A Deniability Analysis of Signal's Initial Handshake PQXDH

Rune Fiedler Christian Janson

Cryptoplexity, Technische Universität Darmstadt, Germany
{firstname.lastname}@cryptoplexity.de

Version 1.1*, June 2025

Abstract. Many use messaging apps such as Signal to exercise their right to private communication. To cope with the advent of quantum computing, Signal employs a new initial handshake protocol called PQXDH for post-quantum confidentiality, yet keeps guarantees of authenticity and deniability classical. Compared to its predecessor X3DH, PQXDH includes a KEM encapsulation and a signature on the ephemeral key. In this work we show that PQXDH does not meet the same deniability guarantees as X3DH due to the signature on the ephemeral key. Our analysis relies on plaintext awareness of the KEM, which Signal's implementation of PQXDH does not provide. As for X3DH, both parties (initiator and responder) obtain different deniability guarantees due to the asymmetry of the protocol.

For our analysis of PQXDH, we introduce a new model for deniability of key exchange that allows a more fine-grained analysis. Our deniability model picks up on the ideas of prior work and facilitates new combinations of deniability notions, such as deniability against malicious adversaries in the big brother model, i.e. where the distinguisher knows all secret keys. Our model may be of independent interest.

1 Introduction

Nowadays, messaging applications are widely adopted: Billions of people use messaging apps such as Signal and WhatsApp.[1] In order to provide confidentiality of the users' communication, both apps offer end-to-end encrypted messaging by using the Signal protocol. The Signal protocol consists of an initial handshake, which allows two parties to derive a shared session key, and a ratcheting mechanism [MP16a], which provides forward secrecy and post-compromise security. Until 2023, Signal used the X3DH protocol [MP16b] for the initial handshake. To resist the threat of "harvest now decrypt later attacks" [KS23] posed by future quantum computers, Signal replaced X3DH with the PQXDH protocol; pursuing post-quantum *confidentiality* (without addressing quantum attackers against authenticity or deniability). Similarly, this paper does not consider deniability against quantum adversaries.

Deniability is a subtle privacy property that allows a party to plausibly deny an action. Consider the following example in a scenario without computers and smartphones present, where Alice and Bob have a conversation. Later, Alice tells Charlie she talked to Bob, and Bob tells Charlie he never talked to Alice. Charlie does not know whose word to trust; hence, Bob can *deny* his conversation with Alice since their conversation did not leave any evidence behind. For sensitive topics such as medical conditions, sexuality, or coordinating a protest against an oppressive regime, this type of privacy guarantee can be not just helpful but even vital.

*An extended abstract of this paper appears in the Proceedings on Privacy Enhancing Technologies (PoPETS) 2024(4), 978-928. DOI: https://doi.org/10.56553/popets-2024-0148. This is the full version. Please see Appendix C for an overview of major changes from earlier versions of this work.

[1] https://en.wikipedia.org/wiki/WhatsApp#User_statistics

Conversations in the digital world entail sending data over the internet. Moving our previous example to conversing via messaging, Alice and Bob exchange messages as specified by the messaging protocol, resulting in a *protocol transcript*. This transcript may convince Charlie that Alice and Bob *did* converse and *what* they conversed about. However, if Alice and Bob use a deniable protocol, then either party can deny involvement in a protocol run.

A deniable key exchange is necessary (but not sufficient, as pointed out in [CCH23]) to build deniable messaging. Signal desires deniability[2] and, therefore, a deniable initial handshake, i.e. previously X3DH and now PQXDH [MP16b, KS23]. Before X3DH, Signal used a derivative of Off-the-Record Messaging (OTR) [OTR20], which also employs a deniable key exchange.

For a key exchange protocol to be deniable, the transcript must not prove Bob's involvement in the conversation. In particular, there must exist a Fake algorithm that *simulates* Bob's involvement. Hence, Alice acting as the *adversary* can simulate a transcript on her own by using the Fake algorithm. Charlie acting as *distinguisher* must not be able to tell the simulated transcript apart from a real transcript. Honest parties would likely use the freshly established session key in some subsequent protocol. Hence, the Fake algorithm must simulate the session key indistinguishably as well.

Several deniability definitions were produced by prior works [DGK06, CF11, HKKP22, DFG+13, BFG+22, BFG+21, YZ13, JS08, Jia14, JCL+22]; each definition follows the outline in the previous paragraph. Additionally, [DGK06] introduces *auxiliary information* (e.g. valid protocol transcripts anybody could have observed a priori) that all algorithms (adversary, their equivalent of the Fake algorithm, and the distinguisher) can access. The definitions from the literature are incomparable, vary in the precise guarantee they capture and which formalisms they use. In consequence, capturing several deniability notions may require using several formalisms. Instead, we propose a new model for deniable key exchange, which integrates cherry-picked ideas from prior works: Our model is parameterized in the auxiliary info (*Are some protocol transcripts available?*), the power of the adversary (*Does the adversary follow the protocol?*), the power of the Fake algorithm (*How much help do you need to simulate a transcript?*), the power of the distinguisher (*Is the distinguisher a* big brother *who can force everybody to give up their secret keys? Does it learn the auxiliary info?*). Our new model facilitates new combinations of ideas, e.g. combining malicious adversaries [DGK06] with a big brother distinguisher [BFG+22], thereby augmenting the design space for deniability.

Starting a conversation in Signal. Observe that the Signal protocol allows starting a conversation with offline users through the asynchronicity of the initial handshake: Bob prepares a so-called pre-key bundle and uploads it to Signal's key server. To initiate a conversation with Bob, Alice retrieves his pre-key bundle from the key server and sends a message to Bob. (See Figure 3 on page 17 for an algorithmic description of X3DH and PQXDH.)

The X3DH protocol uses the Diffie–Hellman (DH) key exchange, where each user has DH key pairs of different lifetimes (long-term, semi-static, and ephemeral). Bob's pre-key bundle contains his public keys (of all three lifetimes) and a signature certifying his semi-static key under the long-term key. Upon retrieving Bob's pre-key bundle, Alice verifies the signature and generates a fresh ephemeral DH key pair. She computes four (or in case Bob's ephemeral keys run out, only three) DH shared secrets by combining DH keys of different lifetimes, uses the resulting session key to encrypt her first user message with an AEAD scheme, and sends her ephemeral public key and the AEAD ciphertext to Bob. Bob computes the session key in the same manner, decrypts the AEAD ciphertext, and aborts if decryption fails.

The new PQXDH protocol does the same operations as X3DH and adds a KEM encapsulation to achieve post-quantum confidentiality: Bob includes an ephemeral KEM public key in his pre-key bundle and a signature for his KEM key (again under Bob's long-term key). If the ephemeral KEM keys run out, a

[2]https://signal.org/blog/simplifying-otr-deniability/

semi-static KEM key and the corresponding signature are instead included in the pre-key bundle. Alice verifies the signatures on the semi-static DH key and the KEM key and encapsulates against Bob's KEM key. She derives the session key from the DH shared secrets and the KEM shared secret, and encrypts her first user message with the AEAD scheme under the session key. She sends the resulting (KEM and AEAD) ciphertexts and the public key of her freshly generated ephemeral DH key pair to Bob. Bob decapsulates the KEM ciphertext. Bob computes the session key in the same manner, decrypts the AEAD ciphertext, and aborts if decryption fails.

In order to show deniability, we need to specify a Fake algorithm that simulates protocol messages and the session key indistinguishably. To simulate Bob's message, the Fake algorithm needs to produce signatures under Bob's long-term key without Bob's long-term secret key. For signatures on a semi-static key, the Fake algorithm can reuse any signature from the auxiliary info (in practice, this can be a transcript that the attacker learned from eavesdropping or a pre-key bundle) or from an independent session with Bob. For PQXDH, the Fake algorithm needs a signature on an ephemeral key, i.e. a signature that the distinguisher is not aware of. In Section 4 we call these *private signatures*.

To simulate the session key against malicious adversaries, the Fake algorithm requires all DH shared secrets – and for PQXDH additionally the KEM shared secret. In particular, the Fake algorithm does not know the DH secret keys (some are known to the honest party and some to the adversary); following [VGIK20], we extract the DH shared secrets from the adversary under a knowledge of DH assumption (see Section 2.3 for details; assume if a party produces a DH public key, this party can produce a DH shared secret with this public key, or nobody can). On Alice's side, the Fake algorithm learns the KEM shared secret by encapsulating; on Bob's side the corresponding KEM secret key is not available for decapsulation. Hence, the Fake algorithm extracts the KEM shared secret from the adversary under the plaintext awareness assumption (see Definition 2.7 for details; assume anybody who produces an encapsulation knows the encapsulated secret).

1.1 Related Work

Built on the literature for deniable authentication [DNS98, Kat03, Pas03, DGK05, DG05], Di Raimondo, Gennaro, and Krawczyk [DGK06] have initiated the formal study of deniable key exchange: A distinguisher cannot conclude whether some evidence stems from an actual protocol execution between two parties or whether it was produced by some other means. Many papers have considered deniability for key exchange [Kra96, HKK$^+$02, MP02, BMP03, BMP04, LLPM07, YZ10, JS08, DKSW09, YZ13, Jia14, BMS20, JCL$^+$22]. Deniability was also named *repudiability*, mostly in the context of Off-the-Record (OTR) messaging [BGB04, UDB$^+$15, UG15, UG18]. Next, we revisit some more prominent work on deniability notions in general and their application to Signal's initial handshake.

1.1.1 (Non-)Interactive distinguisher

Deniable authentication [DNS98, DG05] and some forms of deniable key exchange [DGK06] are defined with respect to a distinguisher[3] that is presented with evidence after the fact, i.e. an *offline distinguisher*. Dodis, Katz, Smith, and Walfish [DKSW09] have introduced the notion of an interactive, *online distinguisher*. Unger and Goldberg [UG15] argue that online deniability is not achievable for an asynchronous key exchange protocol (i.e. one party uploads pre-keys to a central server) with forward secrecy.[4] Hence, in this paper we write *deniability* and refer to offline deniability, unless explicitly stated otherwise.

[3]Prior work uses the term *judge*, which we avoid for ambiguity with the legal profession.

[4]They call this assumed impossibility "Iron Triangle".

1.1.2 Deniable Key Exchange

Di Raimondo et al. [DGK06] have introduced *concurrent deniability* and *partial deniability* of key exchange protocols, following the simulation-based approach of deniable authentication. They further share the observation of Pass [Pas03] that a simulator for deniability needs to be implementable; in particular, the simulator needs to refrain from rewinding the adversary, reprogramming a random oracle, and similar techniques for simulation in thought experiments. Their deniability definition was since used in [VGIK20] and slightly adapted in [CF11, HKKP22]. Dagdelen, Fischlin, Gagliardoni, Marson, Mittelbach, and Onete [DFG$^+$13] have introduced the first game-based definition for a weak version of deniability of key exchange, named *outsider deniability.* Dodis et al. [DKSW09] defined online deniability for key exchange in the Generalized UC framework, which was further refined by Unger and Goldberg [UG15, UG18]. Several works [CF11, JS08, Jia14, JCL$^+$22] consider (adaptive) corruptions in the sense that the attacker, the simulator, and possibly the distinguisher learn the corrupted party's secrets. Appendix A discusses the relation between notions of prior work and our new model in more detail.

1.1.3 Signal's initial handshake

Until recently, Signal employed X3DH [MP16b] as initial handshake protocol, whose security can be based on the GapDH assumption [CCD$^+$17]. Hashimoto, Katsumata, Kwiatowski, and Prest [HKKP22] have proposed a generic post-quantum construction for Signal's initial handshake based on KEMs for confidentiality and either signatures (named SC-AKE), or ring signatures (SC-DAKE), or ring signatures and NIZKs (SC-DAKE') for authentication and increasing levels of deniability. In concurrent work Brendel, Fiedler, Günther, Janson, and Stebila [BFG$^+$22] have proposed a similar generic construction named SPQR based on KEMs and designated verifier signatures (DVS). Dobson and Galbraith [DG22] adopt the X3DH handshake to isogenies, resulting in a protocol named SI-X3DH, which was broken by the SIDH attack [CD23, MMP$^+$23, Rob23]. Collins, Huguenin-Dumittan, Nguyen, Rolin, and Vaudenay [CHN$^+$24] have proposed K-Waay, which is based on a primitive named split-KEMs that was introduced by [BFG$^+$20]. In 2023, Signal deployed PQXDH [KS23], which uses a more conservative hybrid approach: It combines the classically secure X3DH handshake with a post-quantum secure KEM. This modification aims to add confidentiality against future quantum attackers but is not concerned with quantum attacks against authentication or deniability. Bhargavan, Jacomme, Kiefer, and Schmidt [BJKS23, BJK23] formally verified PQXDH. Fiedler and Günther [FG24] provide a reductionist key indistinguishability analysis of PQXDH, adding coverage for maximum exposure security to the analysis of [BJKS23, BJK23].

1.1.4 Deniability of Signal's initial handshake

Vatandas, Gennaro, Ithurburn, and Krawczyk [VGIK20] have shown that Signal's initial handshake X3DH is deniable in the model of [DGK06] under a novel knowledge of DH assumption. Dobson and Galbraith [DG22] sketch a similar deniability argument for SI-X3DH. Hashimoto et al. [HKKP22] show deniability of SC-DAKE and SC-DAKE' under the model of [DGK06] against semi-honest adversaries and malicious adversaries, respectively, assuming plaintext awareness of the KEM and a related knowledge assumption. Brendel et al. [BFG$^+$22, BFG$^+$21] show that SPQR is deniable under a new deniability notion (1-out-of-2 deniability against semi-honest adversaries in the big brother model), which they model after the intuition of Signal's specification [MP16b]. Collins et al. [CHN$^+$24] show deniability of K-Waay under a new deniability notion close to the one of Brendel et al. [BFG$^+$22, BFG$^+$21].

1.1.5 Deniable Secure Messaging

A protocol combining a key exchange with a subsequent exchange of user messages is called *Secure Messaging*. Unger, Dechang, Bonneau, Fahl, Perl, Goldberg, and Smith [UDB+15] cover several forms of deniability for Secure Messaging; among them the distinction between *participation deniability* (a user can deny participation in a specific session) and *message deniability* (a user can deny having written a particular (user) message). Cremers and Zhao [CZ24] extended the game-based deniability model of [BFG+22] for key exchange to cover (extended) secure messaging. Reitinger, Malkin, Akgul, Mazurek, and Miers [RMA+23], as well as Yadav, Gosain, and Seamons [YGS23] study the social implications of cryptographic deniability for secure messaging.

1.2 Contributions

To capture the precise deniability guarantees of PQXDH, we introduce a new, more expressive deniability model for key exchange protocols (cp. Section 3) that facilitates combinations of previous deniability notions. Our new, game-based deniability model is designed for, but not limited to, asynchronous key exchange. Our model is parameterized to cover deniability notions for a multitude of scenarios: circumstantial knowledge (auxiliary info aux), several adversarial capabilities (malicious, semi-honest, and combinations), how much help the Fake algorithm needs to simulate (partial deniability, or only the peer can simulate, i.e. 1-out-of-2 deniability), and distinguisher capabilities (big brother model, knowledge of auxiliary info). The flexibility of our model allows for easy comparisons of deniability notions and easy adaptability of a proof of deniability to another notion (simply by changing the four aforementioned parameters while keeping the structure of the game, and, hence, the proof).

Furthermore, we provide the first deniability analysis of Signal's new initial handshake PQXDH (cp. Section 4). We show that—due to the signed ephemeral key on Bob's side—the deniability guarantees of PQXDH fall short of those of X3DH. Assuming plaintext awareness of the KEM, which PQXDH adds onto X3DH, and knowledge of a signature hidden from the distinguisher, PQXDH retains the deniability guarantees of X3DH. Table 2 gives the full results.

2 Preliminaries

In this section we review necessary preliminaries.

2.1 Notation

We denote the empty list by [], append *element* to *list* by $list \xleftarrow{+} element$, and append the content of $list2$ to $list1$ by $list1 \xleftarrow{+} list2$. In abuse of notation, we allow combining appends, e.g. $(l1, l2) \xleftarrow{+} (l3, l4)$ returns $((l1 \xleftarrow{+} l3), (l2 \xleftarrow{+} l4))$, and allow the same syntax for adding to sets. We denote by $a, b, c \in S$ that S contains a, b, c. Given a map a, we refer to its entry at position i with $a[i]$.

