Post-quantum Asynchronous Deniable Key Exchange and the Signal Handshake

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

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

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

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

1 Introduction

The Signal protocol [67,66], 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" [67], 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 [66]), and symmetric authenticated encryption for application data. Each of these components contributes to Signal's interesting and useful security features:

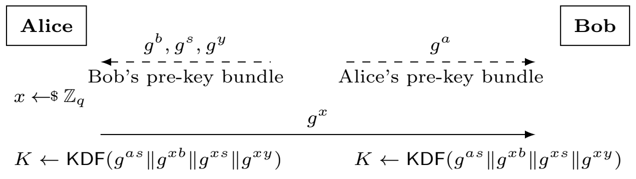
Alice
Bob
g^b, g^s, g^y
Bob's pre-key bundle
g^a
Alice's pre-key bundle
$x \leftarrow\$ \mathbb{Z}_q$
g^x
$K \leftarrow \mathsf{KDF}(g^{as} \| g^{xb} \| g^{xs} \| g^{xy})$
$K \leftarrow \mathsf{KDF}(g^{as} \| g^{xb} \| g^{xs} \| g^{xy})$

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

– *Implicit mutual authentication in the handshake*: The session key K established in the handshake can only be computed by the intended peer. This comes from the terms involving the long-term secret keys a and b in Figure 1.

– *Forward secrecy in the handshake*: The session key K established in the handshake remains secret even if long-term keys are later compromised. This comes from the terms involving the ephemeral keys x and y in Figure 1.

– *Offline deniability of the handshake*: A judge seeing a transcript of an honest communication session cannot be convinced that a particular party was actually involved in the session. This comes from the use of Diffie–Hellman for authentication rather than signatures; all of the DH shared secrets input to the key derivation function in Figure 1 could have been computed unilaterally either by Alice or by Bob (e.g., both Alice and Bob can compute g^{as}, using a and s respectively). We provide a new formalization of deniability reflecting the specification of Signal more closely. We discuss the differences between the deniability notions in Section 3 and in more detail in the full version. While a formal proof that X3DH fulfills our new notion is not known to the authors, we expect it to hold without any additional assumptions. See [81] for a detailed analysis of the deniability of X3DH with respect to the deniability notion of [29].

– *Asynchronicity*: The two communicating parties need never be online simultaneously, and can leave packets at an untrusted relay server until the other party comes back online. The handshake is made asynchronous by allowing each party to upload a *pre-key bundle* to an untrusted server in advance, consisting of long-term, medium-term, and ephemeral public keys, and an initiator can start sending text messages before their peer comes online. The restrictions on communication flow in an asynchronous protocol are weaker than those of non-interactive key exchange [41].

– *Forward secrecy and post-compromise security [22] in long-lived conversations*: Keys are updated using a new DH key exchange with each chat message via the asymmetric ratchet, enabling secrecy of past and future messages after a compromise.

1.1 Making Signal Post-quantum

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

The symmetric ratchet and authenticated encryption components of Signal are built on symmetric primitives, and thus are not in immediate danger from quantum algorithms. The asymmetric ratchet was phrased by Marlinspike and Perrin [66] and analyzed by Cohn-Gordon, Cremers, Dowling, Garratt, and Stebila [21] in terms of Diffie–Hellman. Alwen, Coretti, and Dodis [1] generalized it into a primitive called continuous key agreement that can be built from KEMs, yielding post-quantum security. Hence, our focus in the rest of this paper is on the handshake.

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

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

1.2 Options for PQ Asynchronous DAKE

There are several examples of authenticated key exchange protocols built generically from KEMs which have the potential for deniability [12,11,42,27,75] 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 [20], a primitive based on supersingular isogenies that yields a commutative group action which enables non-interactive key exchange. CSIDH could be used to achieve implicit Alice-to-Bob authentication while maintaining asynchronicity and deniability; indeed several key exchange protocols from CSIDH have been proposed [26,52]. Unfortunately, there are several reasons CSIDH may not be a fully satisfactory solution: it is much more computationally expensive than most other forms of post-quantum cryptography, and there is ongoing debate about the security of its concrete parameters [69,10].

Most other post-quantum assumptions used in KEMs, including SIDH [50] and learning-with-errors (LWE) [72], are insecure against key reuse attacks without additional protection such as the Fujisaki–Okamoto transform [43] 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 [2,36,31,32], and one attempt relying on an additional novel assumption [9] the security of which is unknown.

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

Unger and Goldberg [79,80] also consider deniable authenticated key exchange (DAKE) protocols for secure messaging. Their protocol permits the optional use of a PQ KEM for ephemeral key exchange to achieve forward secrecy against future-quantum adversaries. To achieve deniability, they employ ring signatures with classical security and further rely on dual receiver encryption, which does not yet appear to have a PQ instantiation in the literature. Observe that their formalization of deniability is given in the UC model.

The recent work by Hashimoto, Katsumata, Kwiatkowski, and Prest [46] is closest to ours. Their core protocol is meant to replace the Signal handshake based on (post-quantum) KEMs and signatures. It achieves security against exposure of long-term keys and session state and a weaker deniability level. Unlike Signal (and our proposed protocol), it does however not provide security against randomness exposure and lacks support for semi-static keys to mitigate the exhaustion of ephemeral pre-keys. Hashimoto et al. provide an implementation

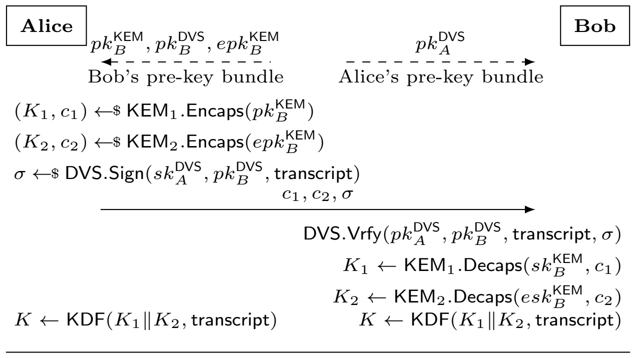
Alice
Bob
$pk_B^{\mathsf{KEM}}, pk_B^{\mathsf{DVS}}, epk_B^{\mathsf{KEM}}$
Bob's pre-key bundle
pk_A^{DVS}
Alice's pre-key bundle
$(K_1, c_1) \leftarrow\$ \mathsf{KEM}_1.\mathsf{Encaps}(pk_B^{\mathsf{KEM}})$
$(K_2, c_2) \leftarrow\$ \mathsf{KEM}_2.\mathsf{Encaps}(epk_B^{\mathsf{KEM}})$
$\sigma \leftarrow\$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

Fig. 2. Our core asynchronous DAKE protocol, combining static and ephemeral key encapsulation schemes KEM_1 and KEM_2, and a designated verifier signature DVS (top); and the corresponding Fake algorithm to forge a transcript for deniability (bottom).

for their weakly-deniable protocol and further discuss two additional variants achieving stronger deniability. The second protocol achieves deniability against semi-honest adversaries based on ring signatures, while their third protocol additionally uses non-interactive zero-knowledge arguments and strong knowledge-type assumptions for plaintext-aware [4] KEMs to achieve deniability against malicious adversaries.

