFSEC property of KDF, this time keyed via K_2:

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

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 $b_{\text{test}} = 0$ or a uniformly random string if $b_{\text{test}} = 1$. 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

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

Case D (π^*.type = full and $\mathsf{clean}_{\mathsf{EE}}(\pi^*)$). 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:

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

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 π^*.type = 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:

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

Game D.2 (Twisted PRF randomness). 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 π^*.role = initiator, and $\pi = \pi^*_{\text{p}}$, if π^*.role = 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 $(n, r) \leftarrow \pi.\text{coins}$, $\mathcal{B}_{11}$ sets $r_1 \| r_2 \| r_3 \| r_4 \leftarrow z$; as in prior cases, $\mathcal{B}_{11}$ can still answer a potential $\textsc{CorruptLTKey}(U)$ query, but r remains hidden since $\textsc{RevealRandom}(\pi)$ 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:

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

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 π^*.role = initiator, and $\pi = \pi^*_{\text{p}}$, if π^*.role = 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 $(t, \epsilon_{\mathsf{KEM}_3}, 1)$-IND-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 $\textsc{RevealRandom}(\tilde{\pi})$ query, where $\tilde{\pi} = \pi^*$ if the π^* is the responder, and $\tilde{\pi} = \pi^*_{\text{p}}$, 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 $\textsc{Decaps}$ 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 $b' = b_{\text{test}}$ and 1 otherwise.

Observe that $\mathcal{B}_{12}$ perfectly simulates $\mathcal{G}_{D.2}$ if $b = 0$ in $\mathcal{G}^{\text{indcca}}_{\mathsf{KEM}_3}(\mathcal{B}_{12})$ 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:

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

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 8.2 (Deniability of SPQR). *If* DVS *is a* $(t, \epsilon_{\mathsf{srchid}}, Q_C)$*-source hiding designated verifier signature, then the* SPQR *protocol as shown in Figure 10 is* (t', ϵ', Q'_C)*-deniable, where* $t' \approx t$, $\epsilon' \leq n_p^2 n_{ss} \cdot \epsilon_{\mathsf{srchid}}$*, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party, and* $Q'_C = Q_C$.

Proof. An attacker $\mathcal{B}$ against the source hiding property of DVS can use a successful attacker $\mathcal{A}$ against deniability of SPQR[KEM_1, KEM_2, KEM_3, DVS, KDF, tPRF] to succeed in its own game. The challenger starts $\mathcal{B}$ with two DVS key pairs who then simulates the asynchronous DAKE key exchange deniability game $\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A})$ for $\mathcal{A}$ as follows. For each of the n_p parties $\mathcal{B}$ 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 attacker can distinguish between Run and Fake. Let a number $i \in [n_p^2 n_{ss}]$ uniquely denote three independent values iid, rid, 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^*, $\mathcal{B}$ 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^*, $\mathcal{B}$ 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.

$\mathcal{B}$ answers the queries of $\mathcal{A}$ to the Chall oracle as follows: First, it runs the responder ephemeral key generation. Then, it runs the initiator key agreement until computing the DVS signature (i.e., it computes a nonce and the three KEM ciphertexts (n, c_1, c_2, c_3) and 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 $\mathcal{B}$ distinguishes between three cases: The first case is that the query is for $1 \leq i < i^*$. Then the reduction behaves as if $b = 0$, i.e., it executes DVS.Sign. 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 behaves as if $b = 1$, i.e., it executes DVS.Sim. 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 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', $\mathcal{B}$ 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 let us analyze the winning probability of $\mathcal{A}$ against deniability. 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 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 and H_0 which answers all queries by Fake. Observe that H_{i-1} and H_i only differ in an execution of Run or Fake depending on the reduction $\mathcal{B}$'s challenge oracle. Hence, it is easy to see that the probability of distinguishing between H_{i-1} and H_i is bounded by $\epsilon_{\mathsf{srchid}}$.