Two distributions are said to be $(t_\mathcal{D}, \epsilon_\mathcal{D})$-indistinguishable if no algorithm running in time $t_\mathcal{D}$ has an advantage better than $\epsilon_\mathcal{D}$ in distinguishing the two distributions. We use $y \leftarrow\!\!\$\, A(x)$ to denote the random output y of algorithm A for input x, where the probability is over A's internal randomness. We use the arrow $\leftarrow$ for any assignment statements. We write $\mathcal{E}_{\mathcal{A};(Q,r)}$ for an extractor $\mathcal{E}$ depending on the adversary $\mathcal{A}$, where $\mathcal{A}$ has received oracle responses described by Q and r from its oracles.

2.2 Signatures

We briefly review the syntax for digital signature schemes.

Definition 2.1 (Signature schemes)**.** *A* digital signature scheme *is a triple of algorithms* $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *with associated message space* $\mathcal{M}_{\mathsf{SIG}}$*, defined as follows:*

- $\mathsf{KGen}() \overset{\$}{\rightarrow} (\mathsf{pk}, \mathsf{sk})$*: This probabilistic algorithm returns a key pair* $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Sign}(\mathsf{sk}, m) \overset{\$}{\rightarrow} \sigma$*: On input a secret key* sk *and a message* $m \in \mathcal{M}_{\mathsf{SIG}}$*, this probabilistic algorithm returns a signature* σ*;*
- $\mathsf{Vf}(\mathsf{pk}, m, \sigma) \rightarrow d$*: On input a public verification key* pk*, a message* m*, and a candidate signature* σ*, this deterministic algorithm returns a bit* $d \in \{0, 1\}$*. If* $d = 1$ *we say that the signature is valid, otherwise not.*

We say that a digital signature scheme SIG is *correct* if, for every $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}()$, every $m \in \mathcal{M}_{\mathsf{SIG}}$, and random $\sigma \leftarrow\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$, it holds that $\Pr[\mathsf{Vf}(\mathsf{pk}, m, \sigma) = 1] = 1$.

We consider the security notion unforgeability under chosen message attacks for digital signature schemes.

Definition 2.2 (Unforgeability for digital signature Schemes)**.** *A digital signature scheme* SIG *is* $(q_{\mathsf{SIG}}, t_{\mathsf{SIG}}, \epsilon_{\mathsf{SIG}})$*-unforgeable if, for every adversary* $\mathcal{A}$ *running in time* t_{SIG} *and making at most* q_{SIG} *queries to the* SIGN *oracle, it holds that* $\Pr[\mathcal{G}^{uf}_{\mathsf{SIG}}(\mathcal{A}) = 1] \leq \epsilon_{\mathsf{SIG}}$*, where* $\mathcal{G}^{uf}_{\mathsf{SIG}}(\mathcal{A})$ *is defined as:*

```
$\mathcal{G}^{uf}_{\mathsf{SIG}}(\mathcal{A})$:
1 $Q \leftarrow \emptyset$
2 $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{SIG.KGen}()$
3 $(m^*, \sigma^*) \leftarrow\!\$\ \mathcal{A}^{\text{SIGN}}(\mathsf{pk})$
4 return $\mathsf{Vf}(\mathsf{pk}, m^*, \sigma^*) \wedge m^* \notin Q$

SIGN$(m)$:
5 $\sigma \leftarrow\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$
6 $Q \leftarrow Q \cup \{m\}$
7 return $\sigma$
```

2.3 Diffie–Hellman Key Exchange

We briefly review the syntax for the Diffie–Hellman key exchange.

Definition 2.3 (Diffie–Hellman Key Exchange)**.** *A* Diffie–Hellman Key Exchange (DH) scheme *is a tuple of algorithms* $(\mathsf{KGen}, \mathsf{DH})$*, defined as follows:*

- $\mathsf{KGen}() \overset{\$}{\rightarrow} (\mathsf{pk}, \mathsf{sk})$*: This probabilistic algorithm returns a key pair* $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{DH}(\mathsf{pk}_A, \mathsf{sk}_B) \rightarrow \mathsf{DH}_{AB}$*: On input a public key* pk_A *and a secret key* sk_B*, this deterministic algorithm returns the shared DH secret* DH_{AB}*.*

We say that a DH key exchange DH is *correct* if, for every $(\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B) \leftarrow\!\$\ \mathsf{KGen}()$, it holds that $\Pr[\mathsf{DH}(\mathsf{pk}_A, \mathsf{sk}_B) = \mathsf{DH}(\mathsf{pk}_B, \mathsf{sk}_A)] = 1$. For notational convenience we allow the arguments to be in arbitrary order. Hence, $\mathsf{DH}(\mathsf{pk}_A, \mathsf{sk}_B) = \mathsf{DH}(\mathsf{sk}_A, \mathsf{pk}_B)$.

The following assumption differs slightly from the EKDH assumption in Definition 2.5: Here, $\mathcal{A}$ is first given two DH public keys and outputs a third DH public key. Then, the extractor tries to extract the two DH shared secrets between the output public key and each of the input public keys.

Definition 2.4 (K2DH Assumption [VGIK20])**.** *Let* G *be a cyclic group with generator* g *and* $\mathsf{AuxPrep}$ *a sampler for auxiliary inputs. Let* $\mathcal{A}$ *be any algorithm running in time* $t_{\mathcal{A}}$ *which runs on input* (U, W, aux) *where* $U, W \in G$*, and* $\mathsf{aux} \leftarrow\!\$\ \mathsf{AuxPrep}$*, and outputs* $Z \in G$*; we denote with* $Z = \mathcal{A}(U, W, \mathsf{aux}; r)$ *the output of running* $\mathcal{A}$ *on input* U, W, aux *with coins* r*.*

We say that the $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-Knowledge of 2DH (K2DH) Assumption holds over group* G *and for auxiliary info* $\mathsf{AuxPrep}$*, if for every such* $\mathcal{A}$*, there exists a probabilistic companion algorithm* $\mathcal{E}^{DH}_{\mathcal{A}}$ *(called the extractor for* $\mathcal{A}$*) such that:* $\mathcal{E}^{DH}_{\mathcal{A}}$ *runs on input* U, W, aux, r *in time* $t_{\mathcal{E}}$ *and outputs* $\hat{Z}_1, \hat{Z}_2 \in G$ *or* $\perp$ *such that*

- *If* $\mathcal{E}_{\mathcal{A}}^{DH}(U, W, \mathsf{aux}, r) \neq \perp$ *then* $\hat{Z}_1 = DH(U, Z)$ *and* $\hat{Z}_2 = DH(W, Z)$
- *For every algorithm* C *running in time* $t_{\mathcal{D}}$*, we have that* $\Pr[C(U, W, Z, r, \mathsf{aux}) \in \{DH(U, Z), DH(W, Z)\} \mid \mathcal{E}_{\mathcal{A}}^{DH}(U, W, \mathsf{aux}, r) = \perp] \leq \epsilon_{\mathcal{D}}$*, where* $Z = \mathcal{A}(U, W, \mathsf{aux}, r)$ *and the probability is taken over the coins of* C *and uniform distribution on* (U, W, r)*.*

We express whether a party can compute functions of its own DH secret key with the following assumption: Consider an algorithm $\mathcal{A}$ that outputs a DH public key. Then an extractor $\mathcal{E}_{\mathcal{A}}^{DH}$ with runtime $t_{\mathcal{E}}$ given the same inputs as $\mathcal{A}$ and an extra DH public key can extract the DH shared secret of both public keys from $\mathcal{A}$. If $\mathcal{E}_{\mathcal{A}}^{DH}$ fails to extract the shared DH secret, then no other algorithm with runtime $t_{\mathcal{D}}$ succeeds with probability better than $\epsilon_{\mathcal{D}}$. To put it in a nutshell, extraction succeeds within $t_{\mathcal{E}}$, or not even within $t_{\mathcal{D}}$. We denote the DH shared secret between two DH public keys U, Z with respect to some generator g as $DH(U, Z)$.

Observe that in [VGIK20], Vatandas et al. show that the knowledge of exponent assumption and their novel knowledge of discrete logarithm assumption imply the knowledge of DH (KDH) assumption. Compared to the EKDH assumption, the algorithm $\mathcal{A}$ of the KDH assumption receives the public key U as input.

Definition 2.5 (EKDH Assumption [VGIK20])**.** *Let* G *be a cyclic group with generator* g *and* $\mathsf{AuxPrep}$ *a sampler for auxiliary inputs. Let* $\mathcal{A}$ *be any algorithm running in time* $t_{\mathcal{A}}$ *which runs on input* aux *where* $\mathsf{aux} \leftarrow_\$ \mathsf{AuxPrep}$*, and outputs* $Z \in G$*; we denote with* $Z = \mathcal{A}(\mathsf{aux}; r)$ *the output of running* $\mathcal{A}$ *on input* aux *with coins* r*.*

We say that the $(t_{\mathcal{A}}, t_{\mathcal{E}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-Extended Knowledge of DH (EKDH) Assumption holds over group* G *and for auxiliary info* $\mathsf{AuxPrep}$*, if for every such* $\mathcal{A}$*, there exists a probabilistic companion algorithm* $\mathcal{E}_{\mathcal{A}}^{DH}$ *(called the extractor for* $\mathcal{A}$*) such that:* $\mathcal{E}_{\mathcal{A}}^{DH}$ *runs on input* aux, r *and an additional input* $U \in G$ *in time* $t_{\mathcal{E}}$ *and outputs* $\hat{Z} \in G$ *or* $\perp$ *such that*

- *If* $\mathcal{E}_{\mathcal{A}}^{DH}(U, \mathsf{aux}, r) \neq \perp$ *then* $\hat{Z} = DH(U, Z)$
- *For every algorithm* C *running in time* $t_{\mathcal{D}}$*, we have that* $\Pr[C(U, r, \mathsf{aux}) = DH(U, Z) \mid \mathcal{E}_{\mathcal{A}}^{DH}(U, \mathsf{aux}, r) = \perp] \leq \epsilon_{\mathcal{D}}$*, where* $Z = \mathcal{A}(\mathsf{aux}, r)$ *and the probability is taken over the coins of* C *and uniform distribution on* r*.*

Looking ahead, in our analysis the group G implicitly refers to the group over which the protocol computes DH operations, which is curve448 or curve25519. [MP16b, KS23]

2.4 Key Encapsulation Mechanisms (KEMs)

We briefly review the syntax for KEMs.

Definition 2.6 (Key Encapsulation Mechanisms)**.** *A* key encapsulation mechanism (KEM) *is a triple of algorithms* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *with associated ciphertext space* $\mathcal{C}$ *and key space* $\mathcal{K}$*. In more detail:*

- $\mathsf{KGen}() \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$*: A probabilistic algorithm that outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Enc}(\mathsf{pk}) \xrightarrow{\$} (ct, ss)$*: A probabilistic algorithm taking as input a public key* pk *and outputs a ciphertext* $ct \in \mathcal{C}$ *and the therein encapsulated key* $ss \in \mathcal{K}$*.*
- $\mathsf{Dec}(\mathsf{sk}, ct) \rightarrow ss'$*: A deterministic algorithm taking as input a ciphertext* $ct \in \mathcal{C}$ *and secret key* sk *and outputs* $ss \in \mathcal{K} \cup \{\perp\}$*, where* $\perp$ *indicates an error.*

```
$\underline{\mathcal{G}^{dec}_{\mathsf{KEM}}(\mathcal{C},\mathcal{D})\text{:}}$
for $i \in [n_k]$
    $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$ \mathsf{KEM.KGen}()$
$\vec{\mathsf{pk}} \leftarrow \{\mathsf{pk}_i\}_{i\in[n_k]}$
$r \leftarrow\$ \mathcal{R}_\mathcal{C}$
$v \leftarrow \mathcal{C}^{\textsc{Decaps}}(\vec{\mathsf{pk}}; r)$
return $\mathcal{D}(v)$

$\underline{\textsc{Decaps}(i, ct)\text{:}}$
return $\mathsf{KEM.Dec}(\mathsf{sk}_i, ct)$
```

```
$\underline{\mathcal{G}^{ext}_{\mathsf{KEM}}(\mathcal{C},\mathcal{E}^{PA}_{\mathcal{C}},\mathcal{D})\text{:}}$
for $i \in [n_k]$
    $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$ \mathsf{KEM.KGen}()$
$\vec{\mathsf{pk}} \leftarrow \{\mathsf{pk}_i\}_{i\in[n_k]}$
$r \leftarrow\$ \mathcal{R}_\mathcal{C}$
$Q \leftarrow \emptyset$
$v \leftarrow \mathcal{C}^{\textsc{Decaps}}(\vec{\mathsf{pk}}; r)$
return $\mathcal{D}(v)$

$\underline{\textsc{Decaps}(i, ct)\text{:}}$
$ss \leftarrow\$ \mathcal{E}^{PA}_{\mathcal{C};Q}(i, ct, r)$
$Q \leftarrow Q \cup \{(i, ct, ss)\}$
return $ss$
```

Figure 1: Games for plaintext awareness, see Definition 2.7.

We define the key collision probability γ_{coll} for a KEM KEM *as the probability of the public keys of two independently sampled key pairs to coincide.*

We say that $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is *δ-correct* if, for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}()$ and every encapsulation $(ct, ss) \leftarrow\$ \mathsf{Enc}(\mathsf{pk})$, it holds that $\Pr[ss' \neq ss \mid ss' \leftarrow \mathsf{Dec}(\mathsf{sk}, ct)] \leq \delta$.

Plaintext awareness for KEMs denotes whether one can create a valid KEM encapsulation without knowing the corresponding shared secret. Based on plaintext awareness for PKE [BR95, BP04], and for KEMs [Den06, JW10], Hashimoto et al. [HKKP22] have adapted the definition to plaintext awareness for KEMs in the multi-key setting. We adapt their definition from asymptotic security to concrete security.

Definition 2.7 (Plaintext Awareness for KEMs [HKKP22])**.** *A KEM scheme* KEM *is* $(n_k, t_\mathcal{C}, t_\mathcal{E}, t_\mathcal{D}, \epsilon_\mathcal{D})$*-plaintext aware (PA1-secure) if for all (non-uniform) ciphertext creators $\mathcal{C}$ running in time $t_\mathcal{C}$, there exists an efficient extractor $\mathcal{E}^{PA}_\mathcal{C}$ running in time $t_\mathcal{E}$ such that for any distinguisher $\mathcal{D}$ running in time $t_\mathcal{D}$, the two experiments $\mathcal{G}^{dec}_{\mathsf{KEM}}(\mathcal{C},\mathcal{D})$ and $\mathcal{G}^{ext}_{\mathsf{KEM}}(\mathcal{C},\mathcal{E}^{PA}_\mathcal{C},\mathcal{D})$ in Figure 1 are indistinguishable except with a probability of $\epsilon_\mathcal{D}$.*

Intuitively, a KEM is plaintext aware if creating a KEM ciphertext implies knowledge of the corresponding shared secret. This is formally modeled with a ciphertext creator who has access to a decryption oracle. This decryption oracle is implemented either with the actual decryption or with an extractor that depends on the ciphertext creator, its randomness, and the oracle queries and responses so far. Finally, the ciphertext creator outputs a string, which is fed into a distinguisher. The distinguisher needs to distinguish how the decryption oracle is implemented.

Note that Kyber [SAB$^+$22] using the FO transform with explicit rejection instead of implicit rejection is plaintext aware.

2.5 Authenticated Encryption with Associated Data (AEAD)

We follow the exposition of [Rog02].