Dobson and Galbraith [30] recently proposed using SIDH key exchange to replace the DH key exchange in the (slightly modified) X3DH protocol. Even though SIDH is in general insecure against adaptive attacks, Dobson and Galbraith show that carefully adding a zero-knowledge proof enables them to prove that the long-term SIDH public keys are generated honestly. In order to prove deniability, they require strong knowledge-type assumptions following [81].

1.3 Our Contributions

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

Introduced by Jakobsson, Sako, and Impagliazzo [49], a *designated verifier signature (DVS) scheme* allows a signer to convince a chosen recipient, called the designated verifier, of the authenticity of a message, but in such a way that the designated verifier cannot convince any other party of the authenticity. In a DVS scheme, both the signer and the verifier have a public-key/secret-key pair; signing requires both the signer's secret key and the verifier's public key, and verification uses both parties' public keys. To achieve the non-transferability property (called "source hiding"), a DVS scheme is accompanied by an additional simulation algorithm with which the designated verifier can, using its own secret key, construct a signature indistinguishable from one generated by the signer.

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

Deniability. We model the informal deniability requirement from the Signal specification [67, §4.4] through a new deniability notion (for asynchronous DAKE) capturing the following scenario: Alice wants to convince a judge that a certain conversation took place between her and Bob. Hence, Alice gives the corresponding transcript to the judge. The judge may coerce Alice and Bob to give up their secret keys (e.g., by law). Under these circumstances, the judge should not be able to tell if this transcript stems from a real conversation or if Alice faked the transcript on her own without Bob's interaction. On the one hand, our new notion is weaker than the definition of [29] in the sense that we limit Alice to stick to the protocol description (i.e., be semi-honest) and allow the use of a secret key for faking a transcript. On the other hand, our new notion is stronger in the sense that we allow the judge to know *all* secret keys. On a more technical note, we provide a game-based definition while [29] uses the simulation paradigm. In a nutshell, a strength of our notion is that it provides deniability against powerful judges that can compromise secret keys of users. A consequence of our new, incomparable deniability definition is that we can achieve it without strong knowledge assumptions needed for X3DH [81] and in the work of Hashimoto et al. [46,47]; we conjecture both protocols can likewise be shown to be deniable wrt. our definition without such assumptions.

Post-quantum designated verifier signatures. To achieve our goal of post-quantum asynchronous DAKE, we thus need a post-quantum designated verifier signature scheme. While there is a long line of research on DVS schemes from prequantum assumptions (including [49,74,55,77,59,84,16,25]), comparatively little is available in the literature on post-quantum DVS schemes. An isogeny-based DVS scheme was proposed in [78] but is insecure due to key reuse attacks identified in [44]. There are several lattice-based DVS schemes which may fit the bill [82,83,68,57,87], but these have not received much scrutiny in the mainstream cryptographic literature; we summarize this literature in Section 2.1. These lattice-based DVS schemes are direct constructions not based on any NIST candidates, so they would require their own thorough analysis.

DVS from ring signatures. Rather than constructing DVS schemes directly, it is possible to use a *ring signature scheme* [73] as a designated verifier signature scheme. In a ring signature scheme, one signer can sign a message intended to verify under a *ring* of public keys, only one of which is theirs; yet no one should be able to determine which signer produced such a signature. Following ideas sketched in [73,6], we show in Section 2.2 how to use a 2-user ring signature scheme to build a DVS scheme: the ring used by the signer consists of the public keys of themselves and the one designated verifier. There are several candidates for post-quantum ring signatures whose properties we discuss in Section 2.2.

In a concurrent update to their work, Hashimoto et al. have shown the reverse, i.e., constructing a ring signature scheme from a DVS scheme [47] (which is the full version of [46]). Hence, in the 2-user case ring signatures and DVS are equivalent under the security notions put forward in this paper.

Given this equivalence, observe that our core asynchronous DAKE protocol (Figure 2) is indeed similar to the second construction of [46]. While our construction sends the DVS signature as is, their construction employs a ring signature that is masked with the output of a PRF evaluation.

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

Outline of the paper. Section 2 focuses on the security properties of designated verifier schemes and how to construct these in a post-quantum setting, including existing direct constructions as well as via ring signatures; the full version of this paper [14] gives a discussion of our failed attempts at constructing DVS from chameleon hash functions in an earlier version of this work. In Section 3 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 4 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 5 we introduce a complete post-quantum version of the Signal handshake that extends our core protocol to include additional components present in the Signal handshake (e.g., semi-static keys); we provide a security model and proof of session key indistinguishability and deniability for the full protocol in the full version [14]. In Section 6, we conclude with a discussion of the results and some limitations.

1.4 Notation

To sample an element x uniformly at random from a set $\mathcal{S}$ (or a distribution on an underlying set) we write $x \leftarrow_{\$} \mathcal{S}$. For deterministic algorithms A we denote by $y \leftarrow \mathsf{A}(x)$ the execution of A on input x with output y. Similarly, $y \leftarrow_{\$} \mathsf{A}(x)$

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

2 Designated Verifier Signatures

Designated verifier signature (DVS) schemes were introduced by Jakobsson, Sako, and Impagliazzo [49]. Their goal is for a signer to convince a chosen recipient (the "designated verifier") that a message is authentic but in such a way that the designated verifier cannot convince any other party of the authenticity of the message[4]. This property is typically modeled by requiring that the designated verifier can efficiently simulate signatures that are indistinguishable from signatures produced by the signer.

Definition 1. *A* designated verifier signature scheme *(DVS) is a tuple of algorithms* $\mathsf{DVS} = (\mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Sim})$ *along with a message space* $\mathcal{M}$.

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

A DVS scheme DVS *is* correct*, if, for any honestly generated key pairs* (pk_S, sk_S), (pk_D, sk_D) *and every message* $m \in \mathcal{M}$*, it holds that* $\Pr[\mathsf{Vrfy}(pk_S, pk_D, m, \mathsf{Sign}(sk_S, pk_D, m)) = \mathsf{true}] = 1$.