Now let us analyze $\mathcal{A}$'s advantage in more detail and we denote by $H_i^{\mathcal{A}}$ the output of the adversary in the ith hybrid. Then it follows:

$$\begin{aligned}
&\mathsf{Adv}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) \\
&= \left| \Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 1\right] - \frac{1}{2} \right| \\
&= \left| \Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 1 | b = 1\right] \cdot \Pr[b = 1] \right. \\
&\qquad \left. + \Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 1 | b = 0\right] \cdot \Pr[b = 0] - \frac{1}{2} \right| \\
&= \left| \frac{1}{2} \cdot \left(\Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 1 | b = 1\right] \right. \right. \\
&\qquad \left. \left. + \Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 1 | b = 0\right] \right) - \frac{1}{2} \right| \\
&= \left| \frac{1}{2} \cdot \left(\Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 1 | b = 1\right] \right. \right. \\
&\qquad \left. \left. + 1 - \Pr\left[\mathcal{G}^{\text{adake-den}}_{\mathsf{SPQR}}(\mathcal{A}) = 0 | b = 0\right] \right) - \frac{1}{2} \right| \\
&= \left| \frac{1}{2} \cdot \left(\Pr[1 \leftarrow\!\!\$\, \mathcal{A} | b = 1] - \Pr[1 \leftarrow\!\!\$\, \mathcal{A} | b = 0] \right) \right| \\
&= \left| \frac{1}{2} \cdot \left(\Pr\left[H_0^{\mathcal{A}} = 1\right] - \Pr\left[H_{n_p^2 n_{ss}}^{\mathcal{A}} = 1\right] \right) \right| \\
&\leq \frac{1}{2} \cdot \left(\left| \Pr\left[H_0^{\mathcal{A}} = 1\right] - \Pr\left[H_1^{\mathcal{A}} = 1\right] \right| + \ldots \right. \\
&\qquad \left. + \left| \Pr\left[H_{n_p^2 n_{ss}-1}^{\mathcal{A}} = 1\right] - \Pr\left[H_{n_p^2 n_{ss}}^{\mathcal{A}} = 1\right] \right| \right) \\
&= \frac{1}{2} \cdot \sum_{i=1}^{n_p^2 n_{ss}} \left| \Pr\left[H_{i-1}^{\mathcal{A}} = 1\right] - \Pr\left[H_i^{\mathcal{A}} = 1\right] \right|
\end{aligned}$$

$$= \sum_{i=1}^{n_p^2 n_{ss}} \frac{1}{2} \cdot |\Pr[1 \leftarrow\!\$\ \mathcal{B} | b_{\text{srchid}} = 1] - \Pr[1 \leftarrow\!\$\ \mathcal{B} | b_{\text{srchid}} = 0]|$$

$$= \sum_{i=1}^{n_p^2 n_{ss}} \Big| \Pr[1 \leftarrow\!\$\ \mathcal{B} | b_{\text{srchid}} = 1] \cdot \Pr[b_{\text{srchid}} = 1] + \Pr[0 \leftarrow\!\$\ \mathcal{B} | b_{\text{srchid}} = 0] \cdot \Pr[b_{\text{srchid}} = 0] - \frac{1}{2} \Big|$$

$$= \sum_{i=1}^{n_p^2 n_{ss}} \left| \Pr\left[\mathcal{G}_{\text{DVS}}^{\text{srchid}}(\mathcal{B}) = 1 - \frac{1}{2} \right] \right|$$

$$= \sum_{i=1}^{n_p^2 n_{ss}} \text{Adv}_{\text{DVS}}^{\text{srchid}}(\mathcal{B}) = n_p^2 n_{ss} \cdot \text{Adv}_{\text{DVS}}^{\text{srchid}}(\mathcal{B})$$

Hence by this analysis, it follows that $\mathcal{A}$'s probability of winning the deniability game is bounded by $\epsilon' \leq n_p^2 n_{ss} \cdot \epsilon_{\text{srchid}}$. □