Definition 2.8 (AEAD)**.** *An Authenticated Encryption scheme with Associated Data is a pair of algorithms* $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$ *with associated key space $\mathcal{K}$, nonce space $\mathcal{N}$, associated data (header) space $\mathcal{H}$, message space $\mathcal{M}_{\mathsf{AE}}$, and ciphertext space $\mathcal{C}$, defined as follows:*

- $\mathsf{Enc}(k, n, AD, \mu) \rightarrow ct$*: On input a key k, a nonce n, associated data AD, and a message μ, this deterministic algorithm returns a ciphertext ct.*
- $\mathsf{Dec}(k, n, AD, ct) \rightarrow \mu$*: On input a key k, a nonce n, associated data AD, and a ciphertext ct, this deterministic algorithm returns a message μ or a distinguished error symbol $\perp \notin \mathcal{M}_{\mathsf{AE}}$.*

We say that an AEAD scheme is correct *if, for every $k \in \mathcal{K}, n \in \mathcal{N}, AD \in \mathcal{H}, \mu \in \mathcal{M}_{\mathsf{AE}}$ it holds that* $\Pr[\mathsf{Dec}(k, n, AD, \mathsf{Enc}(k, n, AD, \mu)) = \mu] = 1$.

Note that throughout the paper we omit the nonce since Signal's implementations of $\mathsf{X3DH}$ and PQXDH derive the nonce deterministically from the key.

2.6 Key Exchange Protocols

We review the syntax for key exchange protocols, essentially following [BFG+22, BFG+21]. We differentiate between protocol messages m and user (plaintext) messages μ (such as "Hi Bob, how are you doing?"). A key exchange protocol may allow a user message to be sent along with a protocol message during the execution, e.g. $\mathsf{X3DH}$ and PQXDH.

Definition 2.9 (Key Exchange Protocol)**.** *A 2-party key exchange protocol is a triple of algorithms* $\mathsf{KE} = (\mathsf{KGenLT}, \mathsf{KGenSS}, \mathsf{Run})$*, defined as follows:*

- $\mathsf{KGenLT}() \overset{\$}{\rightarrow} (ltpk_U, ltsk_U)$*: A probabilistic long-term key generation algorithm that outputs a public-key/secret-key pair attributed to user U.*
- $\mathsf{KGenSS}() \overset{\$}{\rightarrow} (sspk_U^{\mathsf{ssid}}, sssk_U^{\mathsf{ssid}})$*: A probabilistic semi-static key generation algorithm that outputs a public-key/secret-key pair attributed to user U and semi-static identifier* ssid*.*
- $\mathsf{Run}(ltsk_U, \vec{sssk}_U, \vec{ltpk}, \vec{sspk}, \pi, m, \mu) \overset{\$}{\rightarrow} (\pi', m', \mu')$*: A probabilistic session execution algorithm that takes as input a party's long-term secret key $ltsk_U$, a list of that party's semi-static secret keys $\vec{sssk}_U$, lists of long-term and semi-static public keys for all honest parties $\vec{ltpk}$ and $\vec{sspk}$, a session state π, an incoming message m, and a (potentially empty) user message μ to be sent, and outputs an updated session state π', a (possibly empty) outgoing message m', and a (possibly empty) incoming user message μ'.*

We assume a distinguished message $m = (\mathsf{create}, \mathsf{info}_{\mathsf{create}})$ upon which Run sets up the session and produces the first message. Note that an ephemeral key generation algorithm is not mandatory and can happen inside of Run.

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 in this session.
- $\mathsf{K} \in \mathcal{K}_{\mathsf{KE}} \cup \{\perp\}$: The session key established in this session, initialized to $\perp$.
- $\mathsf{info}_{\mathsf{create}}$ is a tuple of elements that indicates the components of the first message. We use the following elements:
  - $\mathsf{ssid} \in [n_{ss}]$ denotes the identifier of the initiator's semi-static key in this session. If $\pi.\mathsf{role} = \mathsf{initiator}$ this refers to $sspk_{\pi.\mathsf{oid}}^{\mathsf{ssid}}$, if $\pi.\mathsf{role} = \mathsf{responder}$ this refers to $sspk_{\pi.\mathsf{pid}}^{\mathsf{ssid}}$.
  - $e_{\mathsf{DH}} \in \{\mathsf{true}, \mathsf{false}\}$ denotes whether an ephemeral DH key is included in the first message.
  - $e_{\mathsf{KEM}} \in \{\mathsf{true}, \mathsf{false}\}$ denotes whether an ephemeral KEM key is included in the first message.

Asynchronous key exchange. In principle, a key exchange protocol can have an arbitrary number of message flows n_m, 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 initiator Bob with $m = (\mathsf{create}, \mathsf{info}_{\mathsf{create}})$, which sets up the initiator session and outputs the pre-key bundle, possibly including ephemeral public keys; 2) a call to Run at the responder Alice with the initiator's pre-key bundle, which generates a session key and outputs a key exchange message; and 3) a call to Run at the initiator with the responder's long-term public key and key exchange message, which generates a session key and has no output message. If a party outputs an empty message before the protocol execution is finished, it aborts the protocol.

3 Our deniability model

We introduce our deniability model that facilitates combining ideas of deniability definitions from prior literature, hence gaining expressiveness. We show how our model can match definitions from prior literature in Appendix A.

We want to capture that artifacts of a key exchange between Alice and Bob do not convince a third party that the key exchange actually took place. Consider Alice, who tries to prove Bob's involvement. To defend against Alice's claim, Bob must be able to argue that the artifacts were produced without him, e.g. by Alice or by just anybody. Hence, we require the existence of a Fake algorithm that produces these artifacts, where the artifacts are indistinguishable from artifacts of an actual execution of the key exchange protocol.

Our model captures this with the following game: First, the challenger prepares the game by initializing variables, sampling keys for all users, preparing auxiliary info aux according to AuxPrep, and sampling a secret bit b. Second, the adversary interacts with some oracles. One of the oracles depends on the secret bit b and embeds the challenge: To distinguish whether (a part of) a protocol transcript was generated honestly or with a dedicated Fake algorithm, which needs to be given in the proof. Third, the distinguisher guesses the challenge bit, based on the previous actions of the adversary. In particular, the distinguisher gets the transcript of the interactions between the adversary and the challenge oracle and the (honestly computed) session keys for all sessions. Note that $\mathcal{A}$ and Fake get access to the auxiliary info aux.

The game is parameterized by the adversary's capabilities O_A, i.e. the oracles it can query, the capabilities of the Fake algorithm O_F, the capabilities of the distinguisher O_D, and the sampling method for auxiliary information AuxPrep. Extending O_A or O_D strengthens the deniability guarantee, while extending O_F or AuxPrep weakens the guarantee. In Sections 3.1 to 3.4 we explain these parameters in detail. In Section 3.5 we discuss how to compare definitions with different parameters. The flexibility and expressiveness of these parameters is the novelty of our model.

Definition 3.1 (Deniability for Key Exchange)**.** *We say that a key exchange protocol* $\mathsf{KE} = (\mathsf{KGenLT}, \mathsf{KGenSS}, \mathsf{Run})$ *is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable wrt.* (O_A, O_F, O_D)*-oracles, where* $O_A \subseteq \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}, \textsc{ChallResp}, \textsc{RegHon}, \textsc{ChallHonInit}, \textsc{ChallHonResp}\}$*,* $O_F \subseteq \{SK, \textsc{User}_{x,y}\}$*,* $O_D \subseteq \{SK\textsc{s}, \textsc{aux}\}$*, and auxiliary inputs sampled with* AuxPrep*, if for any adversary* $\mathcal{A}$ *running in time* $t_{\mathcal{A}}$*, making queries to* O_A *limited by* q_{O_A}*, there exists an algorithm* Fake *running in time* t_{F}*, making queries to* O_F *limited by* q_{O_F} *such that for any distinguisher* $\mathcal{D}$ *making queries to* O_D *limited by* q_{O_D} *and running in time* $t_{\mathcal{D}}$*, it holds that*

$$\Pr[\mathcal{G}^{O_A,O_F,O_D\text{-}den}_{\mathsf{KE},\mathsf{AuxPrep},n_p,n_{ss}}(\mathcal{A},\mathcal{D}) = 1] \leq \frac{1}{2} + \epsilon_{\mathcal{D}},$$

where $\mathcal{G}^{O_A,O_F,O_D\text{-}den}_{\mathsf{KE},\mathsf{AuxPrep},n_p,n_{ss}}(\mathcal{A},\mathcal{D})$ *is defined in Figure 2.*

```
$\mathcal{G}^{\mathrm{O}_A,\mathrm{O}_F,\mathrm{O}_D\text{-}den}_{\mathsf{KE},\mathsf{AuxPrep},n_p,n_{ss}}(\mathcal{A},\mathcal{D})$:
1  $Q[\cdot] \leftarrow [];\quad K[\cdot] \leftarrow \perp;\quad C \leftarrow \emptyset$
2  $(\vec{\mathsf{pk}}, \vec{\mathsf{sk}}) \leftarrow\!\$\ \mathsf{KeyPrep}(n_p, n_{ss})$
3  $\mathsf{aux} \leftarrow\!\$\ \mathsf{AuxPrep}(\vec{\mathsf{pk}}, \vec{\mathsf{sk}})$
4  $b \leftarrow\!\$\ \{0,1\}$
5  $(\vec{r_R}, \vec{r_C}) \leftarrow\!\$\ \mathcal{R}_{RH}^{q_{RH}} \times \mathcal{R}_{CH}^{q_{CH}}$ //randomness for ChallHon, RegHon oracles
6  $r_{\mathcal{A}} \leftarrow\!\$\ \mathcal{R}_{\mathcal{A}};\quad r \leftarrow (r_{\mathcal{A}}, \vec{r_C}, \vec{r_R})$ //randomness for the adversary
7  $\mathcal{A}^{\mathrm{O}_A}(\vec{\mathsf{pk}}, \mathsf{aux}, (\vec{r_C}, \vec{r_R}); r_{\mathcal{A}})$
8  $b' \leftarrow\!\$\ \mathcal{D}^{\mathrm{O}_D}(\vec{\mathsf{pk}}, r, Q, K)$
9  return $[\![b' = b]\!]$

$\mathsf{KeyPrep}(n_p, n_{ss})$:
10 for $U \in [n_p]$ //number of keys to prepare
11   $(ltpk_U, ltsk_U) \leftarrow\!\$\ \mathsf{KGenLT}()$
12   for $\mathsf{ssid} \in [n_{ss}]$
13     $(sspk_U^{\mathsf{ssid}}, sssk_U^{\mathsf{ssid}}) \leftarrow\!\$\ \mathsf{KGenSS}()$
14 $\vec{ltpk} \leftarrow \{ltpk_U\}_{U\in[n_p]};\quad \vec{sspk} \leftarrow \{sspk_U^{\mathsf{ssid}}\}_{U\in[n_p]}^{\mathsf{ssid}\in[n_{ss}]}$
15 $\vec{ltsk} \leftarrow \{ltsk_U\}_{U\in[n_p]};\quad \vec{sssk} \leftarrow \{sssk_U^{\mathsf{ssid}}\}_{U\in[n_p]}^{\mathsf{ssid}\in[n_{ss}]}$
16 return $((\vec{ltpk}, \vec{sspk}), (\vec{ltsk}, \vec{sssk}))$

$\textsc{Reg}(U, \vec{\mathsf{pk}}_U)$:
17 if $U \in [n_p] \cup C$ //abort if another key is already known for $U$
18   return $\perp$
19 $C \leftarrow C \cup \{U\}$
20 $\vec{\mathsf{pk}} \xleftarrow{+} \vec{\mathsf{pk}}_U$ //includes semi-static keys
21 return success

$\textsc{Init}(U, s, \mathsf{role}, V)$:
22 if $\pi_U^s = \perp \wedge U \notin C$
23   $\pi_U^s.\mathsf{role} \leftarrow \mathsf{role}$
24   $\pi_U^s.\mathsf{oid} \leftarrow U$ //set owner identity
25   $\pi_U^s.\mathsf{pid} \leftarrow V$ //set partner identity ($\star$ if post-specified peers)
26   return success
27 return $\perp$

$\textsc{RegHon}(U)$:
28 if $U \in [n_p] \cup C$ //abort if another key is already known for $U$
29   return $\perp$
30 $(\vec{\mathsf{pk}}_U, \vec{\mathsf{sk}}_U) \leftarrow \mathsf{KeyPrep}(1, n_{ss}; r_R)$ //next $r_R$ pre-sampled in line 5
31 $\vec{\mathsf{pk}} \xleftarrow{+} \vec{\mathsf{pk}}_U;\quad \vec{\mathsf{sk}} \xleftarrow{+} \vec{\mathsf{sk}}_U$
32 $C \leftarrow C \cup \{U\}$
33 return $(\vec{\mathsf{pk}}_U, \vec{\mathsf{sk}}_U)$ //simulates $\mathcal{A}$ honestly generating keys

$\textsc{ChallInit}(U, s, m, \mu)$ $\boxed{\textsc{ChallResp}(U, s, m, \mu)}$:
34 if $\pi_U^s = \perp \vee \pi_U^s.\mathsf{role} = \mathsf{responder}$ $\boxed{\mathsf{initiator}}$ //initialized sessions only
35   return $\perp$
36 if $b = 0$
37   $(\pi_U^s, m') \leftarrow\!\$\ \mathsf{Run}(\mathsf{sk}_U, \vec{\mathsf{pk}}, \pi_U^s, m, \mu)$
38 else
39   $(\pi_U^s, m') \leftarrow\!\$\ \mathsf{Fake}^{\mathrm{O}_F}(\vec{\mathsf{pk}}, \pi_U^s, m, \mu, \mathsf{aux}, r, Q)$ //only $r_R, r_C$ so far
40 $Q[\pi_U^s] \xleftarrow{+} (m, m')$
41 $K[\pi_U^s] \leftarrow \pi_U^s.\mathsf{K}$
42 return $m'$

$\textsc{ChallHonInit}(U, V, \mathsf{info}_{\mathsf{create}}, \vec{\mu})$:
43 if $U \in C$ //do not allow challenging a party under $\mathcal{A}$'s control
44   return $\perp$
45 $\pi_U.\mathsf{oid} \leftarrow U;\quad \pi_U.\mathsf{pid} \leftarrow V$
46 $\pi_V.\mathsf{oid} \leftarrow V;\quad \pi_V.\mathsf{pid} \leftarrow U$
47 $\pi_U.\mathsf{role} \leftarrow \mathsf{initiator};\quad \pi_V.\mathsf{role} \leftarrow \mathsf{responder}$
48 $m_1 \leftarrow (\mathsf{create}, \mathsf{info}_{\mathsf{create}})$
49 $(\mu_1, \ldots, \mu_{n_m}) \leftarrow \vec{\mu}$
50 for $i \in [1, 3, \ldots, n_m - 1]$ //until $n_m$ if $n_m$ is odd
51   if $b = 0$
52     $(\pi_U, m_{i+1}) \leftarrow\!\$\ \mathsf{Run}(\mathsf{sk}_U, \vec{\mathsf{pk}}, \pi_U, m_i, \mu_i)$
53   else
54     $(\pi_U, m_{i+1}) \leftarrow\!\$\ \mathsf{Fake}^{\mathrm{O}_F}(\vec{\mathsf{pk}}, \pi_U, m_i, \mu_i, \mathsf{aux}, r_C)$
55   $(\pi_V, m_{i+2}) \leftarrow \mathsf{Run}(\mathsf{sk}_V, \vec{\mathsf{pk}}, \pi_V, m_{i+1}, \mu_{i+1}; r_C)$
       //next $r_C$ pre-sampled in line 5
56 $Q[\pi_U] \leftarrow (m_i)_{i=1}^{n_m};\qquad K[\pi_U] \leftarrow \pi_U.K$
57 return $(Q[\pi_U], K[\pi_U])$

$\textsc{ChallHonResp}(U, V, \mathsf{info}_{\mathsf{create}}, \vec{\mu}, r_C)$:
58 if $V \in C$ //do not allow challenging a party under $\mathcal{A}$'s control
59   return $\perp$
60 $\pi_U.\mathsf{oid} \leftarrow U;\quad \pi_U.\mathsf{pid} \leftarrow V$
61 $\pi_V.\mathsf{oid} \leftarrow V;\quad \pi_V.\mathsf{pid} \leftarrow U$
62 $\pi_U.\mathsf{role} \leftarrow \mathsf{initiator};\quad \pi_V.\mathsf{role} \leftarrow \mathsf{responder}$
63 $m_1 \leftarrow (\mathsf{create}, \mathsf{info}_{\mathsf{create}})$
64 $(\mu_1, \ldots, \mu_{n_m}) \leftarrow \vec{\mu}$
65 for $i \in [1, 3, \ldots, n_m - 1]$ //until $n_m$ if $n_m$ is odd
66   $(\pi_U, m_{i+1}) \leftarrow \mathsf{Run}(\mathsf{sk}_U, \vec{\mathsf{pk}}, \pi_U, m_i, \mu_i; r_C)$
67   if $b = 0$
68     $(\pi_V, m_{i+2}) \leftarrow\!\$\ \mathsf{Run}(\mathsf{sk}_V, \vec{\mathsf{pk}}, \pi_V, m_{i+1}, \mu_{i+1})$
69   else
70     $(\pi_V, m_{i+2}) \leftarrow\!\$\ \mathsf{Fake}^{\mathrm{O}_F}(\vec{\mathsf{pk}}, \pi_V, m_{i+1}, \mu_{i+1}, \mathsf{aux}, r_C)$
71 $Q[\pi_V] \leftarrow (m_i)_{i=1}^{n_m};\qquad K[\pi_V] \leftarrow \pi_V.K$
72 return $(Q[\pi_V], K[\pi_V])$

$\textsc{User}_{x,y}(\pi, u, m, \mu)$:
73 $U \leftarrow \pi.\mathsf{oid}$
74 $H \leftarrow H_\pi^u$ //easier notation for $u^{\text{th}}$ honest dummy for session $\pi$
75 if $\pi_U^H = \perp$
76   $(\mathsf{pk}_H, \mathsf{sk}_H) \leftarrow\!\$\ \mathsf{KeyPrep}(1, 1)$
77   $\vec{\mathsf{pk}} \leftarrow \vec{\mathsf{pk}} \cup \{\mathsf{pk}_H\}$
78   if $\pi.\mathsf{role} = \mathsf{initiator}$
79     $\pi_U^H.\mathsf{role} \leftarrow x$
80   else
81     $\pi_U^H.\mathsf{role} \leftarrow y$
82   $\pi_U^H.\mathsf{oid} \leftarrow U$
83   $\pi_U^H.\mathsf{pid} \leftarrow H$
84 $(\pi_U^H, m', \mu') \leftarrow\!\$\ \mathsf{Run}(\mathsf{sk}_U, \vec{\mathsf{pk}}, \pi_U^H, m, \mu)$
85 return $m'$

$\textsc{SK}(\pi)$:
86 return $\mathsf{sk}_{\pi.\mathsf{pid}}$

$\textsc{SKs}()$:
87 return $\vec{\mathsf{sk}}$

$\textsc{aux}()$:
88 return $\mathsf{aux}$
```