We follow Laguillaumie and Vergnaud [55] in defining separate key generation algorithms for signers and designated verifiers; in some cases these two algorithms may be identical.

A long line of research has scrutinized the security of DVS schemes in different settings, e.g. strong DVS schemes, including [49,74,55,77,59,84,16,25]. For the

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

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

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

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

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

Fig. 3. Unforgeability (top) and source hiding (bottom) of a designated verifier signature scheme DVS.

purpose of this paper, it suffices to define the security notions of *unforgeability* and *source hiding*. Unforgeability for DVS schemes is similar to that for standard signature schemes, providing the adversary with a signing oracle and asking it to forge a signature on a (fresh) message of its choice. Prior work restricts the signing oracle to the challenge designated verifier key. In contrast, and to account for settings where a signer's key is used with many other users' verifier keys (cf. Section 4), we allow the adversary to pick the designated verifier key to be used in the signing oracle from a set of additional, honestly generated key pairs.

Definition 2. *A designated verifier signature scheme* DVS *is* (t, ϵ, n, Q_S)-unforgeable *if, for any adversary* $\mathcal{A}$ *with running time at most* t*, having access to* n *additional DVS verifier key pairs beyond the challenge keys, and making at most* Q_S *queries to the* Sign *oracle, we have that* $\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A}) = 1\right] \leq \epsilon$*, where* $\mathcal{G}^{\mathsf{uf}}_{\mathsf{DVS}}(\mathcal{A})$ *is as in Figure 3.*

The second property we consider is called source hiding [55], demanding that it should be infeasible for an adversary to determine whether a given signature has been generated by the signer (using Sign) or by the designated verifier (using Sim), even if the adversary learns the secret keys of both parties.

Definition 3. *A designated verifier signature scheme* DVS *is* (t, ϵ, Q_{Ch})-source hiding *if, for any adversary* $\mathcal{A}$ *with running time at most* t *and making at most* Q_{Ch} *to the* Chall *oracle, we have that* $\mathsf{Adv}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A}) = \left|\Pr\left[\mathcal{G}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right| \leq \epsilon$*, where* $\mathcal{G}^{\mathsf{srchid}}_{\mathsf{DVS}}(\mathcal{A})$ *is defined in Figure 3.*

The property of source hiding also appears under different terms in the literature such as the designated verifier property [49,74], non-transferability [77], source deniable [40], untransferability [16], and recently off-the-record [25]. While all these definitions share the intuition that the sender can blame another party (in particular, the designated receiver) as the originator of a signature, they vary in the adversary capabilities, i.e., whether the adversary is unbounded or whether it gets access to the secret keys.

2.1 Post-quantum DVS Schemes: Prior Work

For this work, we are interested in DVS constructions that promise post-quantum security. Despite the long line of research on DVS schemes, there are only a few candidate post-quantum constructions available in the literature; furthermore, most of those have not received much scrutiny in the mainstream cryptographic literature. In the following, we summarize prior direct constructions before turning to generic constructions from ring signatures in Section 2.2.

An isogeny-based strong DVS scheme was proposed by Sun, Tian, and Wang [78] which turned out to be insecure due to key reuse attacks identified by Galbraith, Petit, Shani, and Ti [44].

Wang, Hu, and Wang [82] construct a strong DVS scheme directly from lattice assumptions (LWE and SIS) by combining the Bonsai tree lattice trapdoor of [19] with the GPV lattice-based signature scheme [45]; a subsequent paper of theirs [83] extends this to the identity-based setting.

Noh and Jeong [68] improve on [82,83] by giving direct constructions from lattices that can be proven without relying on random oracles; they do so by replacing the random oracle with a chameleon hash function.

Li, Liu, and Yang [57] construct a universal DVS scheme directly from ideal lattice assumptions (ring-SIS) by combining a ring version of the GPV signature scheme [64] with a ring chameleon hash function [35] and adding a Fiat–Shamir-with-aborts technique [62,63].

Zhang, Liu, Tang, and Tian [87] also give a universal DVS constructed directly from SIS by adapting the Lyubashevsky signature scheme [63].

2.2 Building Post-quantum DVS Schemes from Ring Signatures

We now turn to building DVS schemes generically from ring signatures, show which properties are required to obtain a post-quantum-secure instantiation and evaluate several ring signature candidates. Our constructions draws from the idea sketched in [73,6], with syntax and security closely following the exposition of Bender, Katz, and Morselli [6].

Definition 4. *A* ring signature scheme *is a tuple of algorithms* $\mathsf{Ring} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *along with a message space* $\mathcal{M}$.

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

– $\mathsf{Vrfy}(\mathsf{R}, m, \sigma) \to \mathsf{true}/\mathsf{false}$*: A deterministic verification algorithm that checks a message* m *and signature* σ *against a ring* R.

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

The unforgeability and anonymity property we require for ring signatures are subtly different from prior literature. Like in the unforgeability notion w.r.t. insider corruption defined in [6], we consider an unforgeability adversary with access to a corruption oracle CORR. However, our unforgeability adversary is limited to rings consisting of honestly generated public keys for both its final forgery as well as the queries to the signing oracle (like the unforgeability against chosen-subring attacks defined in [6]). It is easy to see that unforgeability w.r.t. insider corruption implies our unforgeability notion. Herranz [48] informally discusses a similar notion.

Definition 5. *A ring signature scheme* Ring *is* $(t, \epsilon, n, Q_S, Q_{Co})$-unforgeable w.r.t. honest-ring insider corruption *if, for any adversary* $\mathcal{A}$ *with running time at most* t*, having access to* n *public keys, and making at most* Q_S *queries to the* SIGN *oracle and* Q_{Co} *queries to the* CORR *oracle, we have that* $\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A}) = 1\right] \leq \epsilon$*, where* $\mathcal{G}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A})$ *is as in Figure 4.*

We consider an anonymity notion based on anonymity against full key exposure [6]. The first difference is that we directly give all secret keys to the adversary instead of providing a signing and a corruption oracle to the adversary, where the latter in [6] returns the key generation randomness. The other difference is that we parameterize the game in the number of queries Q_{Ch} allowed to the challenge oracle. As a result, anonymity against full key exposure implies our anonymity notion with $Q_{Ch} = 1$. Similarly, the anonymity notions of [60] and [39], where the attacker has access to a key generation oracle, imply our anonymity notion with $Q_{Ch} = 1$.