9 DISCUSSION AND LIMITATIONS

Our protocols demonstrate that designated verifier signatures are helpful for constructing practical authenticated key exchange 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 3.5, we believe that their security merits further scrutiny before adoption. Regarding our two DVS constructions (GPVDVS and FSDVS), one limitation is that our proofs are in the classical random oracle model, whereas it would be preferable for post-quantum schemes to have security proofs that consider random oracle queries in quantum superposition [8]. While the signature scheme components of our constructions (GPV and FS) have many well-studied practical realizations including some Round 3 candidates in the NIST PQC standardization process, instantiations of chameleon hash functions are much less common. Another limitation of our DVS constructions is that the digest space of the chameleon hash function must match some input space in another function, which may require some care to achieve since chameleon hash functions tend to have structured output spaces rather than opaque bitstrings. We give an estimate of security parameters for instantiating the GPVDVS scheme; note that the formulas that our analysis of the CHF parameters is based on [18, §4.1],[34, §B.3] have asymptotic terms for which the hidden constants are not yet worked out. This may affect the parameters somewhat and should be resolved before adopted in practice.

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 [22].

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 [1, 6, 67].

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 disclosure of randomness, but not use of malicious randomness. This has been considered 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. One should be careful with deniability as a cryptographic property. How cryptographers understand deniability may be different from how a judge in the legal system would understand it [73]. Additionally there are stronger notions of deniability [33] 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 also confirm deniability at all levels of the protocol, and that deniability of individual components composes appropriately. Despite all these subtleties, steps toward deniability are helpful, as Unger and Goldberg write [73]: "we should strive to design deniable protocols to avoid unintentionally incriminating users."

Acknowledgements

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] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. 2019. The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol. In *EUROCRYPT 2019, Part I (LNCS)*, Yuval Ishai and Vincent Rijmen (Eds.), Vol. 11476. Springer, Heidelberg, 129–158. https://doi.org/10.1007/978-3-030-17653-2_5

[2] Reza Azarderakhsh, David Jao, and Christopher Leonardi. 2017. Post-Quantum Static-Static Key Agreement Using Multiple Protocol Instances. In *SAC 2017 (LNCS)*, Carlisle Adams and Jan Camenisch (Eds.), Vol. 10719. Springer, Heidelberg, 45–63. https://doi.org/10.1007/978-3-319-72565-9_3

[3] Mihir Bellare and Phillip Rogaway. 1994. Entity Authentication and Key Distribution. In *CRYPTO'93 (LNCS)*, Douglas R. Stinson (Ed.), Vol. 773. Springer, Heidelberg, 232–249. https://doi.org/10.1007/3-540-48329-2_21

[4] Mihir Bellare and Phillip Rogaway. 1995. Optimal Asymmetric Encryption. In *EUROCRYPT'94 (LNCS)*, Alfredo De Santis (Ed.), Vol. 950. Springer, Heidelberg, 92–111. https://doi.org/10.1007/BFb0053428

[5] Mihir Bellare and Phillip Rogaway. 1996. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *EUROCRYPT'96 (LNCS)*, Ueli M. Maurer (Ed.), Vol. 1070. Springer, Heidelberg, 399–416. https://doi.org/10.1007/3-540-68339-9_34

[6] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. 2017. Ratcheted Encryption and Key Exchange: The Security of Messaging. In *CRYPTO 2017, Part III (LNCS)*, Jonathan Katz and Hovav Shacham (Eds.), Vol. 10403. Springer, Heidelberg, 619–650. https://doi.org/10.1007/978-3-319-63697-9_21

[7] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. 1997. Key Agreement Protocols and Their Security Analysis. In *6th IMA International Conference on*

Cryptography and Coding (LNCS), Michael Darnell (Ed.), Vol. 1355. Springer, Heidelberg, 30–45.

[8] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random Oracles in a Quantum World. In *ASIACRYPT 2011 (LNCS)*, Dong Hoon Lee and Xiaoyun Wang (Eds.), Vol. 7073. Springer, Heidelberg, 41–69. https://doi.org/10.1007/978-3-642-25385-0_3

[9] Dan Boneh, Darren Glass, Daniel Krashen, Kristin Lauter, Shahed Sharif, Alice Silverberg, Mehdi Tibouchi, and Mark Zhandry. 2020. Multiparty Non-Interactive Key Exchange and More From Isogenies on Elliptic Curves. *Journal of Mathematical Cryptology* 14, 1 (2020), 5–14.