Figure 2: Our deniability game parameterized in the adversary's capabilities $\mathrm{O}_A \subseteq \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}, \textsc{ChallResp}, \textsc{RegHon}, \textsc{ChallHonInit}, \textsc{ChallHonResp}\}$, the capabilities of the Fake algorithm $\mathrm{O}_F \subseteq \{\textsc{SK}, \textsc{User}_{x,y}\}$, the capabilities of the distinguisher $\mathcal{D}$ $\mathrm{O}_D \subseteq \{\textsc{SKs}, \textsc{aux}\}$, and the sampling algorithm for auxiliary info $\mathsf{AuxPrep}$.

Type of Deniability	O_A	O_F	O_D
against malicious adversaries	$\textsc{Reg}, \textsc{Init}$, $\textsc{Chall}$	$\Diamond$	$\Diamond$
against semi-honest adversaries	$\textsc{RegHon}$, $\textsc{ChallHon}$	$\Diamond$	$\Diamond$
against passive adversaries	$\textsc{ChallHon}$	$\Diamond$	$\Diamond$
partial deniability wrt. an x, y oracle	$\Diamond$	$\textsc{User}_{x,y}$	$\Diamond$
1-out-of-2 deniability	$\Diamond$	$\textsc{SK}$	$\Diamond$
in the big brother model	$\Diamond$	$\Diamond$	$\textsc{SKs}$
with aux known to the distinguisher	$\Diamond$	$\Diamond$	$\textsc{aux}$

Table 1: Translating oracles into names for adversarial capabilities and deniability guarantees. We refer to an arbitrary set with $\Diamond$. For the challenge oracles we omit the suffix indicating which role they act as.

Intuitively, the Fake algorithm models that a protocol message was not necessarily produced by the alleged sender. Consider an adversary $\mathcal{A}$ acting as Alice (i.e. the adversary generates keys for Alice), who produces Alice's message(s) according to Run and Bob's message(s) and the session key according to the Fake algorithm. Note that the Fake algorithm also needs to produce the session key to ensure deniability of the subsequent protocol. Since the Fake algorithm is public, Bob can argue that the transcript and session key were produced by Alice using the Fake algorithm, *without* his involvement. On a technical note, the Fake algorithm knows the code and randomness of the adversary $\mathcal{A}$[5], modeling that the adversary $\mathcal{A}$ and the Fake algorithm are in cahoots. This allows the Fake algorithm to extract knowledge from the adversary under some knowledge assumptions.

In the following we describe the choices for using our model and how our model allows easy comparisons of deniability notions.

3.1 AuxPrep: Sampling auxiliary information

The auxiliary info aux models that an adversary may obtain relevant information prior to the experiment. To the best of our knowledge, this idea was introduced by [DGK06]: They use aux to model that the adversary may have obtained transcripts of previous protocol executions by eavesdropping. Following [DGK06], the auxiliary info is also available to the Fake algorithm, i.e. anybody who knows the auxiliary info can simulate a transcript and session key. Optionally, the distinguisher gets access to aux as well (cp. Section 3.4.2).

3.2 O_A: capabilities of $\mathcal{A}$

The oracles in O_A model the behavior of the adversary, i.e. whether it is malicious or semi-honest,[6] and define the challenge. In the following, we remark on how we handle the roles of parties and detail the usage of the respective subsets of oracles.

3.2.1 Initiator and receiver deniability

We obtain *initiator deniability* by having $\textsc{ChallInit} \in \mathrm{O}_A$ or $\textsc{ChallHonInit} \in \mathrm{O}_A$, and *responder deniability* by having $\textsc{ChallResp} \in \mathrm{O}_A$ or $\textsc{ChallHonResp} \in \mathrm{O}_A$; i.e. we give separate challenge oracles per role. This allows separate analyses of deniability per role. (Section 4 shows that the deniability

[5]For malicious adversaries (see Section 3.2), the Fake algorithm additionally gets the responses of the challenge oracle so far since the adversary's messages may depend on those responses.

[6]In Appendix B we also treat passive adversaries for the sake of completeness. We do not consider passive adversaries for our analysis and therefore defer it to the appendix.

guarantees for both roles differ for X3DH and PQXDH.) Also, this enables more precise specifications, e.g. deniability for the responder may have stricter requirements than for the initiator.

3.2.2 Malicious adversaries

We obtain *deniability against malicious adversaries* with $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}, \textsc{ChallResp}\}$, allowing at most q_R queries to the Reg oracle, q_I queries to the Init oracle, q_{CI} queries to the ChallInit oracle, and q_{CR} queries to the ChallResp oracle, for $q_R, q_I, q_{CI}, q_{CR} \in q_{\mathrm{O}_A}$. The Reg oracle allows the adversary to register maliciously generated keys with the challenger. The Init oracle allows the adversary to initialize sessions for honest users and to partner these sessions arbitrarily. The ChallInit and ChallResp oracles allow the adversary to send arbitrary messages to sessions (including, e.g. messages replayed from other sessions) and to interleave sessions at will. The ChallInit and ChallResp oracles respond either according to the protocol specification via Run or "simulate" via Fake, depending on the challenge bit b. For the case of Signal's initial handshake, note that this also captures an adversarially controlled server, i.e. a server who forwards messages wrongly. Only sessions of honestly generated users can be challenged since the Init oracle does not create sessions for users with maliciously generated keys.

3.2.3 Semi-honest Adversaries

Next, we obtain the notion of *deniability against semi-honest adversaries* by setting the oracle $\mathrm{O}_A = \{\textsc{RegHon}, \textsc{ChallHonInit}, \textsc{ChallHonResp}\}$, allowing at most q_{RH} queries to the RegHon oracle, q_{CI} queries to the ChallHonInit oracle, and q_{CR} queries to the ChallHonResp oracle, for q_{RH}, $q_{CI}, q_{CR} \in q_{\mathrm{O}_A}$. To register a key pair for a new user, the adversary can query the RegHon oracle, which returns the user's private key(s) to the adversary. A semi-honest adversary partners sessions correctly and obeys the protocol flow. Hence, we merge the Init oracle and the challenge oracles. These ChallHonInit, ChallHonResp oracles partner the sessions correctly and directly produce the complete transcript, thereby enforcing correct message flow.

Only honest users can be challenged (cp. lines 43 and 58). The randomness used by these three oracles (RegHon, ChallHonInit, ChallHonResp) is sampled at the beginning of the game (in line 5) and passed to the adversary[7]. This models that the adversary knows the randomness for honest key generation and honest protocol participation. Furthermore, The adversary provides the protocol-specific $\mathsf{info}_{\mathsf{create}}$ to indicate the components of the first message.

3.2.4 Combining different adversarial capabilities

It is possible to combine the aforementioned oracles at will, e.g. achieving responder deniability against malicious adversaries and initiator deniability against semi-honest adversaries with malicious keys via $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{ChallResp}, \textsc{ChallHonInit}\}$. Note that O_A should always contain at least one challenge oracle; and for the ChallInit, ChallResp oracles the Init oracle is necessary. Oracles for both malicious and semi-honest adversaries seem redundant but may have some use cases.

3.2.5 Adaptive Corruptions

Several works [CF11, JS08, Jia14, JCL+22] allow the adversary to adaptively corrupt parties. The attacker, the simulator, and possibly the distinguisher learn the corrupted parties' secrets. Fundamentally, deniability is a statement that a Fake algorithm can simulate a transcript (and session key). The Fake algorithm needs to work even if the adversary does not corrupt any party. Hence, the strategy of the Fake algorithm should not rely on secret keys of corrupted parties and that is why our model does not consider

[7] In consequence, the distinguisher learns this randomness as well.

adaptive corruptions. Appendices A.2 and A.8 describe how to represent non-adaptive corruptions in our model.

3.3 O_F: capabilities of Fake

The oracles in O_F are accessed by the Fake algorithm. They offer help in simulating a transcript and session key and should reflect the capabilities of an adversary who wants to frame a user for a particular transcript. If fewer oracles are required, then more people can simulate, i.e. the deniability guarantee gets stronger.

3.3.1 Partial deniability (User$_{x,y}$ oracle (parameterized by x, y))

We obtain *partial deniability wrt.* x, y *oracles* by having $\textsc{User}_{x,y} \in \mathrm{O}_F$, allowing at most q_U sessions with the $\textsc{User}_{x,y}$ oracle per session π, where $x, y \in \{\mathsf{initiator}, \mathsf{responder}, \perp\}$. Each invocation of the Fake algorithm (lines 54, 70, and 39) has access to exactly one session π. The Fake algorithm can query the $\textsc{User}_{x,y}$ oracle with this session π to get access to another session with $\pi.\mathsf{oid}$ (the owner of π) with the role x (if $\pi.\mathsf{oid}$ is the initiator in π) or y (if $\pi.\mathsf{oid}$ is the responder in π), where $\perp$ denotes no access. The Fake algorithm can have up to $q_U \in q_{\mathrm{O}_F}$ sessions with the $\textsc{User}_{x,y}$ oracle per session π. For $u \in [q_U]$, the user $\pi.\mathsf{oid}$ is partnered with a (freshly generated) honest user H_π^u (and in particular not with $\pi.\mathsf{pid}$), where $H_\pi^u \in \mathcal{H}$ and $\mathcal{H} \cap [n_p] = \emptyset$.

This idea is sourced from *partial deniability* [DGK06] (informally known as *peer independence* [DGK06] or *receiver obliviousness* [HKKP22], following the idea of *post-specified peers* [CK02]).

This oracle models that an adversary can start a separate session with the victim $\pi.\mathsf{oid}$ to produce the transcript. This oracle can be helpful if messages do not contain session-specific data.

3.3.2 1-out-of-2 or 1-out-of-∞ deniability (SK oracle)

We obtain 1-*out-of*-2 *deniability* by having $\mathrm{SK} \in \mathrm{O}_F$. If $\mathrm{SK} \notin \mathrm{O}_F$, then we obtain 1-*out-of*-∞ *deniability*. The SK oracle grants the Fake algorithm access to the peer's secret keys, thereby modeling that the peer (and only the peer) can execute the Fake algorithm. Hence, 1-out-of-∞ deniability is more desirable. Note that this refers to long-term and semi-static keys, i.e. keys that are registered with the challenger. In particular, the SK oracle cannot yield secret keys of users that the adversary registered via the $\textsc{Reg}$ oracle.

If a transcript (and session key) can stem from either an honest execution or the peer creating the transcript on his or her own (with the Fake algorithm) then there are two options for this; hence, the name 1-out-of-2 deniability. Absence of this oracle indicates that anybody could have produced a transcript, hence, we dub it 1-out-of-∞ deniability. These terms were coined by [HLLC11] for authentication protocols[8] and adopted for key exchange by [BFG+22].

The idea of using one party's secret key for faking a transcript was also used by [VGIK20], though coming from a different angle: They allow the simulator (roughly corresponding to our Fake algorithm) to use the adversary's long-term secret key to fake a transcript. They justify this with the adversary having registered its long-term key to some form of PKI, possibly with an extractable proof of knowledge, from which the simulator can learn the secret key.

[8]They also call a ring signature with a ring of n members 1-out-of-n deniable. This allows for a philosophical debate whether this applies to messaging services with n registered users. We argue that anybody can sign up for the service and, hence, there is no practical difference to 1-out-of-∞ deniability.

3.4 O_D: capabilities of $\mathcal{D}$

The distinguisher always gets all users' public keys, the transcripts and session keys from the interaction with the Chall oracle, and the randomness of the adversary. The oracles in O_D define the extra power that the distinguisher may have in the form of access to all (honestly generated) secret keys (i.e. the big brother model) or the auxiliary info. The more oracles the distinguisher has access to, the stronger the deniability guarantee we obtain.

3.4.1 Big brother model

We obtain *deniability in the big brother model* by having $\textsc{SKs} \in \mathrm{O}_D$. By querying the SKs oracle, the distinguisher gets access to all secret keys that the challenger prepared for all users as well as secret keys registered by the adversary via the RegHon oracle. This models that the distinguisher can subpoena all parties into giving up their private keys or can exert social pressure in a different manner. Note that the SKs oracle does not yield secrets of keys registered via the Reg oracle—the challenger does not know these secret keys and possibly the adversary does not either. Similarly, the SKs oracle does not yield ephemeral secret keys, as they are not registered with the challenger (and, for X3DH and PQXDH, honest parties erase ephemeral secret keys directly after use).

The term *big brother* was introduced by [Nao02] in the context of deniable ring authentication. The adversary divulging its secret key to the distinguisher was coined complete deniability by [MP02]. Informal descriptions [UDB$^+$15, UG15] and the UC-based definitions of [Ung21] are set in this model without using this name. The game-based definition of [BFG$^+$22] first uses the term big brother model for deniability of key exchange protocols.

3.4.2 Auxiliary info known to the distinguisher

We obtain *deniability with* aux *known to the distinguisher* by having $\textsc{aux} \in \mathrm{O}_D$. By querying the aux oracle, the distinguisher gets access to the auxiliary information (cp. Section 3.1). This models that the distinguisher learns whatever the adversary has eavesdropped on before the beginning of the experiment. If $\textsc{aux} \notin \mathrm{O}_D$ then the deniability guarantee *relies on* the distinguisher never learning aux. This is important for analyzing PQXDH, where the Fake algorithm reuses signatures from aux, which are not reused in honest executions, and thereby allows the distinguisher to tell Run and Fake apart.