Definition 6. *A ring signature scheme* Ring *is* (t, ϵ, n, Q_{Ch})-anonymous against key exposure *if, for any adversary* $\mathcal{A}$ *with running time at most* t*, having access to* n *key pairs, and making at most* Q_{Ch} *queries to the* CHALL *oracle, we have that* $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A}) = 1\right] \leq \epsilon$*, where* $\mathcal{G}^{\mathsf{anon}}_{\mathsf{Ring}}(\mathcal{A})$ *is as in Figure 4.*

It is easy to see that one can transform any $(t, \epsilon, n, 1)$-anonymous (as per Definition 6) ring signature scheme into a $(t, \epsilon \cdot Q_{Ch}, n, Q_{Ch})$-anonymous scheme via a hybrid argument.

The construction. Our construction, denoted $\mathsf{RingDVS}$, is a straightforward adaption of a 2-user ring signature Ring to the DVS setting as detailed in Figure 5. The security of the resulting DVS scheme hinges on the unforgeability and anonymity of the ring signature as per Definitions 5 and 6.

```
$\mathcal{G}^{\mathsf{uf}}_{\mathsf{Ring}}(\mathcal{A})$:
1  $Q_S \leftarrow \emptyset$
2  $Q_{Co} \leftarrow \emptyset$
3  $\mathcal{L} \leftarrow \emptyset$
4  for $i \in [n]$
5      $(pk_i, sk_i) \leftarrow\$ \mathsf{Ring.KGen}()$
6      $\mathcal{L} \leftarrow \mathcal{L} \cup \{pk_i\}$
7  $(\mathsf{R}^\star, m^\star, \sigma^\star) \leftarrow\$ \mathcal{A}^{\textsc{Sign},\textsc{Corr}}(\mathcal{L})$
8  $d_1 \leftarrow \mathsf{Ring.Vrfy}(\mathsf{R}^\star, m^\star, \sigma^\star)$
9  $d_2 \leftarrow [\![(m^\star, \mathsf{R}^\star) \notin Q_S]\!]$
10 $d_3 \leftarrow [\![\mathsf{R}^\star \subseteq \mathcal{L} \backslash Q_{Co}]\!]$
11 return $[\![d_1 \wedge\ d_2 \wedge\ d_3]\!]$

$\textsc{Sign}(s, m, \mathsf{R})$:
12 if $pk_s \notin \mathsf{R} \vee s \notin [n]$ //sign wrt. honest key
13     return $\perp$
14 if $\mathsf{R} \not\subseteq \mathcal{L}$ //sign wrt. honest ring
15     return $\perp$
16 $Q_S \leftarrow Q_S \cup \{(m, \mathsf{R})\}$
17 $\sigma \leftarrow\$ \mathsf{Ring.Sign}(sk_S, m, \mathsf{R})$
18 return $\sigma$

$\textsc{Corr}(i)$:
19 $Q_{Co} \leftarrow Q_{Co} \cup \{pk_i\}$
20 return $sk_i$
```