[10] Xavier Bonnetain and André Schrottenloher. 2020. Quantum Security Analysis of CSIDH. In *EUROCRYPT 2020, Part II (LNCS)*, Anne Canteaut and Yuval Ishai (Eds.), Vol. 12106. Springer, Heidelberg, 493–522. https://doi.org/10.1007/978-3-030-45724-2_17

[11] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. 2018. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*. IEEE, 353–367. https://cryptojedi.org/papers/#kyber.

[12] Colin Boyd, Yvonne Cliff, Juan Manuel Gonzalez Nieto, and Kenneth G. Paterson. 2009. One-round key exchange in the standard model. *IJACT* 1 (2009), 181–199. Issue 3.

[13] Colin Boyd and Kai Gellert. 2020. A Modern View on Forward Security. *Comput. J.* 64, 4 (08 2020), 639–652. https://doi.org/10.1093/comjnl/bxaa104 arXiv:https://academic.oup.com/comjnl/article-pdf/64/4/639/37161647/bxaa104.pdf

[14] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. 2020. Towards Post-Quantum Security for Signal's X3DH Handshake. In *27th Conference on Selected Areas in Cryptography (SAC)*. Springer.

[15] Jie Cai, Han Jiang, Pingyuan Zhang, Zhihua Zheng, Hao Wang, Guangshi Lü, and Qiuliang Xu. 2019. ID-Based Strong Designated Verifier Signature over $\mathcal{R}$-SIS Assumption. *Secur. Commun. Networks* 2019 (2019), 9678095:1–9678095:8. https://doi.org/10.1155/2019/9678095

[16] Ran Canetti and Hugo Krawczyk. 2001. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT 2001 (LNCS)*, Birgit Pfitzmann (Ed.), Vol. 2045. Springer, Heidelberg, 453–474. https://doi.org/10.1007/3-540-44987-6_28

[17] Ran Canetti and Hugo Krawczyk. 2002. Security Analysis of IKE's Signature-based Key-Exchange Protocol. In *CRYPTO 2002 (LNCS)*, Moti Yung (Ed.), Vol. 2442. Springer, Heidelberg, 143–161. https://doi.org/10.1007/3-540-45708-9_10 https://eprint.iacr.org/2002/120/.

[18] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. 2012. Bonsai Trees, or How to Delegate a Lattice Basis. *Journal of Cryptology* 25, 4 (Oct. 2012), 601–639. https://doi.org/10.1007/s00145-011-9105-2

[19] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. 2018. CSIDH: An Efficient Post-Quantum Commutative Group Action. In *ASIACRYPT 2018, Part III (LNCS)*, Thomas Peyrin and Steven Galbraith (Eds.), Vol. 11274. Springer, Heidelberg, 395–427. https://doi.org/10.1007/978-3-030-03332-3_15

[20] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. 2020. Breaking the Decisional Diffie-Hellman Problem for Class Group Actions Using Genus Theory. In *CRYPTO 2020, Part II (LNCS)*, Daniele Micciancio and Thomas Ristenpart (Eds.), Vol. 12171. Springer, Heidelberg, 92–120. https://doi.org/10.1007/978-3-030-56880-1_4

[21] Melissa Chase, David Derler, Steven Goldfeder, Daniel Kales, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Xiao Wang, and Greg Zaverucha. 2020. The Picnic Signature Scheme: Design Document. (Sept. 2020). https://microsoft.github.io/Picnic/ Submission to NIST Post-Quantum Standardization Project, Round 3.

[22] Katriel Cohn-Gordon, Cas J. F. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2017. A Formal Security Analysis of the Signal Messaging Protocol. In *IEEE European Symposium on Security and Privacy, EuroS&P 2017*. 451–466. https://doi.org/10.1109/EuroSP.2017.27

[23] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. 2016. On Post-compromise Security. In *CSF 2016Computer Security Foundations Symposium*, Michael Hicks and Boris Köpf (Eds.). IEEE Computer Society Press, 164–178. https://doi.org/10.1109/CSF.2016.19

[24] Cas Cremers and Michele Feltz. 2011. 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.