3.5 Comparability of deniability notions

Our model allows fairly easy comparisons between any two deniability notions by weighing $\mathrm{O}_A, \mathrm{O}_F, \mathrm{O}_D, \mathsf{AuxPrep}$ between the two notions. If the adversary $\mathcal{A}$ or the distinguisher $\mathcal{D}$ gain more capabilities, i.e. if $\mathrm{O}_A, \mathrm{O}_D$ contain more or stronger oracles, the deniability guarantee becomes stronger (since it holds in more cases). Though, the aux oracle does not make a difference in case $\mathsf{aux} = \perp$. However, if Fake gains more capabilities, i.e. if O_F contains more oracles, the deniability guarantee becomes weaker: Simulating a transcript requires access to the oracles in O_F. Similarly, the more information is included in the auxiliary information aux, the harder it is to match these circumstances and the weaker the guarantee[9].

It is easy to see that an algorithm has more capabilities if it gains an extra oracle. The adversary $\mathcal{A}$ can have access to oracles for malicious adversaries ($\textsc{Reg}, \textsc{ChallInit}, \textsc{ChallResp}$) and for semi-honest adversaries ($\textsc{RegHon}, \textsc{ChallHonInit}, \textsc{ChallHonResp}$). It is clear that the former oracles are stronger than the latter, yielding a stronger deniability guarantee.[10]

[9]Subsequent work by Fiedler and Langrehr [FL25, Appendix B] points out that different types of auxiliary info can be helpful to the adversary or to the Fake algorithm. This paper considers only auxiliary info helpful to the Fake algorithm.

[10]Assuming that the $\textsc{ChallInit}, \textsc{ChallResp}$ oracles come with an Init oracle, since they are otherwise useless.

This leaves some relations open, e.g. how does deniability against semi-honest adversaries in the big brother model compare to deniability against malicious adversaries? While both notions remain incomparable, our model makes the differences explicit: We can argue about the "upper bound" (malicious adversaries in the big brother model) and the "lower bound" (semi-honest adversaries). Note that a deniability guarantee also holds for smaller numbers of users and semi-static keys: If it holds for n_p, n_{ss}, it also holds for $n'_p \leq n_p, n'_{ss} \leq n_{ss}$.[11]

4 Deniability of Signal's initial handshake

We look at Signal's initial handshake protocols—X3DH and its recent replacement PQXDH adding post-quantum confidentiality—and which deniability notions they fulfill. Note that both protocols aim for classical authentication and deniability, and not post-quantum deniability. Figure 3 gives an algorithmic overview of both protocols and the respective subsections give a textual description. The protocol specification [MP16b, KS23] uses only one long-term key per user, which is used for both the DH scheme and the signature scheme XEdDSA [Per16]. We follow [BJKS23] in treating them as two separate keys. We follow the protocol specification in using the session key directly as key for the AEAD scheme. Signal's implementation derives several values from the session key, among them the key and nonce used for the AEAD scheme. Beware that we consider Bob, who creates the pre-key bundle, as initiator, following e.g. [UG15, CCD+17, UG18] and contrasting [BFG+22].

We summarize our findings for both X3DH and PQXDH in Table 2, stated separately per role. For initiator (Bob) deniability, we have two results per adversary model to differentiate whether the Fake algorithm has access to a signature that the distinguisher does not know, i.e. a *private signature*, or if Fake and the distinguisher know the same signatures, i.e. *public signatures*. For X3DH either case works (see Remark 4.1), while for PQXDH we *must have* a private signature for the ephemeral KEM key (see Theorem 4.4 and Remark 4.4). Without further restrictions, we achieve 1-out-of-2 deniability against semi-honest adversaries. We use a knowledge of DH assumption (see Definitions 2.4 and 2.5) against malicious adversaries and additionally plaintext awareness of the KEM (see Definition 2.7) for PQXDH to simulate the session key. Concerning responder (Alice) deniability, PQXDH matches the guarantees of X3DH without limitations. For malicious adversaries in the big brother model, i.e. the strongest possible adversary model, we give our thoughts in Remarks 4.2 and 4.5. Note that Table 2 implies all other notions: Each theorem still holds if you weaken the adversary, weaken the distinguisher, add an oracle to O_F, or add auxiliary info (without the Fake algorithm using it).

While our analysis relies on the random oracle model, we do not program the RO, following [DGK06] and in line with the results of [Pas03]. All Fake algorithms abort on receiving malformed inputs. We state the expected message format explicitly in the pseudocode.

4.1 X3DH

We recall how Signal's classical initial handshake protocol X3DH [MP16b] works, see Figure 3. First, Bob signs his semi-static public key (or retrieves a previously created signature) under his long-term key. Bob samples an ephemeral DH key pair if the boolean e_{DH} is set. Bob's three public keys (long-term, semi-static with id ssid, and optionally ephemeral) and the signature are Bob's first message. We call this first message Bob's *pre-key bundle*, which he sends to the key server and is not tied to any particular peer. Second, Alice parses the message, verifies the signature, samples an ephemeral DH key herself, and computes four DH shared secrets (or three if Bob's message does not include an ephemeral key) between her keys and Bob's keys (long-term–semi-static, ephemeral–long-term, ephemeral–semi-static, ephemeral–ephemeral),

[11] A reduction can simply withhold some keys from the inner adversary.

```
KGenLT:
(ltpk_U^DH, ltsk_U^DH) ←$ DH.KGen()
(ltpk_U^SIG, ltsk_U^SIG) ←$ SIG.KGen()
return ((ltpk_U^DH, ltpk_U^SIG), (ltsk_U^DH, ltsk_U^SIG))
```

```
KGenSS:
(sspk_U^DH, sssk_U^DH) ←$ DH.KGen()
(sspk_U^KEM, sssk_U^KEM) ←$ KEM.KGen()
return ((sspk_U^DH, sspk_U^KEM), (sssk_U^DH, sssk_U^KEM))
```

Alice **Bob**

```
Bob:
Run(ltsk_B, ltpk⃗, sssk⃗_B, sspk⃗, π_B, m_0, μ_0)
(create, (ssid, e_DH, e_KEM)) ← m_0
π_B.pid ← ⋆
(sspk_B^DH, sspk_B^KEM) ← sspk_B^ssid
if σ_DH^ssid = ⊥ //saved from a previous run?
    σ_DH^ssid ←$ Sign(ltsk_B, sspk_B^DH)
if e_DH = true
    (epk_B^DH, esk_B^DH) ←$ DH.KGen()
else then epk_B^DH ← ⊥
if e_KEM = true
    (epk_B^KEM, esk_B^KEM) ←$ KEM.KGen()
    σ_KEM ←$ Sign(ltsk_B, epk_B^KEM)
else
    if σ_KEM^ssid = ⊥ //saved from a previous run?
        σ_KEM^ssid ←$ Sign(ltsk_B, sspk_B^KEM)
    σ_KEM ← σ_KEM^ssid
    epk_B^KEM ← ⊥
epk_B ← (epk_B^DH, epk_B^KEM)
m_1 ← (B, ssid, σ_DH^ssid, epk_B, σ_KEM)
return (π_B, m_1, ε)
```

m_1

```
Alice:
Run(ltsk_A, sssk⃗_A, ltpk⃗, sspk⃗, π_A, m_1, μ_1)
(epk_A^DH, esk_A^DH) ←$ DH.KGen()
(B, ssid, σ_DH^ssid, epk_B, σ_KEM) ← m_1
(sspk_B^DH, sspk_B^KEM) ← sspk_B^ssid
(epk_B^DH, epk_B^KEM) ← epk_B
if SIG.Vf(ltpk_B, sspk_B^DH, σ_DH^ssid) = false
    return (π_A, ε, ε)
DH_1 ← DH(ltsk_A, sspk_B^DH)
DH_2 ← DH(esk_A^DH, ltpk_B)
DH_3 ← DH(esk_A^DH, sspk_B^DH)
if epk_B^DH ≠ ⊥ //ephemeral DH key present
    DH_4 ← DH(esk_A^DH, epk_B^DH)
else DH_4 ← ε
if epk_B^KEM ≠ ⊥ //ephemeral KEM key present
    if SIG.Vf(ltpk_B, epk_B^KEM, σ_KEM) = false
        return (π_A, ε, ε)
    (ct_KEM, ss) ←$ KEM.Enc(epk_B^KEM)
else //no ephemeral KEM key present
    if SIG.Vf(ltpk_B, sspk_B^KEM, σ_KEM) = false
        return (π_A, ε, ε)
    (ct_KEM, ss) ←$ KEM.Enc(sspk_B^KEM)
ms ← DH_1‖DH_2‖DH_3‖DH_4‖ss
π_A.K ← KDF(ms)
AD ← ltpk_A‖ltpk_B
ct_AE ← Enc(π_A.K, AD, μ_1)
π_A.pid ← B
m_2 ← (A, epk_A^DH, ct_AE, ct_KEM)
return (π_A, m_2, ε)
```

m_2

```
Bob:
Run(ltsk_B, sssk⃗_B, ltpk⃗, sspk⃗, π_B, m_2, μ_2)
(sssk_B^DH, sssk_B^KEM) ← sssk_B^ssid
(A, epk_A^DH, ct_AE, ct_KEM) ← m_2
DH_1 ← DH(ltpk_A, sssk_B^DH)
DH_2 ← DH(epk_A^DH, ltsk_B)
DH_3 ← DH(epk_A^DH, sssk_B^DH)
if epk_B^DH ≠ ⊥ //ephemeral DH key present
    DH_4 ← DH(epk_A^DH, esk_B^DH)
else then DH_4 ← ε
if epk_B^KEM ≠ ⊥ //ephemeral KEM key present
    ss ← KEM.Dec(esk_B^KEM, ct_KEM)
else //no ephemeral KEM key present
    ss ← KEM.Dec(sssk_B^KEM, ct_KEM)
ms ← KDF(DH_1‖DH_2‖DH_3‖DH_4‖ss)
π_B.K ← KDF(ms)
AD ← ltpk_A‖ltpk_B
μ' ← Dec(π_A.K, AD, ct_AE)
if μ' = ⊥ then π_B.K ← ⊥
π_B.pid ← A
return (π_B, ε, μ')
```

Figure 3: Signal's initial handshake protocols X3DH and PQXDH. The KEM with gray background is exclusive to PQXDH.

X3DH	PQXDH	σ	Assumption
semi-honest adv., big brother dist., 1-out-of-2			
✓ Th. 4.1	✗ Th. 4.4	👁	-
✓ Rm. 4.1	✓ Th. 4.5	👁̸	-
malicious adversaries with honest keys, 1-out-of-2			
✓ Th. 4.2	✗ Rm. 4.4	👁	K2DHA, KDF RO
✓ Rm. 4.1	✓ Th. 4.6	👁̸	K2DHA, KDF RO, KEM PA1
malicious adversaries, 1-out-of-∞			
✓ Th. 4.3	✗ Rm. 4.4	👁	EKDHA, KDF RO
✓ Rm. 4.1	✓ Th. 4.7	👁̸	EKDHA, KDF RO, KEM PA1
malicious adv, big brother dist., 1-out-of-∞			
● Rm. 4.2	● Rm. 4.5	👁̸	-

(a) Initiator (Bob) deniability

X3DH	PQXDH	σ	Assumption
malicious adv. w/ honest keys, big brother dist., 1-out-of-2			
✓ Th. 4.8		$\perp$	-
malicious adversaries, 1-out-of-∞			
✓ Th. 4.9		$\perp$	EKDHA, KDF RO
malicious adversaries, big brother distinguisher, 1-out-of-∞			
● Rm. 4.2	● Rm. 4.5	$\perp$	-

(b) Responder (Alice) deniability

Table 2: Comparing deniability of X3DH and PQXDH: Theorems (and Remarks) in the first two columns address the model in the continuous line above (indicating adversarial capabilities, big brother model, and how many people can fake a transcript, i.e. whether SK $\in \mathrm{O}_F$), the third column indicates if Fake requires a signature that the distinguisher is not aware of with 👁̸ ($\mathrm{USER}_{\mathsf{initiator},y} \in \mathrm{O}_F$ or both AuxPrep yields appropriate signatures and $\textsc{aux} \notin \mathrm{O}_D$) or a signature that the distinguisher may learn with 👁 ($\mathrm{USER}_{\mathsf{initiator},y} \in \mathrm{O}_F$ or AuxPrep yields appropriate signatures) or no signature at all with $\perp$, and the final column states the assumptions needed. We denote that a property holds by ✓, does not hold by ✗, or that we currently do not have a proof by ●. For the knowledge of DH assumptions (K2DH, EKDHA), see Definitions 2.4 and 2.5; the plaintext awareness assumption on KEM (gray background , see Definition 2.7) only applies to PQXDH.

and finally derives the session key from the DH shared secrets. She encrypts her first user message with the AEAD scheme under the session key[12] and sends this AEAD ciphertext and her ephemeral public key back to Bob. Third, Bob computes the DH shared secrets and derives the session key in the same way. He tries to decrypt the AEAD ciphertext and rejects the session key if decryption fails.

To simulate a transcript and session key, we face two challenges: obtaining all DH shared secrets and obtaining a valid signature on Bob's semi-static key. We briefly discuss our approaches for each of the two challenges, as well as the difficulties arising with big brother distinguishers, before going into the actual proofs.

4.1.1 Obtaining the DH shared secrets

For adversaries that create keys honestly, the Fake algorithm knows the necessary DH secret keys to compute all DH shared secrets itself (via the SK oracle or the semi-honest peer's randomness), as shown in Figure 5 and Figure 7 on the left-hand side. Against malicious adversaries, the Fake algorithm uses the knowledge of DH extractor (of the K2DH or EKDH assumption, see Definitions 2.4 and 2.5) to extract the remaining DH shared secrets from the adversary, as shown in Figures 5 and 6 on the right-hand side, following [VGIK20]. By assumption, if the extractor fails, no other extractor can succeed in a given time with more than a given probability. This allows Fake to set a random session key in case the extraction fails.

4.1.2 Obtaining a signature on Bob's semi-static key

Recall that the semi-static key can be used for several sessions, so neither the semi-static key nor the signature on the semi-static key are tied to a particular session. The Fake algorithm has two options (shown in Figure 4) to obtain a signature on Bob's semi-static key: First, from a pre-key bundle in the auxiliary info (the pre-key bundle may be part of a complete transcript), as previously done by [VGIK20]. Second, Fake may query the $\mathrm{USER}_{\mathsf{initiator},\perp}$ oracle (i.e. the oracle acts as initiator) to get a pre-key bundle

[12]In practice, Signal uses an elaborate key scheduling algorithm. The actual key and nonce used for encryption are deterministically derived from the session key.

including the signature on the semi-static key. Note that the Fake algorithm may use the SK oracle to learn the peer's secret key but does not have an option to learn the secret key of the session owner.

Both options rely on Fake obtaining a signature from the pre-key bundle. The first option (using aux) models that the adversary is able to eavesdrop on previous protocol executions and, hence, Fake can use aux as well. The second option (the $\textsc{User}_{\mathsf{initiator},\perp}$ oracle) models that Fake may start an independent session with the victim, i.e. anybody who can start a session with the victim can produce a transcript and session key. If Fake uses aux and aux is known to the distinguisher, then we have *public signatures.* Otherwise, we have *private signatures*, i.e. Fake knows signatures that the distinguisher does not. For X3DH either setting works. Looking ahead, this will make a difference for PQXDH.

4.1.3 Big Brother distinguishers

Ideally, we want to have deniability results against malicious adversaries in the big brother model. Against malicious adversaries, we rely on the knowledge of DH assumption, which we cannot apply in the big brother model. We discuss the challenges to show responder deniability in the big brother model in Remark 4.2.

4.1.4 Theorems for X3DH

Here we give our results for initiator deniability of X3DH. For responder deniability, we show the same results for both X3DH and PQXDH. We state them in Theorems 4.8 and 4.9.

We show that Bob can deny (*initiator deniability*) participation in a protocol run against a semi-honest Alice (*against semi-honest adversaries*, Theorem 4.1), against a malicious Alice that honestly generated her keys (*against malicious adversaries with honest keys*, Theorem 4.2), and against a malicious Alice (*against malicious adversaries*, Theorem 4.3). He does so by showing that Alice (1-*out-of*-2, i.e. anybody using Alice's secret keys, Theorems 4.1 and 4.2) or anybody (1-*out-of*-∞, Theorem 4.3) could have produced the transcript and session key herself, assuming she has previously observed a valid transcript with Bob as initiator (AuxPrep *yielding a valid pre-key bundle per* ssid *and user*). This holds even against a big brother distinguisher (*in the big brother model*, Theorem 4.1), who shares Alice's observations (aux *known to the distinguisher*, Theorems 4.1 to 4.3).