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

$\textsc{Chall}(m, i_0, i_1, \mathsf{R})$:
8  if $\{pk_{i_0}, pk_{i_1}\} \not\subseteq \mathsf{R}$ //challenge signers in ring
9      return $\perp$
10 if $\{i_0, i_1\} \not\subseteq [n]$ //sign with honest keys only
11     return $\perp$
12 $\sigma \leftarrow\$ \mathsf{Ring.Sign}(sk_{i_b}, m, \mathsf{R})$
13 return $\sigma$
```

Fig. 4. Unforgeability w.r.t. honest-ring insider corruption (top) and anonymity against key exposure (bottom) of a ring signature scheme Ring. The latter game is specialized for the ring size 2.

Theorem 1 (Unforgeability of RingDVS). *If* Ring *is a* $(t, \epsilon, n+2, Q_S, Q_{Co})$-*unforgeable w.r.t. honest-ring insider corruption 2-user ring signature scheme, then* RingDVS *defined in Figure 5 is* (t', ϵ, n, Q_S)*-existentially unforgeable under chosen message attacks, with* $t' \approx t$.

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

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

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

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

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

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

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

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

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

Fig. 5. Designated-verifier signature scheme $\mathsf{RingDVS} = \mathsf{RingDVS}[\mathsf{Ring}]$ constructed from a 2-user ring signature scheme Ring.

The reduction soundly simulates the unforgeability game against $\mathsf{RingDVS}$. It simulates the signing oracle truthfully by using its own signing oracle.

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

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

Theorem 2 (Source hiding of $\mathsf{RingDVS}$). *If Ring is a (t, ϵ, n, Q_{Ch})-anonymous against key exposure 2-user ring signature for $n \geq 2$, then $\mathsf{RingDVS}$ as shown in Figure 5 is (t', ϵ, Q_{Ch})-source hiding, with $t' \approx t$.*

Proof. We reduce the source hiding of $\mathsf{RingDVS}$ to the anonymity against key exposure of Ring.

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

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

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

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

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

Implications and the inverse direction. Our construction above establishes that DVS schemes with the security properites needed for this work (i.e., unforgeability and source hiding) can be generically constructed from 2-user ring signatures that provide unforgeability w.r.t. honest-ring insider corruption and anonymity against key exposure. We note that the latter security properties are weaker than those put forward by Bender, Katz, and Morselli [6].

Hashimoto et al. have recently shown in the full version of their work [47] that it is indeed possible to construct also the reverse direction (in contrast to an earlier statement of ours). For their construction each ring member has a signer key pair and a designated verifier key pair. In the signing procedure, depending on the lexicographical order of the signer public keys either $\mathsf{DVS.Sign}$ or $\mathsf{DVS.Sim}$ is executed generating a ring signature. Verification follows analogously.

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

Lu, Au, and Zhang [61] introduce Raptor, which uses a chameleon hash function based on the NIST finalist FALCON [71], producing signatures of size approximately 5 KB for a 2-user ring. However, they argue that the best-known attack is inefficient instead of proving unforgeability and anonymity.

Yuen, Esgin, Liu, Au, and Ding [85] propose DualRing-LB (which is a lattice-based instantiation of their generic construction DualRing) with a signature size of 4.4 KB for rings of size 2. They prove anonymity (against full key exposure) of their scheme under a slightly different notion, where only the first-stage attacker has access to a signing oracle and only the second-stage attacker gets the randomness used in creating all keys (i.e., access to the secret keys).

The following two schemes use zero-knowledge proofs based on symmetric primitives, akin to the NIST alternate candidate Picnic [86]: Derler, Ramacher, and Slamanig [28] provide a scheme using NIZK proofs and accumulators. For their smallest reported ring size 2^5, signatures can have a size of 719 KB. Katz, Kolesnikov, and Wang [51] use NIZKPoK with the MPC-in-the-head paradigm. For their smallest ring size 2^7, signing takes 2 seconds and produces signatures of size 285 KB.

In terms of lattice-based constructions, a series of works [38,37,39] by Esgin et al. provide constructions relying on the hardness of M-LWE and M-SIS. The most recent candidate has a signature size of 18 KB for a 2-user ring. A construction by Lyubashevsky, Nguyen, and Seiler [65] relies on (variants of) M-LWE and M-SIS and their smallest signature for rings of size 2^5 is 16 KB. Beullens, Katsumata, and Pintore [7] introduce Falafl that also relies on M-LWE and M-SIS and produces signatures of size 29 KB in less than 100 milliseconds.

Sheikhi-Garjan, Kiliç, and Cenk [76] recently presented an isogeny-based ring signature in which signing and verifying scale in the product nq of the ring size n and isogeny security parameter q.

3 Security Model for Asynchronous Deniable Key Exchange

From a formal perspective, an asynchronous authenticated key exchange protocol is just a traditional authenticated key exchange protocol with a specific type of message flow. In particular, asynchronicity allows one party to post pre-key bundles containing long-term and possibly ephemeral public keys, provided that they can be constructed without knowing the intended partner. We will formalize security for this setting based on a Bellare–Rogaway-type model [3] with implicit authentication and (weak) forward secrecy using post-specified peers [18,53]. The model presented in this section is simplified to deal with basic Bellare–Rogaway-type security with only long-term keys; in the full version [14] 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.
- $\mathsf{st}_{\mathsf{exec}} \in \{\perp, \mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$: The status of this session's execution.
- $\mathsf{sid} \in \{0,1\}^* \cup \{\perp\}$: A session identifier defining partnering.
- $\mathsf{cid} \in \{0,1\}^* \cup \{\perp\}$: A contributive identifier, defining a preliminary form of partnering (often as a substring or prefix of the session identifier) for the case the session is not yet bound to an authenticated peer [34].
- $\mathsf{K} \in \mathcal{K}_{\mathsf{KE}} \cup \{\perp\}$: 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:

- $\mathsf{KGenLT}() \stackrel{\$}{\rightarrow} (pk, sk)$: A probabilistic long-term key generation algorithm that outputs a public-key/secret-key pair.

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

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

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

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

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

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

- $\textsc{Send}(U, i, m)$: Sends message m to session π_U^i, which corresponds to executing $\mathsf{Run}(sk_U, \vec{pk}, \pi_U^i, m)$, saving the updated session state π' as π_U^i, and returning the outgoing message m' to the adversary.

[5]Note that we call Bob the *responder* in our model despite Bob outputting the first, asynchronous key exchange message. Based on the high-level protocol interaction, we deem it more natural to call Alice, who decides to *initiate* a Signal session with Bob, the initiator (in contrast to, e.g., [79,21,80]).

– $\textsc{CorruptLTKey}(U)$: Returns party U's long-term secret key sk_U to the adversary.
– $\textsc{RevealSessKey}(U, i)$: If session π_U^i has accepted, return its session key $\pi_U^i.\mathsf{K}$ to the adversary.
– $\textsc{Test}(U, i)$: If the $\textsc{Test}$ query has been called before or session π_U^i has not accepted, then return $\perp$. Otherwise, if $b_{\mathsf{test}} = 0$, return $\pi_U^i.\mathsf{K}$, otherwise return an element of $\mathcal{K}_{\mathsf{KE}}$ chosen uniformly at random. Record $\pi^* \leftarrow \pi_U^i$.

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

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

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

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