[25] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. 2013. A Cryptographic Analysis of OPACITY - (Extended Abstract). In *ESORICS 2013 (LNCS)*, Jason Crampton, Sushil Jajodia, and Keith Mayes (Eds.), Vol. 8134. Springer, Heidelberg, 345–362. https://doi.org/10.1007/978-3-642-40203-6_20

[26] Ivan Damgård, Helene Haagh, Rebekah Mercer, Anca Nitulescu, Claudio Orlandi, and Sophia Yakoubov. 2019. Stronger Notions and Constructions for Multi-Designated Verifier Signatures. Cryptology ePrint Archive, Report 2019/1153. (2019). https://eprint.iacr.org/2019/1153.

[27] Bor de Kock, Kristian Gjøsteen, and Mattia Veroni. 2020. Practical Isogeny-Based Key-exchange with Optimal Tightness. In *27th Conference on Selected Areas in Cryptography (SAC)*. Springer.

[28] Cyprien Delpech de Saint Guilhem, Nigel P. Smart, and Bogdan Warinschi. 2017. Generic Forward-Secure Key Agreement Without Signatures. In *ISC 2017 (LNCS)*, Phong Q. Nguyen and Jianying Zhou (Eds.), Vol. 10599. Springer, Heidelberg, 114–133.

[29] Mario Di Raimondo and Rosario Gennaro. 2005. New Approaches for Deniable Authentication. In *ACM CCS 2005*, Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels (Eds.). ACM Press, 112–121. https://doi.org/10.1145/1102120.1102137

[30] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. 2006. Deniable authentication and key exchange. In *ACM CCS 2006*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, 400–409. https://doi.org/10.1145/1180405.1180454

[31] Samuel Dobson, Steven D. Galbraith, Jason LeGrow, Yan Bo Ti, and Lukas Zobernig. 2019. An Adaptive Attack on 2-SIDH. Cryptology ePrint Archive, Report 2019/890. (2019). https://eprint.iacr.org/2019/890.

[32] Samuel Dobson, Trey Li, and Lukas Zobernig. 2019. A Note on a Static SIDH Protocol. Cryptology ePrint Archive, Report 2019/1244. (2019). https://eprint.iacr.org/2019/1244.

[33] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. 2009. Composability and On-Line Deniability of Authentication. In *TCC 2009 (LNCS)*, Omer Reingold (Ed.), Vol. 5444. Springer, Heidelberg, 146–162. https://doi.org/10.1007/978-3-642-00457-5_10

[34] Léo Ducas and Daniele Micciancio. 2014. Improved Short Lattice Signatures in the Standard Model. In *CRYPTO 2014, Part I (LNCS)*, Juan A. Garay and Rosario Gennaro (Eds.), Vol. 8616. Springer, Heidelberg, 335–352. https://doi.org/10.1007/978-3-662-44371-2_19

[35] Ines Duits. 2019. *The Post-Quantum Signal Protocol: Secure Chat in a Quantum World.* Master's thesis. University of Twente. http://essay.utwente.nl/77239/

[36] Cynthia Dwork, Moni Naor, and Amit Sahai. 1998. Concurrent Zero-Knowledge. In *30th ACM STOC*. ACM Press, 409–418. https://doi.org/10.1145/276698.276853

[37] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS)*, Andrew M. Odlyzko (Ed.), Vol. 263. Springer, Heidelberg, 186–194. https://doi.org/10.1007/3-540-47721-7_12

[38] Marc Fischlin and Sogol Mazaheri. 2015. Notions of Deniable Message Authentication. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, WPES 2015, Denver, Colorado, USA, October 12, 2015*, Indrajit Ray, Nicholas Hopper, and Rob Jansen (Eds.). ACM, 55–64. https://doi.org/10.1145/2808138.2808143

[39] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. 2020. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. (Oct. 2020). https://falcon-sign.info Submission to NIST Post-Quantum Standardization Project, Round 3.