Theorem 4.1. *The* X3DH *protocol as shown in Figure 3 is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{RegHon}, \textsc{ChallHonInit}\}, \{SK\}, \{SK\textsc{s}, \textsc{aux}\})$*-oracles and auxiliary info sampled with* AuxPrep *yielding a valid pre-key bundle per* ssid *and user and* aux *known to the distinguisher, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party, and* $q_{O_A}, q_{O_F}, q_{O_D}$, $t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}$ *are arbitrary,* t_{F} *is small, and* $\epsilon_{\mathcal{D}} = 0$.

Proof. We give the Fake algorithm in Figure 4 on the left-hand side and Figure 5 on the left-hand side.

The Fake algorithm simulates Bob's pre-key bundle by sampling a fresh ephemeral key (if needed) and learns a signature on the semi-static key from the auxiliary info aux. Furthermore, to process Alice's message, the Fake algorithm learns Alice's long-term secret key from the SK oracle and Alice's ephemeral secret key esk_A^{DH} from the randomness that was previously used to create Alice's message. Using these secret keys, it can compute DH_1 (with esk_A^{DH}), and DH_2, DH_3, DH_4 (with $ltsk_A^{\mathsf{DH}}$), and in consequence the session key. Using the session key, Fake decrypts the AEAD ciphertext. If decryption fails, it aborts.

Since Fake produces a transcript and session key in the same way as Run, the distinguisher cannot have an advantage in winning his game, even if the distinguisher has access to all long-term and semi-static secret keys. □

Remark 4.1 (Obtaining the signature)**.** *In Theorems 4.1 to 4.3, the* Fake *algorithm obtains the signature on the semi-static DH key from the auxiliary info* aux *(cp. the left-hand side of Figure 4). Instead,*

```
$\underline{\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m, \mu, \mathsf{aux}, r_C)\text{:}}$
$(\mathsf{create}, (\mathsf{ssid}, e_{\mathsf{DH}}, e_{\mathsf{KEM}})) \leftarrow m$
$\pi.\mathsf{pid} \leftarrow \star$
$(\sigma_{\mathsf{DH}}, \sigma^{\mathsf{ssid}}_{\mathsf{KEM}}) \leftarrow$ from $\mathsf{aux}$ for $sspk^{\mathsf{ssid}}_B$
$\textbf{if } e_{\mathsf{DH}} = \mathsf{true}$
    $(epk^{\mathsf{DH}}_B, esk^{\mathsf{DH}}_B) \leftarrow\$ \mathsf{DH.KGen}()$
$\textbf{else } epk^{\mathsf{DH}}_B \leftarrow \bot$
$\textbf{if } e_{\mathsf{KEM}} = \mathsf{true}$
    $(epk^{\mathsf{KEM}}_B, \sigma_{\mathsf{KEM}}) \leftarrow$ from $\mathsf{aux}$
$\textbf{else}$
    $epk^{\mathsf{KEM}}_B \leftarrow \bot$
    $\sigma_{\mathsf{KEM}} \leftarrow \sigma^{\mathsf{ssid}}_{\mathsf{KEM}}$
$epk_B \leftarrow (epk^{\mathsf{DH}}_B, epk^{\mathsf{KEM}}_B)$
$\textbf{return } (\pi, (B, \mathsf{ssid}, \sigma^{\mathsf{ssid}}_{\mathsf{DH}}, epk_B, \sigma_{\mathsf{KEM}}), \epsilon)$
```

```
$\underline{\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m, \mu, \mathsf{aux}, r_C)\text{:}}$
$(\mathsf{create}, (\mathsf{ssid}, e_{\mathsf{DH}}, e_{\mathsf{KEM}})) \leftarrow m$
$\pi.\mathsf{pid} \leftarrow \star$
$(B, \mathsf{ssid}, \sigma^{\mathsf{ssid}}_{\mathsf{DH}}, epk_B, \sigma_{\mathsf{KEM}}) \leftarrow \textsc{User}_{\mathsf{initiator},\bot}(\pi, 1, m, \epsilon)$
$(epk^{\mathsf{DH}}_B, epk^{\mathsf{KEM}}_B) \leftarrow epk_B$
$\textbf{if } e_{\mathsf{DH}} = \mathsf{true}$
    $(epk^{\mathsf{DH}}_B, esk^{\mathsf{DH}}_B) \leftarrow\$ \mathsf{DH.KGen}()$
$\textbf{else } epk^{\mathsf{DH}}_B \leftarrow \bot$
$epk_B \leftarrow (epk^{\mathsf{DH}}_B, epk^{\mathsf{KEM}}_B)$
$\textbf{return } (\pi, (B, \mathsf{ssid}, \sigma^{\mathsf{ssid}}_{\mathsf{DH}}, epk_B, \sigma_{\mathsf{KEM}}), \epsilon)$
```

Figure 4: The Fake algorithms simulating Bob's pre-key bundle (initiator deniability) in Theorems 4.1 to 4.3 and 4.5 to 4.7.

```
$\underline{\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m = (A, epk^{\mathsf{DH}}_A, ct_{\mathsf{AE}}, ct_{\mathsf{KEM}}), \mu, \mathsf{aux}, r_C)\text{:}}$
$\pi.\mathsf{pid} \leftarrow A$
$(ltsk^{\mathsf{DH}}_A, ltsk^{\mathsf{SIG}}_A) \leftarrow$ extract from $\mathrm{SK}(\pi)$
$esk^{\mathsf{DH}}_A \leftarrow$ from $r_C$
$(sspk^{\mathsf{DH}}_B, sspk^{\mathsf{KEM}}_B) \leftarrow sspk^{\mathsf{ssid}}_B$
$DH_1 \leftarrow \mathsf{DH}(ltsk^{\mathsf{DH}}_A, sspk^{\mathsf{ssid}}_B)$
$DH_2 \leftarrow \mathsf{DH}(esk^{\mathsf{DH}}_A, ltpk^{\mathsf{DH}}_B)$
$DH_3 \leftarrow \mathsf{DH}(esk^{\mathsf{DH}}_A, sspk^{\mathsf{ssid}}_B)$
$\textbf{if } epk_B \neq \bot$ //full handshake
    $DH_4 \leftarrow \mathsf{DH}(esk^{\mathsf{DH}}_A, epk^{\mathsf{DH}}_B)$
$\textbf{else}$ //reduced handshake
    $DH_4 \leftarrow \epsilon$
$ss \leftarrow$ from $r_C$ //encapsulation against $epk^{\mathsf{KEM}}_B$ or $sspk^{\mathsf{KEM}}_B$
$\mathsf{ms} \leftarrow DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$
$\pi.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms})$
$AD \leftarrow ltpk_A \| ltpk_B$
$\mu' \leftarrow \mathsf{Dec}(\pi.\mathsf{K}, AD, ct_{\mathsf{AE}})$
$\textbf{if } \mu' = \bot \textbf{ then } \pi.\mathsf{K} \leftarrow \bot$
$\textbf{return } (\pi, \epsilon, \mu')$
```

```
$\underline{\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m = (A, epk^{\mathsf{DH}}_A, ct_{\mathsf{AE}}, ct_{\mathsf{KEM}}), \mu, \mathsf{aux}, r = (r_{\mathcal{A}}, \vec{r_C}, \vec{r_R}), Q)\text{:}}$
$\pi.\mathsf{pid} \leftarrow A$
$(ltsk^{\mathsf{DH}}_A, ltsk^{\mathsf{SIG}}_A) \leftarrow$ extract from $\mathrm{SK}(\pi)$
$esk^{\mathsf{DH}}_B \leftarrow$ from previous run
$(sspk^{\mathsf{DH}}_B, sspk^{\mathsf{KEM}}_B) \leftarrow sspk^{\mathsf{ssid}}_B$
$DH_1 \leftarrow \mathsf{DH}(ltsk^{\mathsf{DH}}_A, sspk^{\mathsf{ssid}}_B)$
$(DH_2, DH_3) \leftarrow\$ \mathcal{E}^{DH}_{\mathcal{A};(Q,\vec{r_R})}(ltpk^{\mathsf{DH}}_B, sspk^{\mathsf{ssid}}_B, \mathsf{aux}, r_{\mathcal{A}})$
$\textbf{if } epk_B \neq \bot$ //ephemeral DH key present
    $DH_4 \leftarrow \mathsf{DH}(epk^{\mathsf{DH}}_A, esk^{\mathsf{DH}}_B)$
$\textbf{else}$
    $DH_4 \leftarrow \epsilon$
$\textbf{if } epk^{\mathsf{KEM}}_B \neq \bot$ //ephemeral KEM key present
    $ss \leftarrow\$ \mathcal{E}^{PA}_{\mathcal{A};(Q,\vec{r_R})}(epk^{\mathsf{KEM}}_B, ct_{\mathsf{KEM}}, r_{\mathcal{A}})$
$\textbf{else}$ //no ephemeral KEM key present
    $ss \leftarrow\$ \mathcal{E}^{PA}_{\mathcal{A};(Q,\vec{r_R})}(sspk^{\mathsf{KEM}}_B, ct_{\mathsf{KEM}}, r_{\mathcal{A}})$
$\textbf{if } DH_2 = \bot \vee DH_3 = \bot$
    $\pi.\mathsf{K} \leftarrow\$ \{0,1\}^{256}$
$\textbf{else}$
    $\mathsf{ms} \leftarrow DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$
    $\pi.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms})$
$AD \leftarrow ltpk_A \| ltpk_B$
$\mu' \leftarrow \mathsf{Dec}(\pi.\mathsf{K}, AD, ct_{\mathsf{AE}})$
$\textbf{if } \mu' = \bot \textbf{ then } \pi.\mathsf{K} \leftarrow \bot$
$\textbf{return } (\pi, \epsilon, \mu')$
```

Figure 5: The Fake algorithm processing Alice's message (initiator deniability) in Theorems 4.1 and 4.5 (on the left-hand side) and Theorems 4.2 and 4.6 (on the right-hand side).

```
$\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m, \mu, \mathsf{aux}, r = (r_{\mathcal{A}}, \vec{r_C}, \vec{r_R}), Q)$:
$(A, epk_A^{\mathsf{DH}}, ct_{\mathsf{AE}}, ct_{\mathsf{KEM}}) \leftarrow m$
$\pi.\mathsf{pid} \leftarrow A$
$(sspk_B^{\mathsf{DH}}, sspk_B^{\mathsf{KEM}}) \leftarrow sspk_B^{\mathsf{ssid}}$
$DH_1 \leftarrow \mathcal{E}^{DH}_{\mathcal{A};(Q,\vec{r_R})}(sspk_B^{\mathsf{DH}}, \mathsf{aux}, r_{\mathcal{A}})$
$DH_2 \leftarrow \mathcal{E}^{DH}_{\mathcal{A};(Q,\vec{r_R})}(ltpk_B^{\mathsf{DH}}, \mathsf{aux}, r_{\mathcal{A}})$
$DH_3 \leftarrow \mathcal{E}^{DH}_{\mathcal{A};(Q,\vec{r_R})}(sspk_B^{\mathsf{DH}}, \mathsf{aux}, r_{\mathcal{A}})$
if $e_{\mathsf{DH}} = \mathsf{true}$  //with ephemeral DH key?
    $esk_B^{\mathsf{DH}} \leftarrow$ from previous run
    $DH_4 \leftarrow \mathsf{DH}(epk_A^{\mathsf{DH}}, esk_B^{\mathsf{DH}})$
else
    $DH_4 \leftarrow \epsilon$
if $e_{\mathsf{KEM}} = \mathsf{true}$  //with ephemeral KEM key?
    $ss \leftarrow\!\$\ \mathcal{E}^{PA}_{\mathcal{A};(Q,\vec{r_R})}(epk_B^{\mathsf{KEM}}, ct_{\mathsf{KEM}}, r_{\mathcal{A}})$
else
    $ss \leftarrow\!\$\ \mathcal{E}^{PA}_{\mathcal{A};(Q,\vec{r_R})}(sspk_B^{\mathsf{KEM}}, ct_{\mathsf{KEM}}, r_{\mathcal{A}})$
if $DH_1 = \bot \lor DH_2 = \bot \lor DH_3 = \bot$
    $\pi.\mathsf{K} \leftarrow\!\$\ \{0,1\}^{256}$
else
    $\mathsf{ms} \leftarrow DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$
    $\pi.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms})$
$AD \leftarrow ltpk_A \| ltpk_B$
$\mu' \leftarrow \mathsf{Dec}(\pi.\mathsf{K}, AD, ct_{\mathsf{AE}})$
if $\mu' = \bot$ then $\pi.\mathsf{K} \leftarrow \bot$
return $(\pi, \epsilon, \mu')$
```

Figure 6: The Fake algorithm processing Alice's message (initiator deniability) in Theorems 4.3 and 4.7.

the Fake *algorithm may also learn the signature via the* $\text{USER}_{\mathsf{initiator},\bot}$ *oracle (cp. the right-hand side of Figure 4), resulting in* partial deniability wrt. a $\text{USER}_{\mathsf{initiator},\bot}$ oracle. *Hence, these theorems also hold for* $\mathsf{AuxPrep} = \bot$ *if* O_F *additionally includes* $\text{USER}_{\mathsf{initiator},\bot}$.

The following two theorems use knowledge of DH assumptions, which [VGIK20] have introduced to show deniability of X3DH in their model. The first theorem adapts the proof of [VGIK20, Theorem 6] to our model. Since [VGIK20] considers only keys of two lifetimes (which they call long-term and ephemeral), we extrapolate to long-term, semi-static, and ephemeral keys. Their ephemeral key gets signed, just like the semi-static key in our description of X3DH. And if their ephemeral keys were used only once, then the simulator using a signature from the auxiliary info would raise the distinguisher's suspicion (since the distinguisher knows the auxiliary information as well). Hence, we consider their "ephemeral" keys as semi-static. Furthermore, they do not consider the AEAD ciphertext.

The proof of [VGIK20] relies on key registration: They argue that at the time of key registration a user needs to provide an extractable proof of knowledge of the secret key. Hence, they provide the long-term secret key of the adversary to the simulator. We model this by limiting the adversary to honestly generated keys with the REGHON oracle and by giving the Fake algorithm access to the SK oracle.[13]

Theorem 4.2. *If* AuxPrep *yields a valid pre-key bundle per* ssid *and user, the* $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$*-Knowledge of 2DH (K2DH) Assumption holds for* AuxPrep*, and* KDF *is a random oracle, then the* X3DH *protocol as shown in Figure 3 is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\text{REGHON}, \text{INIT}, \text{CHALLINIT}\}, \{SK\}, \{AUX\})$*-oracles and auxiliary inputs sampled with* AuxPrep *and* aux *known to the distinguisher, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{O_A}, q_{O_F}, q_{O_D}$ *are arbitrary, and* $t_{\mathcal{A}} \approx t_{\mathcal{A}^{DH}}$, $t_{\mathsf{F}} \approx t_{\mathcal{E}^{DH}}$, $t_{\mathcal{D}} \approx t_{\mathcal{D}^{DH}}$*, and* $\epsilon_{\mathcal{D}} \leq \epsilon_{\mathcal{D}^{DH}}$.

Proof. We give the Fake algorithm in Figure 4 on the left-hand side and Figure 5 on the right-hand side.

The Fake algorithm uses Alice's secret key from the SK oracle to compute DH_1, Bob's ephemeral secret, which it previously sampled, to compute DH_4, and the extractor from the K2DH assumption to learn DH_2 and DH_3. If the extraction fails, Fake sets a random key as session key. Using the session key, Fake decrypts the AEAD ciphertext. If decryption fails, it aborts.

Formally, the adversary $\mathcal{A}$ never outputs any value. Here, we apply the extractor $\mathcal{E}^{DH}_{\mathcal{A}}$ in line 46 to $\mathcal{A}$ querying the CHALL oracle with the message $(A, epk_A^{\mathsf{DH}}, ct_{\mathsf{AE}})$ after receiving a message from Bob with semi-static key ssid. The runtime of the Fake algorithm is dominated by the runtime of $\mathcal{E}^{DH}_{\mathcal{A}}$.