There are several prior works giving definitions of offline deniability for key exchange [29,23,24,79,80]. Our definition differs from previous ones threefold: Firstly, the challenge oracle executes Run on behalf of the framing party, i.e., we consider semi-honest adversaries only. Secondly, the Fake algorithm (corresponding to the simulator in simulation-based definitions) has access to the receiver's secret key. Thirdly, the adversary (the judge in simulation-based settings) has access to all secret keys. This restricts the deniability to semi-honest adversaries and 1-out-of-2 (one needs a secret key of either party to create a transcript) but lifts us to the so-called big brother setting. The strong point of this deniability notion is that you get some deniability guarantees even against strong judges, who know all secret keys. This models the informal deniability requirement from the Signal specification [67, §4.4]. See the full version [14] for a more detailed discussion.

4 Security of the Core Protocol

We now show that our core protocol Π from Figure 2 achieves the security properties defined in Section 3. Key indistinguishability of Π depends on the $\mathsf{IND\text{-}CCA}$ security of the two KEMs, the unforgeability of the DVS, and the security of the KDF; deniability of Π depends on the source hiding of the DVS. Both proofs are in the standard model.

To formally capture Π in the security model of Section 3, we need to specify a few more details:

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

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

  Note that the session identifier does not include the DVS signature itself to avoid that the latter needs to be non-malleable (akin to strong unforgeability of regular signatures) [58].

4.1 Key Indistinguishability

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

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

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

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

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

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) 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^*).

[6] This is slightly stronger than what freshness condition 3 (b) demands. In the security result for our full SPQR protocol (see Section 5), this is captured more precisely.

[7] In our full SPQR protocol (see Section 5), 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 [8] against Bob, as in the Signal handshake [67, §4.6].

We bound the difference introduced by this step through a reduction to the $\mathsf{IND\text{-}CCA}$ security of the KEM_2 scheme, which simulates $\mathcal{G}_{A.1}$ truthfully except for the following changes and runs in time $t \approx t'$. It embeds the obtained challenge public key pk into the ephemeral KEM public key epk of the responder session among π^* and π^*_{p}, the challenge ciphertext c^* as c_2 of the initiator session (among π^* and π^*_{p}), and the challenge (real-or-random) key K^*_b) as K_2 into both π^* and π^*_{p}. If π^* is an initiator session, it uses its DECAPS oracle (at most once, i.e., $Q_D \leq 1$) to decrypt a potentially different ciphertext $c'_2 \neq c_2 = c^*$ received by π^*_{p}. Depending on the $\mathsf{IND\text{-}CCA}$ KEM challenge bit, the reduction perfectly simulates $\mathcal{G}_{A.1}$ or $\mathcal{G}_{A.2}$, hence $\mathsf{Adv}^{\mathcal{G}_{A.1}}_{\mathsf{ADAKE}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_2} + \mathsf{Adv}^{\mathcal{G}_{A.2}}_{\mathsf{ADAKE}}(\mathcal{A})$.

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 pseudo-randomness 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 $\mathsf{IND\text{-}CCA}$ security of the KEM_1 scheme. The reduction runs in time $t \approx t'$ and simulates $\mathcal{G}_{B.1}$ truthfully, but embeds the obtained challenge public key pk as V's public KEM key pk^{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 $\mathsf{Adv}^{\mathcal{G}_{B.2}}_{\mathsf{ADAKE}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}^{\mathcal{G}_{B.3}}_{\mathsf{ADAKE}}(\mathcal{A})$.

Here, 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 $\mathcal{G}_{C.1}$ with the following modification: Instead of generating parties' DVS keys itself, the reduction embeds the unforgeability game's challenge public keys as $pk_U = pk_S$ and $pk_V = pk_D$, and assigns the additional DVS public-secret key pairs from the unforgeability game's list $\mathcal{L}$ to the remaining parties. (Note that the reduction obtains the secret keys for the latter keys, allowing it to fully simulate those parties.) The reduction uses its signing oracle to compute signatures under $pk_U = pk_S$ (and for any peer public key pk). As U and V remain uncorrupted in this proof case, the reduction never has to answer a CorruptLTKey(U) or CorruptLTKey(V) query. In the case that π^* receives a valid DVS transcript-signature pair ($\mathsf{transcript}$, σ) that no session of U sent (and hence $\mathsf{transcript}$ was not queried to the DVS Sign oracle), the reduction outputs this pair as its forgery and wins. Therefore, $\mathsf{Adv}^{\mathcal{G}_{C.1}}_{\mathsf{ADAKE}}(\mathcal{A}) \leq \epsilon_{\mathsf{DVS}} + \mathsf{Adv}^{\mathcal{G}_{C.2}}_{\mathsf{ADAKE}}(\mathcal{A})$.

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

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 $\mathsf{IND\text{-}CCA}$ security of the KEM_2 scheme. The reduction runs in time $t \approx t'$, embeds the challenge pk and c^* into π^*'s ephemeral KEM public key, resp. π^*_{p}'s c_2 ciphertext, and uses the challenge key K^*_b in place of K_2 in both sessions. It does not need to use its Decaps oracle (i.e., $Q_D = 0$), since pk is not used in another session and we are at this point guaranteed that π^* receives π^*_{p}'s ephemeral ciphertext. (So in fact we only need $\mathsf{IND\text{-}CPA}$ security of KEM_2 here.) The reduction simulates the difference between $\mathcal{G}_{C.3}$ and $\mathcal{G}_{C.4}$, so $\mathsf{Adv}^{\mathcal{G}_{C.3}}_{\mathsf{ADAKE}}(\mathcal{A}) \leq \epsilon_{\mathsf{KEM}_2} + \mathsf{Adv}^{\mathcal{G}_{C.4}}_{\mathsf{ADAKE}}(\mathcal{A})$.

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 $(K_1, \mathsf{transcript})$. Due to π^* and π^*_{p} agreeing on the $\mathsf{transcript}$ input to KDF, the corresponding reduction only makes one query, $Q_{PRF} = 1 \leq n_s$, running in time $t \approx t'$. Simulating the game difference through this reduction, we get $\mathsf{Adv}^{\mathcal{G}_{C.4}}_{\mathsf{ADAKE}}(\mathcal{A}) \leq \epsilon_{\mathsf{KDF}} + \mathsf{Adv}^{\mathcal{G}_{C.5}}_{\mathsf{ADAKE}}(\mathcal{A})$.

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

4.2 Deniability

Observe that we use a different deniability notion compared to prior works as discussed in Section 3. A more thorough discussion of the different deniability notions can be found in the full version [14]. In consequence, we can forgo the strong knowledge assumptions that both [81,46] used to prove deniability of X3DH and their own construction, respectively. We conjecture that either construction can likewise be shown to be deniable wrt. our definition without strong knowledge assumptions.

Theorem 4 (Deniability of Π). *Let* $\mathsf{DVS} = (\mathsf{SKGen}, \mathsf{VKGen}, \mathsf{Sign}, \mathsf{Vrfy}, \mathsf{Sim})$ *be a* $(t, \epsilon_{\mathsf{srchid}}, Q_{Ch})$*-source hiding DVS scheme. Then the asynchronous DAKE* *protocol Π from Figure 2 provides deniability (as defined in Section 3) in that the advantage ϵ' of any adversary $\mathcal{A}$ running in time $t' \approx t$ and making up to Q_{Ch} challenge queries is upper bounded as $\epsilon' \leq n_p^2 \cdot \epsilon_{\mathsf{srchid}}$, where n_p is the number of parties.*

Proof. The proof follows by a standard hybrid argument. Let $\mathcal{A}$ be a successful adversary against deniability of Π, then we can construct a reduction $\mathcal{B}$ against the source hiding property of DVS. Observe that $\mathcal{B}$ computes for each of the n_p parties a long-term key pair. It randomly guesses the identifiers of two parties $\mathsf{iid}^*, \mathsf{rid}^* \in [n_p]$ for which $\mathcal{A}$ can distinguish between Run and Fake. Let a number $i \in [n_p^2]$ uniquely denote two independent values $\mathsf{iid}, \mathsf{rid}$ in a query (e.g., encoded as $(\mathsf{iid} - 1) \cdot n_p + \mathsf{rid}$) and let $i^* \in [n_p^2]$ denote the specific guess $\mathsf{iid}^*, \mathsf{rid}^*$ of $\mathcal{B}$. For party iid^*, $\mathcal{B}$ replaces the sampled long-term key with its challenge key pair (pk_S, sk_S) and similarly it replaces 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 $H_0, \dots, H_{n_p^2}$ with H_i being the hybrid that answers all challenge queries for indices $1, \dots, i$ with Run and the challenge queries for indices $i + 1, \dots, n_p^2$ 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 $\epsilon' \leq n_p^2 \cdot \epsilon_{\mathsf{srchid}}$. □

5 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 6.

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

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

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

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

Alice | **Signal Server** | **Bob**