[40] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. 2013. Non-Interactive Key Exchange. In *PKC 2013 (LNCS)*, Kaoru Kurosawa and Goichiro Hanaoka (Eds.), Vol. 7778. Springer, Heidelberg, 254–271. https://doi.org/10.1007/978-3-642-36362-7_17

[41] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. 2012. Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices. In *PKC 2012 (LNCS)*, Marc Fischlin, Johannes Buchmann, and Mark Manulis (Eds.), Vol. 7293. Springer, Heidelberg, 467–484. https://doi.org/10.1007/978-3-642-30057-8_28

[42] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO'99 (LNCS)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, Heidelberg, 537–554. https://doi.org/10.1007/3-540-48405-1_34

[43] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. 2016. On the Security of Supersingular Isogeny Cryptosystems. In *ASIACRYPT 2016, Part I (LNCS)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.), Vol. 10031. Springer, Heidelberg, 63–91. https://doi.org/10.1007/978-3-662-53887-6_3

[44] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, Richard E. Ladner and Cynthia Dwork (Eds.). ACM Press, 197–206. https://doi.org/10.1145/1374376.1374407

[45] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. 2021. An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable. In *PKC 2021, Part II (LNCS)*, Juan Garay (Ed.), Vol. 12711. Springer, Heidelberg, 410–440. https://doi.org/10.1007/978-3-030-75248-4_15

[46] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A Ring Based Public Key Cryptosystem. In *3rd Algorithmic Number Theory Symposium (ANTS III) (LNCS)*, Vol. 1423. Springer, 267–288.

[47] Andreas Hülsing and Florian Weber. 2021. Epochal Signatures for Deniable Group Chats. In *2021 IEEE Symposium on Security and Privacy*. IEEE Computer