Since Fake produces the transcript in the same way as Run, it does not help the distinguisher to win. If the extractor succeeds, then Fake has computed the session key in the same way as Run, allowing no

[13]These oracles also return semi-static secret keys, which we deem in spirit consistent with the idea of [VGIK20].

advantage for the distinguisher. If the extractor fails, then the distinguisher succeeds in extracting DH_2 or DH_3 with a maximum probability of $\epsilon_{\mathcal{D}^{DH}}$. Hence, the distinguisher can distinguish the real session key from the simulated session key with a probability of at most $\epsilon_{\mathcal{D}^{DH}}$. □

Theorem 4.3. *If* AuxPrep *yields a valid pre-key bundle per* ssid *and user, the* $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$*-Extended Knowledge of DH (EKDH) Assumption holds for* AuxPrep*, and* KDF *is a random oracle, then the* X3DH *protocol as shown in Figure 3 is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}\}, \emptyset, \{\textsc{aux}\})$*-oracles and auxiliary info sampled with* AuxPrep *and* aux *known to the distinguisher, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{O_A}, q_{O_F}, q_{O_D}$ *are arbitrary, and* $t_{\mathcal{A}} \approx t_{\mathcal{A}^{DH}}$, $t_{\mathsf{F}} \approx 3 \cdot t_{\mathcal{E}^{DH}}$, $t_{\mathcal{D}} \approx t_{\mathcal{D}^{DH}}$*, and* $\epsilon_{\mathcal{D}} \leq \epsilon_{\mathcal{D}^{DH}}$.

Proof. We give the Fake algorithm in Figure 4 on the left-hand side and Figure 6.

The Fake algorithm simulates Bob's pre-key bundle by sampling a fresh ephemeral key (if needed) and learns a signature on the semi-static key from the auxiliary info aux. Furthermore, to process Alice's message the Fake algorithm uses two techniques: It relies on the extractor from the EKDH assumption to learn DH_1 through DH_3 and uses Bob's ephemeral key (from its previous run) to compute DH_4. If any of the three extractions fail, Fake sets a random key as session key; otherwise it computes the session key normally. It decrypts the AEAD ciphertext using the session key and erases the session key in case decryption fails.

Formally, the adversary $\mathcal{A}$ never outputs any value. Here, we apply the extractor $\mathcal{E}_{\mathcal{A}}^{DH}$ to those cases were $\mathcal{A}$ queries some oracle with a message: In line 67, the adversary's output refers to $\mathcal{A}$ querying the Reg oracle (with a long-term and a semi-static public key, but we only care about the long-term key); in lines 68 and 69, the adversary's output refers to $\mathcal{A}$ querying the ChallInit oracle on message $(A, epk_A^{\mathsf{DH}}, ct_{\mathsf{AE}})$. The runtime of the Fake algorithm is dominated by calling $\mathcal{E}_{\mathcal{A}}^{DH}$ three times.

The Fake algorithm produces the transcript in the same way as Run and, hence, the transcript is indistinguishable. If the extractor succeeds in all three cases, then the Fake algorithm has computed the session key in the same way as Run, i.e. indistinguishably. If the extractor fails in any of the three cases, then the distinguisher succeeds in extracting one of the shared DH secrets with a maximum probability of $\epsilon_{\mathcal{D}^{DH}}$ as per the EKDH assumption. Hence, the distinguisher can distinguish the real session key from the simulated session key with a probability of at most $\epsilon_{\mathcal{D}^{DH}}$. Note that the second and third extractor call refer to the same output of the adversary, i.e. they are not independent. Conservatively, we bound with $\epsilon_{\mathcal{D}^{DH}}$ and not $(\epsilon_{\mathcal{D}^{DH}})^3$. □

Remark 4.2 (Deniability against malicious adversaries in the big brother model)**.** *In the big brother setting, we cannot apply the EKDH assumption (since the distinguisher gets extra inputs compared to the adversary and* Fake*). Hence, if the extractor fails a big brother distinguisher may be able to compute the correct session key with probability* $> \epsilon_{\mathcal{D}^{DH}}$*. Thereby, the big brother distinguisher can tell the two cases apart. We discuss another (unsuccessful) strategy in Section 5.*

4.2 PQXDH

We review PQXDH [KS23] with the help of Figure 3. PQXDH extends X3DH by including a KEM to achieve post-quantum confidentiality. In particular, Bob's pre-key bundle includes an ephemeral KEM key (if the boolean e_{KEM} is set) or a semi-static KEM key (otherwise), and a signature on this KEM public key. Alice verifies the signature on the KEM key (and on the semi-static DH key as before) and encapsulates against Bob's KEM public key. Alice's message consists of her ephemeral DH key, the AEAD ciphertext, and the KEM ciphertext. Both parties compute the session key from the three or four DH shared secrets and the KEM shared secret, and use the session key for encryption and decryption of the AEAD ciphertext, respectively.

Note that in practice, Bob's pre-key bundle includes a semi-static KEM key only if no ephemeral KEM key is used. We model this by not using the semi-static KEM key if an ephemeral KEM key exists; though we formally unwrap it from the semi-static key.

Since PQXDH builds on top of X3DH, simulating a transcript bears the same difficulties as for X3DH with some new ones added on top: obtaining a signature for the KEM key (which may be ephemeral or semi-static) and the KEM shared secret.

4.2.1 Obtaining a signature on Bob's semi-static and ephemeral keys

We need a signature on Bob's semi-static DH key for X3DH, and also on Bob's ephemeral or semi-static KEM key for PQXDH. So we need to cover signatures on semi-static *and* ephemeral keys.

We have the same two options as for X3DH to obtain a signature: from aux and from the $\textsc{User}_{\mathsf{initiator},y}$ oracle, see Figure 4. If Fake uses aux and aux is known to the distinguisher, then we have *public signatures*. Otherwise, we have *private signatures*, i.e. Fake knows signatures that the distinguisher does not. The distinguisher can detect reuse of public signatures, see Theorem 4.4. Hence, we require private signatures, see Remark 4.4.

4.2.2 Obtaining the KEM shared secret

For responder deniability, Fake can learn the KEM shared secret by encapsulating against the KEM public key. For initiator deniability against malicious adversaries, we rely on plaintext awareness of the KEM (see Definition 2.7), following [HKKP22].[14]

4.2.3 Theorems for PQXDH

Here we give our results for initiator deniability of PQXDH and or responder deniability of both X3DH and PQXDH. We start with initiator deniability and show that faking a transcript requires a signature that the distinguisher is not aware of, i.e. a private signature. In particular, we show that Bob cannot deny (*initiator deniability*) participation in a protocol run against a semi-honest Alice (*against semi-honest adversaries*). This follows from nobody, not even Alice (1-*out-of*-2, i.e. using Alice's secret keys) being able to produce the transcript and session key herself, even assuming she has previously observed a valid transcript with Bob as initiator (AuxPrep *yielding a valid pre-key bundle per* ssid *and user*). This holds even against a big brother distinguisher (*in the big brother model*), who shares Alice's observations (aux *known to the distinguisher*).

An indistinguishable transcript contains a valid signature on the ephemeral KEM key and the distinguisher must not have seen that signature before. Since the Fake algorithm cannot get this signature from anywhere, it must be a forgery of the signature scheme.

Theorem 4.4. *If* SIG *is* $(q_{\mathsf{SIG}}, t_{\mathsf{SIG}}, \epsilon_{\mathsf{SIG}})$*-unforgeable,* KEM *has a key collision probability bounded by* γ_{coll}*, and the public key spaces of* KEM *and* DH *are disjoint, then the* PQXDH *protocol as shown in Figure 3 is* not $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{RegHon}, \textsc{ChallHonInit}\}, \{SK\}, \{SK\textsc{s}, \textsc{aux}\})$*-oracles and auxiliary inputs sampled with* AuxPrep *yielding a valid pre-key bundle per* ssid *and user and* q_{CI} *pre-key bundles per user with ephemeral KEM keys, where* $n_p \geq 2$ *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{O_A}, q_{O_F}, q_{O_D}$*,* $t_{\mathcal{A}}, t_{\mathcal{D}}$ *are arbitrary,* $q_{\mathsf{SIG}} = 2 \cdot n_{ss} + q_{CI}$*,* $t_{\mathsf{SIG}} \approx n_p \cdot (n_{ss} + 1) + q_{\mathsf{SIG}} + t_{\mathsf{F}}$*, and* $\epsilon_{\mathcal{D}} < 1 - n_{ss} \cdot \gamma_{coll} - \epsilon_{\mathsf{SIG}}$.

[14]For initiator deniability against semi-honest adversaries, Fake can learn the randomness used for encapsulation and thereby obtain the KEM shared secret.

Proof. For contradiction, we assume PQXDH was $(n_p, n_{ss}, q_{\mathrm{O}_A}, q_{\mathrm{O}_F}, q_{\mathrm{O}_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$-deniable and build a reduction to $(q_{\mathsf{SIG}}, t_{\mathsf{SIG}}, \epsilon_{\mathsf{SIG}})$-unforgeability of SIG. Note that the reduction relies on the Fake algorithm to win its unforgeability game. As such, the reduction simulates both the deniability game and the deniability adversary around the Fake algorithm. The reduction receives a challenge key $\mathsf{pk}_{\mathsf{SIG}}$ as input and proceeds to run $\mathsf{KeyPrep}(n_p, n_{ss})$ to generate key pairs for all users. Wlog. we determine two distinct users A and B. The reduction saves the long-term signing key pair of B as $(ltpk_B^{\mathsf{SIG}}, ltsk_B^{\mathsf{SIG}})$, replaces the public key with the challenge key $\mathsf{pk}_{\mathsf{SIG}}$, and saves the resulting list of public keys as $\vec{\mathsf{pk}}$. The reduction prepares the auxiliary info aux according to $\mathsf{AuxPrep}$; to produce the pre-key bundles for B, the reduction queries its $\textsc{Sign}$ oracle to obtain signatures under $\mathsf{pk}_{\mathsf{SIG}}$ for all of B's semi-static DH keys and KEM keys and for q_{CI} ephemeral KEM keys, totaling $2 \cdot n_{ss} + q_{CI}$ queries. Since the reduction controls both the (deniability) challenger and the (deniability) adversary, it acts as if the (deniability) adversary challenges B to a full handshake with A[15] while $b = 1$. The Fake algorithm has to answer for this query. If Fake queries its SK oracle, the reduction replies with $(ltsk_A, \vec{sssk}_A)$. The Fake algorithm returns a protocol message containing a signed ephemeral KEM key for B. The reduction returns this ephemeral KEM key and signature tuple $(epk_B^{\mathsf{KEM}}, \sigma_{\mathsf{KEM}})$.

The reduction essentially runs $\mathsf{KeyPrep}(n_p, n_{ss})$ (which runs key generation $n_p \cdot (n_{ss}+1)$ times), queries its $\textsc{Sign}$ oracle q_{SIG} times, and executes Fake (which has runtime t_{F}), resulting in a runtime $\approx n_p \cdot (n_{ss} + 1) + q_{\mathsf{SIG}} + t_{\mathsf{F}}$. The reduction simulates the game $\mathcal{G}^{\mathrm{O}_A, \mathrm{O}_F, \mathrm{O}_D\text{-}den}_{\mathsf{PQXDH}, \mathsf{AuxPrep}, n_p, n_{ss}}(\mathcal{A}, \mathcal{D})$ around Fake faithfully, since it can mimic the behavior of the game and answer as the game would.

The distinguisher can tell if Fake uses a signature on the ephemeral key from aux since the distinguisher knows aux itself (via the $\textsc{aux}$ oracle). We exclude this by assuming PQXDH to be deniable. The probability of epk_B^{KEM} colliding with one of B's n_{ss} KEM keys signed in $\mathsf{AuxPrep}$ is $n_{ss} \cdot \gamma_{coll}$ (and 0 to collide with a DH public key). We need to exclude this probability since these signatures were created by the $\textsc{Sign}$ oracle of the reduction. If the transcript contained an invalid signature, then the transcript could not be $(t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$-indistinguishable. Hence, the signature must be valid and on a fresh message. Therefore, the reduction wins if the distinguisher loses and we get $\epsilon_{\mathsf{SIG}} \geq 1 - n_{ss} \cdot \gamma_{coll} - \epsilon_{\mathcal{D}}$ and $\epsilon_{\mathcal{D}} \geq 1 - n_{ss} \cdot \gamma_{coll} - \epsilon_{\mathsf{SIG}}$. $\square$

Remark 4.3 (Domain separators)**.** *If signatures on the KEM public keys include a domain separator for ephemeral and semi-static use, then we do not have the loss of* $n_{ss} \cdot \gamma_{coll}$ *in the above theorem.*

Remark 4.4 (Fake needs a private signature for initiator deniability)**.** *In Theorem 4.4 the distinguisher detects* Fake *since* Fake *does not learn any "private" signatures on ephemeral keys that the distinguisher does not know. In Theorems 4.5 to 4.7, the* Fake *algorithm obtains signatures on the semi-static DH key and on the KEM key from the auxiliary info* aux *(cp. the left-hand side of Figure 4), while the distinguisher* does not, *i.e.* $\textsc{aux} \notin O_D$*. Alternatively, the* Fake *algorithm may also learn the signatures via the* $\textsc{User}_{\mathsf{initiator},\perp}$ *oracle (cp. the right-hand side of Figure 4), resulting in* partial deniability wrt. a $\textsc{User}_{\mathsf{initiator},\perp}$ oracle. *Hence, these theorems also hold for* $\mathsf{AuxPrep} = \perp$ *if* O_F *additionally includes* $\textsc{User}_{\mathsf{initiator},\perp}$*. In either case the distinguisher does not see the signature, i.e. the signature remains "private".*

Bob can deny (*initiator deniability*) participation in a protocol run against a semi-honest Alice (*against semi-honest adversaries*, Theorem 4.5), against a malicious Alice that honestly generated her keys (*against malicious adversaries with honest keys*, Theorem 4.6), and against a malicious Alice (*against malicious adversaries*, Theorem 4.7). He does so by showing that Alice (1-*out-of*-2, i.e. anybody using Alice's secret keys, Theorems 4.5 and 4.6) or anybody (1-*out-of*-∞, Theorem 4.7) could have produced the transcript and session key herself, assuming she has previously observed a valid transcript with Bob as initiator ($\mathsf{AuxPrep}$ *yielding a valid pre-key bundle per* ssid *and user and* q_{CI} *pre-key bundles per user with ephemeral KEM keys*, Theorems 4.5 to 4.7). This holds against big brother distinguishers (*in the big brother model*,

[15] via $\textsc{ChallHonInit}(B, A, (\mathsf{create}, (\mathsf{ssid} = 1, e_{\mathsf{DH}} = \mathsf{true}, e_{\mathsf{KEM}} = \mathsf{true})))$

Theorem 4.5) who do not learn Alice's observations (aux *unknown to the distinguisher*, Theorems 4.5 to 4.7).

Theorem 4.5. *The* PQXDH *protocol as shown in Figure 3 is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{RegHon}, \textsc{ChallHonInit}\}, \{SK\}, \{SKs\})$*-oracles and auxiliary inputs sampled with* AuxPrep *yielding a valid pre-key bundle per* ssid *and user and* q_{CI} *pre-key bundles per user with ephemeral KEM keys, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{CI} \in q_{O_A}$*,* $q_{O_A}, q_{O_F}, q_{O_D}$*,* $t_{\mathcal{A}}, t_{\mathcal{D}}$ *are arbitrary,* t_{F} *is small, and* $\epsilon_{\mathcal{D}} = 0$*.*

Proof. We give the Fake algorithm in Figure 4 on the left-hand side and Figure 5 on the left-hand side.

The Fake algorithm simulates Bob's pre-key bundle by sampling a fresh ephemeral DH key (if needed) and learns a signature on the semi-static DH key and on the KEM key from the auxiliary info aux. Note that the Fake algorithm may need a signature for any semi-static KEM key or a signed ephemeral key for each of the q_{CI} queries. To process Alice's message, the Fake algorithm computes the DH shared secrets as for Theorem 4.1 (using Alice's freshly sampled ephemeral secret and the long-term secret obtained from the SK oracle). Furthermore, Fake extracts the KEM shared secret from the randomness that was previously used for the encapsulation. It computes the session key honestly as well as decrypts the AEAD ciphertext and aborts if decryption fails.