```
Initiator Registration
$(pk_A, sk_A) \leftarrow\!\$\ \mathsf{KGenLT}()$
```

```
Responder Registration
$(pk_B, sk_B) \leftarrow\!\$\ \mathsf{KGenLT}()$
$(sspk_B, sssk_B) \leftarrow\!\$\ \mathsf{KGenSS}()$

Responder Ephemeral Key Generation
$(epk_B, esk_B) \leftarrow\!\$\ \mathsf{KGenEP}()$
```

Send Pre-Key Bundle to Initiator

$$\xleftarrow{B, pk_B, sspk_B, epk_B}$$

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

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

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

$K \leftarrow \mathsf{KDF}(\mathsf{ms}, sid)$

return $(K, sid, \mathsf{accepted}, \varepsilon)$
```

$$\xrightarrow{m = (A, pk_A, n, c_1, c_2, c_3, \sigma)}$$

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

Fig. 6. 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.

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 key-ciphertext pair (K_3, c_3) against Bob's ephemeral KEM public key epk_B only if the latter is present.

Maximal-exposure security Signal aims for very strong security guarantees, considering beyond long-term and session key compromise and also compromise of semi-static and ephemeral keys (via the randomness of sessions) [17,56,21]. We model this in an accordingly strong key exchange model and prove that SPQR achieves equivalent security in the post-quantum setting as Signal does in the classical setting in the full version [14]. 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 [42,54]. Twisted PRFs can be generically instantiated from regular PRFs (see full version [14]) and yield output indistinguishable from random as long as a session's long-term secret *or* randomness is uncompromised.

Our formal security results establishing key indistinguishability and deniability for SPQR are as follows; see the full version [14] for the game-based formalizations of key indistinguishability and deniability as well as for the full proof details expanding beyond the core ideas from Section 4.

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

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

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

Then the SPQR *protocol with randomness space* $\mathcal{R}_{\mathsf{KE}} = \{0,1\}^\lambda \times \mathcal{R}_{\mathsf{tPRF}} \times \mathcal{R}_{\mathsf{KEM}_3}$ *as shown in Figure 6 provides* $(t', \epsilon', (Q_{Snd}, Q_{CorrLT}, Q_{CorrSS}, Q_{RevR},$

Q_{RevSK}))*–key indistinguishability (formalized in the full version) 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.*

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

6 Discussion and Limitations

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

The key ingredient in our approach for achieving post-quantum asynchronous DAKE is a post-quantum designated verifier signature scheme. While there are several lattice-based DVS schemes in the literature as described in Section 2.1, we believe that their security merits further scrutiny before adoption. In the meantime, we propose instantiations via 2-user ring signatures, for which we discussed post-quantum candidates in Section 2.2.

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

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

SPQR is solely a replacement for the initial handshake (X3DH). A fully post-quantum Signal would require quantum-resistance in the ratcheting and message encryption; fortunately there are several generic treatments of ratcheting [5,70,1].

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

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

Cryptographic deniability should be treated with caution. How cryptographers understand deniability may be different from how a judge in a legal system understands it [79]. 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 further confirm deniability at all protocol levels, and that deniability of individual components composes appropriately. Despite all these subtleties, steps toward deniability are helpful, as Unger and Goldberg write [79]: "we should strive to design deniable protocols to avoid unintentionally incriminating users."

Acknowledgements

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

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

References

1. J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In *EUROCRYPT 2019, Part I*, pages 129–158, 2019.
2. R. Azarderakhsh, D. Jao, and C. Leonardi. Post-quantum static-static key agreement using multiple protocol instances. In *SAC 2017*, pages 45–63, 2017.
3. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO'93*, pages 232–249, 1994.
4. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT'94*, pages 92–111, 1995.
5. M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In *CRYPTO 2017, Part III*, pages 619–650, 2017.
6. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, 2009.
7. W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYPT 2020, Part II*, pages 464–492, 2020.
8. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *6th IMA International Conference on Cryptography and Coding*, pages 30–45, 1997.

9. D. Boneh, D. Glass, D. Krashen, K. Lauter, S. Sharif, A. Silverberg, M. Tibouchi, and M. Zhandry. Multiparty non-interactive key exchange and more from isogenies on elliptic curves. *Journal of Mathematical Cryptology*, 14(1):5–14, 2020.
10. X. Bonnetain and A. Schrottenloher. Quantum security analysis of CSIDH. In *EUROCRYPT 2020, Part II*, pages 493–522, 2020.
11. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*, pages 353–367, 2018. https://cryptojedi.org/papers/#kyber.
12. C. Boyd, Y. Cliff, J. M. Gonzalez Nieto, and K. G. Paterson. One-round key exchange in the standard model. *IJACT*, 1:181–199, 2009.
13. C. Boyd and K. Gellert. A Modern View on Forward Security. *The Computer Journal*, 64(4):639–652, 2020.
14. J. Brendel, R. Fiedler, F. Günther, C. Janson, and D. Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. Cryptology ePrint Archive, Report 2021/769, 2021. https://eprint.iacr.org/2021/769.
15. J. Brendel, M. Fischlin, F. Günther, C. Janson, and D. Stebila. Towards post-quantum security for Signal's X3DH handshake. In *27th Conference on Selected Areas in Cryptography (SAC)*, 2020.
16. J. Cai, H. Jiang, P. Zhang, Z. Zheng, H. Wang, G. Lü, and Q. Xu. Id-based strong designated verifier signature over ∇-sis assumption. *Secur. Commun. Networks*, 2019:9678095:1–9678095:8, 2019.
17. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001*, pages 453–474, 2001.
18. R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *CRYPTO 2002*, pages 143–161, 2002. https://eprint.iacr.org/2002/120/.
19. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, 2012.
20. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: An efficient post-quantum commutative group action. In *ASIACRYPT 2018, Part III*, pages 395–427, 2018.
21. K. Cohn-Gordon, C. J. F. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the Signal messaging protocol. In *IEEE European Symposium on Security and Privacy, EuroS&P 2017*, pages 451–466, 2017.
22. K. Cohn-Gordon, C. J. F. Cremers, and L. Garratt. On post-compromise security. In *CSF 2016 Computer Security Foundations Symposium*, pages 164–178, 2016.
23. C. Cremers and M. Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. https://eprint.iacr.org/2011/300.
24. Ö. Dagdelen, M. Fischlin, T. Gagliardoni, G. A. Marson, A. Mittelbach, and C. Onete. A cryptographic analysis of OPACITY - (extended abstract). In *ESORICS 2013*, pages 345–362, 2013.
25. I. Damgård, H. Haagh, R. Mercer, A. Nitulescu, C. Orlandi, and S. Yakoubov. Stronger security and constructions of multi-designated verifier signatures. In *TCC 2020, Part II*, pages 229–260, 2020.
26. B. de Kock, K. Gjøsteen, and M. Veroni. Practical isogeny-based key-exchange with optimal tightness. In *27th Conference on Selected Areas in Cryptography (SAC)*, 2020.
27. C. D. de Saint Guilhem, N. P. Smart, and B. Warinschi. Generic forward-secure key agreement without signatures. In *ISC 2017*, pages 114–133, 2017.