Society Press, 997–1015. https://doi.org/10.1109/SP40001.2021.00058
[48] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. 1996. Designated Verifier Proofs and Their Applications. In *EUROCRYPT'96 (LNCS)*, Ueli M. Maurer (Ed.), Vol. 1070. Springer, Heidelberg, 143–154. https://doi.org/10.1007/3-540-68339-9_13
[49] David Jao and Luca De Feo. 2011. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, Bo-Yin Yang (Ed.). Springer, Heidelberg, 19–34. https://doi.org/10.1007/978-3-642-25405-5_2
[50] Tomoki Kawashima, Katsuyuki Takashima, Yusuke Aikawa, and Tsuyoshi Takagi. 2020. An Efficient Authenticated Key Exchange from Random Self-Reducibility on CSIDH. Cryptology ePrint Archive, Report 2020/1178. (2020). https://eprint.iacr.org/2020/1178.
[51] Hugo Krawczyk. 1996. SKEME: A versatile secure key exchange for Internet. In *Network and Distributed Systems Security Symposium (NDSS)*. Internet Society and IEEE, 114–127.
[52] Hugo Krawczyk. 2005. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO 2005 (LNCS)*, Victor Shoup (Ed.), Vol. 3621. Springer, Heidelberg, 546–566. https://doi.org/10.1007/11535218_33
[53] Hugo Krawczyk and Tal Rabin. 1998. Chameleon Hashing and Signatures. Cryptology ePrint Archive, Report 1998/010. (1998). https://eprint.iacr.org/1998/010.
[54] Kaoru Kurosawa and Jun Furukawa. 2014. 2-Pass Key Exchange Protocols from CPA-Secure KEM. In *CT-RSA 2014 (LNCS)*, Josh Benaloh (Ed.), Vol. 8366. Springer, Heidelberg, 385–401. https://doi.org/10.1007/978-3-319-04852-9_20
[55] Fabien Laguillaumie and Damien Vergnaud. 2005. Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map. In *SCN 04 (LNCS)*, Carlo Blundo and Stelvio Cimato (Eds.), Vol. 3352. Springer, Heidelberg, 105–119. https://doi.org/10.1007/978-3-540-30598-9_8
[56] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. 2007. Stronger Security of Authenticated Key Exchange. In *ProvSec 2007 (LNCS)*, Willy Susilo, Joseph K. Liu, and Yi Mu (Eds.), Vol. 4784. Springer, Heidelberg, 1–16.
[57] BaoHong Li, YanZhi Liu, and Sai Yang. 2018. Lattice-based universal designated verifier signatures. In *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*. IEEE, 329–334. https://doi.org/10.1109/ICEBE.2018.00062
[58] Yong Li, Willy Susilo, Yi Mu, and Dingyi Pei. 2007. Designated Verifier Signature: Definition, Framework and New Constructions. In *Ubiquitous Intelligence and Computing, 4th International Conference, UIC 2007, Hong Kong, China, July 11-13, 2007, Proceedings (Lecture Notes in Computer Science)*, Jadwiga Indulska, Jianhua Ma, Laurence Tianruo Yang, Theo Ungerer, and Jiannong Cao (Eds.), Vol. 4611. Springer, 1191–1200. https://doi.org/10.1007/978-3-540-73549-6_116
[59] Vadim Lyubashevsky. 2009. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *ASIACRYPT 2009 (LNCS)*, Mitsuru Matsui (Ed.), Vol. 5912. Springer, Heidelberg, 598–616. https://doi.org/10.1007/978-3-642-10366-7_35
[60] Vadim Lyubashevsky. 2012. Lattice Signatures without Trapdoors. In *EUROCRYPT 2012 (LNCS)*, David Pointcheval and Thomas Johansson (Eds.), Vol. 7237. Springer, Heidelberg, 738–755. https://doi.org/10.1007/978-3-642-29011-4_43
[61] Vadim Lyubashevsky and Gregory Neven. 2017. One-Shot Verifiable Encryption from Lattices. In *EUROCRYPT 2017, Part I (LNCS)*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.), Vol. 10210. Springer, Heidelberg, 293–323. https://doi.org/10.1007/978-3-319-56620-7_11
[62] Moxie Marlinspike and Trevor Perrin. November 2016. The double ratchet algorithm. (November 2016). https://www.signal.org/docs/specifications/doubleratchet/
[63] Moxie Marlinspike and Trevor Perrin. November 2016. The X3DH key agreement protocol. (November 2016). https://signal.org/docs/specifications/x3dh/
[64] Geontae Noh and Ik Rae Jeong. 2017. Strong designated verifier signature scheme from lattices in the standard model. *Security Comm. Networks* 9 (Feb. 2017), 6202–6214. https://doi.org/10.1002/sec.1766
[65] Rafael Pass. 2003. On Deniability in the Common Reference String and Random Oracle Model. In *CRYPTO 2003 (LNCS)*, Dan Boneh (Ed.), Vol. 2729. Springer, Heidelberg, 316–337. https://doi.org/10.1007/978-3-540-45146-4_19
[66] Chris Peikert. 2020. He Gives C-Sieves on the CSIDH. In *EUROCRYPT 2020, Part II (LNCS)*, Anne Canteaut and Yuval Ishai (Eds.), Vol. 12106. Springer, Heidelberg, 463–492. https://doi.org/10.1007/978-3-030-45724-2_16
[67] Bertram Poettering and Paul Rösler. 2018. Towards Bidirectional Ratcheted Key Exchange. In *CRYPTO 2018, Part I (LNCS)*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10991. Springer, Heidelberg, 3–32. https://doi.org/10.1007/978-3-319-96884-1_1
[68] Oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, Harold N. Gabow and Ronald Fagin (Eds.). ACM Press, 84–93. https://doi.org/10.1145/1060590.1060603
[69] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. 2004. An Efficient Strong Designated Verifier Signature Scheme. In *ICISC 03 (LNCS)*, Jong In Lim and Dong Hoon Lee (Eds.), Vol. 2971. Springer, Heidelberg, 40–54.
[70] Peter Schwabe, Douglas Stebila, and Thom Wiggers. 2020. Post-Quantum TLS Without Handshake Signatures. In *ACM CCS 20*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 1461–1480. https://doi.org/10.1145/3372297.3423350
[71] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. 2003. Universal Designated-Verifier Signatures. In *ASIACRYPT 2003 (LNCS)*, Chi-Sung Laih (Ed.), Vol. 2894. Springer, Heidelberg, 523–542. https://doi.org/10.1007/978-3-540-40061-5_33
[72] Xi Sun, Haibo Tian, and Yumin Wang. 2012. Toward Quantum-Resistant Strong Designated Verifier Signature from Isogenies. In *4th International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 292–296. https://doi.org/10.1109/iNCoS.2012.70
[73] Nik Unger and Ian Goldberg. 2015. Deniable Key Exchanges for Secure Messaging. In *ACM CCS 2015*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM Press, 1211–1223. https://doi.org/10.1145/2810103.2813616
[74] Nik Unger and Ian Goldberg. 2018. Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging. *PoPETs* 2018, 1 (Jan. 2018), 21–66. https://doi.org/10.1515/popets-2018-0003
[75] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. 2020. On the Cryptographic Deniability of the Signal Protocol. In *ACNS 20, Part II (LNCS)*, Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi (Eds.), Vol. 12147. Springer, Heidelberg, 188–209. https://doi.org/10.1007/978-3-030-57878-7_10
[76] Fenghe Wang, Yupu Hu, and Baocang Wang. 2012. Lattice-based strong designate verifier signature and its applications. *Malaysian Journal of Computer Science* 25 (2012), 11–22. Issue 1.
[77] Fenghe Wang, Yupu Hu, and Baocang Wang. 2014. Identity-based strong designate verifier signature over lattices. *The Journal of China Universities of Post and Telecommunications* 21 (2014), 52–60. Issue 6. https://doi.org/10.1016/S1005-8885(14)60345-9
[78] Bo Yang, Yong Yu, and Ying Sun. 2013. A novel construction of SDVS with secure disavowability. *Clust. Comput.* 16, 4 (2013), 807–815. https://doi.org/10.1007/s10586-013-0254-y
[79] Yongqiang Zhang, Qiang Liu, Chengpei Tang, and Haibo Tian. 2015. A lattice-based designated verifier signature for cloud computing. *International Journal of High Performance Computing and Networking* 8 (June 2015), 135–143. Issue 2. https://doi.org/10.1504/IJHPCN.2015.070013