28. D. Derler, S. Ramacher, and D. Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 419–440, 2018.
29. M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. In *ACM CCS 2006*, pages 400–409, 2006.
30. S. Dobson and S. D. Galbraith. Post-quantum signal key agreement with SIDH. Cryptology ePrint Archive, Report 2021/1187, 2021. https://eprint.iacr.org/2021/1187.
31. S. Dobson, S. D. Galbraith, J. T. LeGrow, Y. B. Ti, and L. Zobernig. An adaptive attack on 2-sidh. *Int. J. Comput. Math. Comput. Syst. Theory*, 5(4):282–299, 2020.
32. S. Dobson, T. Li, and L. Zobernig. A note on a static SIDH protocol. Cryptology ePrint Archive, Report 2019/1244, 2019. https://eprint.iacr.org/2019/1244.
33. Y. Dodis, J. Katz, A. Smith, and S. Walfish. Composability and on-line deniability of authentication. In *TCC 2009*, pages 146–162, 2009.
34. B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *ACM CCS 2015*, pages 1197–1210, 2015.
35. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014, Part I*, pages 335–352, 2014.
36. I. Duits. The post-quantum Signal protocol: Secure chat in a quantum world. Master's thesis, University of Twente, 2019.
37. M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO 2019, Part I*, pages 115–146, 2019.
38. M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS 19*, pages 67–88, 2019.
39. M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In *ACM CCS 2019*, pages 567–584, 2019.
40. M. Fischlin and S. Mazaheri. 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*, pages 55–64, 2015.
41. E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In *PKC 2013*, pages 254–271, 2013.
42. A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. In *PKC 2012*, pages 467–484, 2012.
43. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO'99*, pages 537–554, 1999.
44. S. D. Galbraith, C. Petit, B. Shani, and Y. B. Ti. On the security of supersingular isogeny cryptosystems. In *ASIACRYPT 2016, Part I*, pages 63–91, 2016.
45. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206, 2008.
46. K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. In *PKC 2021, Part II*, pages 410–440, 2021.
47. K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An efficient and generic construction for signal's handshake (X3DH): post-quantum, state leakage

secure, and deniable. Cryptology ePrint Archive, Report 2021/564, 2021. https://eprint.iacr.org/2021/616.

48. J. Herranz. *Some digital signature schemes with collective signers*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, 2005.
49. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT'96*, pages 143–154, 1996.
50. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 19–34, 2011.
51. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS 2018*, pages 525–537, 2018.
52. T. Kawashima, K. Takashima, Y. Aikawa, and T. Takagi. An efficient authenticated key exchange from random self-reducibility on CSIDH. In *ICISC 20*, pages 58–84, 2020.
53. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO 2005*, pages 546–566, 2005.
54. K. Kurosawa and J. Furukawa. 2-pass key exchange protocols from CPA-secure KEM. In *CT-RSA 2014*, pages 385–401, 2014.
55. F. Laguillaumie and D. Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *SCN 04*, pages 105–119, 2005.
56. B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec 2007*, pages 1–16, 2007.
57. B. Li, Y. Liu, and S. Yang. Lattice-based universal designated verifier signatures. In *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, pages 329–334, 2018.
58. Y. Li and S. Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In *ACM CCS 2017*, pages 1343–1360, 2017.
59. Y. Li, W. Susilo, Y. Mu, and D. Pei. 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*, pages 1191–1200, 2007.
60. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT 2016, Part II*, pages 1–31, 2016.
61. X. Lu, M. H. Au, and Z. Zhang. Raptor: A practical lattice-based (linkable) ring signature. In *ACNS 19*, pages 110–130, 2019.
62. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, pages 598–616, 2009.
63. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, pages 738–755, 2012.
64. V. Lyubashevsky and G. Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT 2017, Part I*, pages 293–323, 2017.
65. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions. Cryptology ePrint Archive, Report 2021/564, 2021. https://eprint.iacr.org/2021/564.
66. M. Marlinspike and T. Perrin. The double ratchet algorithm, November 2016.
67. M. Marlinspike and T. Perrin. The X3DH key agreement protocol, November 2016.

68. G. Noh and I. R. Jeong. Strong designated verifier signature scheme from lattices in the standard model. *Security Comm. Networks*, 9:6202–6214, 2017.
69. C. Peikert. He gives C-sieves on the CSIDH. In *EUROCRYPT 2020, Part II*, pages 463–492, 2020.
70. B. Poettering and P. Rösler. Towards bidirectional ratcheted key exchange. In *CRYPTO 2018, Part I*, pages 3–32, 2018.
71. T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.
72. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93, 2005.
73. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, pages 552–565, 2001.
74. S. Saeednia, S. Kremer, and O. Markowitch. An efficient strong designated verifier signature scheme. In *ICISC 03*, pages 40–54, 2004.
75. P. Schwabe, D. Stebila, and T. Wiggers. Post-quantum TLS without handshake signatures. In *ACM CCS 2020*, pages 1461–1480, 2020.
76. M. Sheikhi-Garjan, N. G. O. Kiliç, and M. Cenk. A supersingular isogeny-based ring signature. Cryptology ePrint Archive, Report 2021/1318, 2021. https://eprint.iacr.org/2021/1318.
77. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *ASIACRYPT 2003*, pages 523–542, 2003.
78. X. Sun, H. Tian, and Y. Wang. Toward quantum-resistant strong designated verifier signature from isogenies. In *4th International Conference on Intelligent Networking and Collaborative Systems*, pages 292–296, 2012.
79. N. Unger and I. Goldberg. Deniable key exchanges for secure messaging. In *ACM CCS 2015*, pages 1211–1223, 2015.
80. N. Unger and I. Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *PoPETs*, 2018(1):21–66, 2018.
81. N. Vatandas, R. Gennaro, B. Ithurburn, and H. Krawczyk. On the cryptographic deniability of the Signal protocol. In *ACNS 20, Part II*, pages 188–209, 2020.
82. F. Wang, Y. Hu, and B. Wang. Lattice-based strong designate verifier signature and its applications. *Malaysian Journal of Computer Science*, 25:11–22, 2012.
83. F. Wang, Y. Hu, and B. Wang. Identity-based strong designate verifier signature over lattices. *The Journal of China Universities of Post and Telecommunications*, 21:52–60, 2014.
84. B. Yang, Y. Yu, and Y. Sun. A novel construction of SDVS with secure disavowability. *Clust. Comput.*, 16(4):807–815, 2013.
85. T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding. DualRing: Generic construction of ring signatures with efficient instantiations. In *CRYPTO 2021, Part I*, pages 251–281, 2021.
86. G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, V. Kolesnikov, and D. Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.
87. Y. Zhang, Q. Liu, C. Tang, and H. Tian. A lattice-based designated verifier signature for cloud computing. *International Journal of High Performance Computing and Networking*, 8:135–143, 2015.