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. [73] 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 [24, 25, 30, 73, 74]. 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 [30]. Given the list of all public keys and some auxiliary information (e.g., some legal transcripts), the attacker 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 [30] 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 [24] 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.

Dagdelen, Fischlin, Gagliardoni, Marson, Mittelbach, and Onete [25] proposed a game-based definition called *outsider deniability*. Here, the adversary has access to Init, Exec, Send, Reveal, Corrupt, and Register 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 [73, 74], 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 [73] of post-specified peers.

In [75], Vatandas, Gennaro, Ithurburn, and Krawczyk provided an analysis that Signal's X3DH is deniable (wrt. full deniability of [30]) under a general extractability assumption. The authors emphasize the observation of Pass in [65] that a simulator for deniability must be a real algorithm (unlike a Zero-Knowledge simulator, which can be thought of as a thought experiment allowing, e.g., re-winding).

Recently, Hülsing and Weber have defined deniability for group chats (and not just key exchange) in [47]. 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 7.2, 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 the game-based definition.

We further take into account the informal requirement on deniability for asynchronous DAKE in [63, §4.4]: *Neither party has a proof of the fact that both parties communicated and a purported transcript of the execution can be produced by any party that has access to one of the party's secret keys.* This informal description implies a relaxation compared to previous definitions: The simulated transcript may make use of the secret key of either party.

Please observe that we only consider asynchronous DAKE, which consist of only one message with a specified peer. Hence, we do not need to take any special precautions to achieve deniability for concurrent executions. Furthermore, we give the adversary (i.e., the distinguisher) access to all secret keys. This models the scenario where 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.