Since Fake produces a transcript and session key in the same way as Run and the distinguisher cannot detect reuse of the signature on the ephemeral KEM key, the distinguisher cannot have an advantage in winning the game, even if the distinguisher has access to all long-term and semi-static secret keys. □

For the next two theorems we need plaintext awareness.

Theorem 4.6. *If* AuxPrep *yields a valid pre-key bundle per* ssid *and user and* q_{CI} *pre-key bundles per user with ephemeral KEM keys, the* $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$*-Knowledge of 2DH (K2DH) Assumption holds for* AuxPrep*,* KEM *is* $(n_k, t_{\mathcal{C}}, t_{\mathcal{E}^{PA}}, t_{\mathcal{D}^{PA}}, \epsilon_{\mathcal{D}^{PA}})$*-PA1 secure, and* KDF *is a random oracle, then the* PQXDH *protocol as shown in Figure 3 is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{RegHon}, \textsc{Init}, \textsc{ChallInit}\}, \{SK\}, \emptyset)$*-oracles and auxiliary inputs sampled with* AuxPrep*, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party, the number of keys for the PA assumption is* $n_k = n_p \cdot n_{ss} + q_{CI}$*,* $q_{CI} \in q_{O_A}$*,* $q_{O_A}, q_{O_F}, q_{O_D}$ *are arbitrary,* $t_{\mathcal{A}} \approx t_{\mathcal{C}} \approx t_{\mathcal{A}^{DH}}$*,* $t_{\mathsf{F}} \approx t_{\mathcal{E}^{PA}} + t_{\mathcal{E}^{DH}}$*,* $t_{\mathcal{D}} \approx t_{\mathcal{D}^{DH}} \approx t_{\mathcal{D}^{PA}}$*, and* $\epsilon_{\mathcal{D}} \leq \epsilon_{\mathcal{D}^{DH}} + \epsilon_{\mathcal{D}^{PA}}$*.*

Proof. We give the Fake algorithm in Figure 4 on the left-hand side and Figure 5 on the right-hand side.

The Fake algorithm simulates Bob's pre-key bundle by sampling a fresh ephemeral DH key (if needed) and learns a signature on the semi-static DH key and on the KEM key from the auxiliary info aux. Note that the Fake algorithm may need a signature for any semi-static KEM key or a signed ephemeral key for each of the q_{CI} queries. When processing Alice's message, Fake uses Alice's long-term secret key from the SK oracle and the previously sampled ephemeral key to compute DH_1 and DH_4. It extracts DH_2, DH_3 from the adversary under the K2DH assumption. Furthermore, it uses the PA1 extractor in line 52 or 54 (depending on whether an ephemeral or semi-static KEM key is used) to learn the KEM shared secret. In contrast, Run computes DH_2, DH_3 with Bob's DH secret keys and the KEM shared secret ss with KEM.Dec() and Bob's corresponding KEM secret key. If the K2DH extraction fails, Fake sets a random key as session key. Else, it computes the session key honestly. It decrypts the AEAD ciphertext and aborts if decryption fails.

In a nutshell, the Fake algorithm creates Bob's pre-key bundle virtually identical to Run.

We argue over a series of game hops that the distinguisher cannot distinguish whether Alice's message was processed by Run or Fake.

Game 0 = Run. The initial game is the original deniability game $\mathcal{G}^{\mathrm{O}_A,\mathrm{O}_F,\mathrm{O}_D\text{-}den}_{\mathsf{PQXDH},\mathsf{AuxPrep},n_p,n_{ss}}(\mathcal{A},\mathcal{D})$ with challenge $b = 0$, executing Run.

Game 1 (SK oracle). In this game Fake obtains Bob's long-term and semi-static secret key using the SK oracle, instead of receiving Bob's secret keys as input. This change is purely syntactical and not noticeable to an attacker or distinguisher. Hence, the winning probabilities for Games 0 and 1 are identical.

Game 2 (PA1 extractor). We replace the honest decapsulation of the KEM ciphertext with the PA extractor in lines 52 and 54. (In each invocation of Fake for processing Alice's message exactly one of these two lines is executed, depending on e_{DH}.)

We bound the advantage difference introduced by this step by the $(n_k, t_{\mathcal{C}}, t_{\mathcal{E}^{PA}}, t_{\mathcal{D}^{PA}}, \epsilon_{\mathcal{D}^{PA}})$-PA1 assumption. We consider the reduction $\mathcal{B}$—running the adversary with its oracles, excluding calls to the Decaps oracle—as ciphertext creator $\mathcal{C}$ of the PA1 game. The reduction $\mathcal{B}$ is started with a list of KEM keys $\vec{\mathsf{pk}}_{\mathsf{KEM}}$ and explicit randomness $r_{\mathcal{C}} = (r_{KG}, r_{\mathsf{aux}}, r_{\mathcal{A}}, r_{RH}, r_C)$, where all randomness except $r_{\mathcal{A}}$ is hardwired[16].

The reduction samples $n_p + n_p \cdot n_{ss}$ DH keys using (hardwired) randomness r_{KG} and sorts them by user into $\vec{\mathsf{pk}}$. Next, it prepares the auxiliary info aux, which consists of one pre-key bundle per user and semi-static key and q_{CI} pre-key bundles per user with ephemeral KEM keys, using (hardwired) randomness r_{aux}. Then, it starts the adversary $\mathcal{A}$ on the auxiliary info aux and the list of public keys $\vec{\mathsf{pk}}$ with the (actual) randomness $r_{\mathcal{A}}$. The reduction provides the adversary with access to its oracles $\mathrm{O}_A = \{\text{RegHon}, \text{Init}, \text{ChallInit}\}$. To answer queries to the RegHon oracle the reduction uses the (hardwired) randomness r_{RH}. To answer queries to the ChallInit oracle, the reduction always uses Fake as described for the previous game with the following three changes: First, instead of sampling a new ephemeral KEM key for Bob's pre-key bundle, it uses the next KEM public key from its input list. Second, the reduction replaces lines 52 and 54 (the decapsulation or extraction of the KEM shared secrets) with a call to its own Decaps oracle of the PA1 game. Third, it uses the (hardwired) randomness r_C where needed. After the adversary terminates, the reduction outputs $(\mathsf{aux}, \vec{\mathsf{pk}}, r_{\mathcal{A}}, Q, K)$. The PA1-distinguisher $\mathcal{D}'$ then runs the deniability-distinguisher $\mathcal{D}$ on $(\mathsf{aux}, \vec{\mathsf{pk}}, r_{\mathcal{A}}, Q, K)$ and returns the same guess as the deniability-distinguisher $\mathcal{D}$.

Hence, depending on the secret bit of the PA1 game, the reduction simulates either the previous game or the current game. We include the running time of the $\mathrm{O}_A = \{\text{RegHon}, \text{Init}, \text{Chall}\}$ oracles in $t_{\mathcal{A}}$, allowing for $t_{\mathcal{A}} \approx t_{\mathcal{C}}$.

The PA1 distinguisher $\mathcal{D}'$ executes the deniability distinguisher $\mathcal{D}$, resulting in essentially the same runtime $t_{\mathcal{D}} \approx t_{\mathcal{D}^{PA}}$, and directly returns the guess of the deniability distinguisher $\mathcal{D}$. If the deniability distinguisher $\mathcal{D}$ succeeds, then the PA1 distinguisher $\mathcal{D}'$ succeeds as well and can distinguish between the current and the previous game. By the PA1 assumption, the PA1 distinguisher succeeds with an advantage $\leq \epsilon_{\mathcal{D}^{PA}}$, which also limits the advantage of the deniability distinguisher.

Game 3 = Fake (K2DH assumption). We replace the honest computations of DH_2, DH_3 with the K2DH extractor in line 46, finally resulting in Fake as described in Figure 5 on the right-hand side. We bound the advantage difference introduced by this step by the $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$-K2DH assumption.

Formally, the adversary $\mathcal{A}$ never outputs any value. Here, we apply the extractor $\mathcal{E}^{DH}_{\mathcal{A}}$ in line 46 to $\mathcal{A}$ querying the Chall oracle with the message $(A, epk^{\mathsf{DH}}_A, ct_{\mathsf{AE}}, ct_{\mathsf{KEM}})$ after receiving a message from Bob with semi-static key ssid. The runtime of the attacker in the K2DH assumption limits the runtime of the

[16]The reduction requires the randomnesses $r_{KG}, r_{\mathsf{aux}}, r_{RH}, r_C$ to simulate the deniability game. However, we view the reduction as ciphertext creator, and the PA1 extractor requires as input all randomness of the ciphertext creator. Following [HKKP22], we circumvent the extractor's need for the randomness by hardwiring it.

deniability attacker, i.e. $t_{\mathcal{A}} \approx t_{\mathcal{A}^{DH}}$. The runtime of the Fake algorithm is dominated by the runtimes of the K2DH extractor and the PA1 extractor, i.e. $t_{\mathsf{F}} \approx t_{\mathcal{E}^{PA}} + t_{\mathcal{E}^{DH}}$.

If the K2DH extractor succeeds, then Fake has computed the session key in the same way as Run, allowing no advantage in distinguishing the current game from the previous game. If the extractor fails, then Fake sets a random session key. Furthermore, any K2DH distinguisher running in time $t_{\mathcal{D}^{DH}}$ succeeds in extracting DH_2 or DH_3 with a maximum probability of $\epsilon_{\mathcal{D}^{DH}}$. Hence, the deniability distinguisher with the same limit in runtime can distinguish the previous game from the current game with a probability of at most $\epsilon_{\mathcal{D}^{DH}}$. □

Theorem 4.7. *If* $\mathsf{AuxPrep}$ *yields a valid pre-key bundle per* ssid *and user and* q_{CI} *pre-key bundles per user with ephemeral KEM keys, the* $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$*-Extended Knowledge of DH (EKDH) Assumption holds for* $\mathsf{AuxPrep}$*,* KEM *is* $(n_k, t_{\mathcal{C}}, t_{\mathcal{E}^{PA}}, t_{\mathcal{D}^{PA}}, \epsilon_{\mathcal{D}^{PA}})$*-PA1 secure, and* KDF *is a random oracle, then the* PQXDH *protocol as shown in Figure 3 is* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}\}, \emptyset, \emptyset)$*-oracles and auxiliary inputs sampled with* $\mathsf{AuxPrep}$*, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{CI} \in q_{O_A}$*,* $q_{O_A}, q_{O_F}, q_{O_D}$ *are arbitrary, and* $t_{\mathcal{A}} \approx t_{\mathcal{C}} \approx t_{\mathcal{A}^{DH}}$*,* $t_{\mathsf{F}} \approx t_{\mathcal{E}^{PA}} + 3 \cdot t_{\mathcal{E}^{DH}}$*,* $t_{\mathcal{D}} \approx t_{\mathcal{D}^{DH}} \approx t_{\mathcal{D}^{PA}}$*, and* $\epsilon_{\mathcal{D}} \leq \epsilon_{\mathcal{D}^{DH}} + \epsilon_{\mathcal{D}^{PA}}$*.*

Proof. We give the Fake algorithm in Figure 4 on the right-hand side and Figure 6.

The Fake algorithm reuses the pre-key bundle obtained from the $\textsc{User}_{\mathsf{initiator},\perp}$ oracle but replaces the ephemeral DH key with a freshly sampled key (if mandated by e_{DH}). Furthermore, to process Alice's message, the Fake algorithm requires several techniques: The Fake algorithm relies on the extractor from the EKDH assumption to learn DH_1 through DH_3 (lines 67, 68, and 69) and uses Bob's ephemeral key (from its previous run) to compute DH_4. Additionally, it utilizes the plaintext extractor for the KEM to learn the KEM shared secret (line 76 or 78). If any of the three extractions under the EKDH assumption fail, Fake sets a random key as session key. Else, it computes the session key honestly. It decrypts the AEAD ciphertext and aborts if decryption fails.

The Fake algorithm creates Bob's pre-key bundle virtually identical to Run. We argue via a series of game hops that the distinguisher cannot tell whether Alice's message was processed by Run or Fake.

Game 0 = Run. The initial game is the original deniability game $\mathcal{G}^{\mathrm{O}_A,\mathrm{O}_F,\mathrm{O}_D\text{-}den}_{\mathsf{PQXDH},\mathsf{AuxPrep},n_p,n_{ss}}(\mathcal{A}, \mathcal{D})$ with challenge $b = 0$, executing Run.

Game 1 (PA1 extractor). We replace the honest decapsulation of the KEM ciphertext with the PA extractor in lines 76 and 78. (In each invocation of Fake for processing Alice's message, exactly one of these two lines is executed, depending on e_{DH}.)

We bound the advantage difference introduced by this step by the $(n_k, t_{\mathcal{C}}, t_{\mathcal{E}^{PA}}, t_{\mathcal{D}^{PA}}, \epsilon_{\mathcal{D}^{PA}})$-PA1 assumption. We consider the reduction $\mathcal{B}$—running the adversary with its oracles, excluding calls to the $\textsc{Decaps}$ oracle—as ciphertext creator $\mathcal{C}$ of the PA1 game. The reduction $\mathcal{B}$ is started with a list of KEM keys $\vec{\mathsf{pk}}_{\mathsf{KEM}}$ and explicit randomness $r_{\mathcal{C}} = (r_{KG}, r_{\mathsf{aux}}, r_{\mathcal{A}}, r_R, r_C)$, where all randomness except $r_{\mathcal{A}}$ is hardwired[17].

The reduction samples $n_p + n_p \cdot n_{ss}$ DH keys using (hardwired) randomness r_{KG} and sorts them by user into $\vec{\mathsf{pk}}$. Next, it prepares the auxiliary info aux, which consists of one pre-key bundle per user and semi-static key and q_{CI} pre-key bundles per user with ephemeral KEM keys, using (hardwired) randomness r_{aux}. Then, it starts the adversary $\mathcal{A}$ on the auxiliary info aux and the list of public keys $\vec{\mathsf{pk}}$ with the (actual) randomness $r_{\mathcal{A}}$. The reduction provides the adversary with access to its oracles

[17]The reduction requires the randomnesses $r_{KG}, r_{\mathsf{aux}}, r_R, r_C$ to simulate the deniability game. However, we view the reduction as ciphertext creator, and the PA1 extractor requires as input all randomness of the ciphertext creator. Following [HKKP22] we circumvent the extractor's need for the randomness by hardwiring it.

$\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}\}$. To answer queries to the $\textsc{Reg}$ oracle, the reduction uses the (hardwired) randomness r_R. To answer queries to the $\textsc{ChallInit}$ oracle, the reduction always uses Fake as described for the previous game with the following three changes: First, instead of sampling a new ephemeral KEM key for Bob's pre-key bundle, it uses the next KEM public key from its input list. Second, the reduction replaces lines 76 and 78 (the decapsulation or extraction of the KEM shared secrets) with a call to its own $\textsc{Decaps}$ oracle of the PA1 game. Third, it uses the (hardwired) randomness r_C where needed. After the adversary terminates, the reduction outputs $(\mathsf{aux}, \vec{\mathsf{pk}}, r_{\mathcal{A}}, Q, K)$. The PA1-distinguisher $\mathcal{D}'$ then runs the deniability-distinguisher $\mathcal{D}$ on $(\mathsf{aux}, \vec{\mathsf{pk}}, r_{\mathcal{A}}, Q, K)$ and returns the same guess as the deniability-distinguisher $\mathcal{D}$.

Hence, depending on the secret bit of the PA1 game, the reduction simulates either the previous game or the current game. We include the running time of the $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}\}$ oracles in $t_{\mathcal{A}}$, allowing for $t_{\mathcal{A}} \approx t_{\mathcal{C}}$.

The PA1 distinguisher $\mathcal{D}'$ executes the deniability distinguisher $\mathcal{D}$, resulting in essentially the same runtime $t_{\mathcal{D}} \approx t_{\mathcal{D}^{PA}}$, and directly returns the guess of the deniability distinguisher $\mathcal{D}$. If the deniability distinguisher $\mathcal{D}$ succeeds, then the PA1 distinguisher $\mathcal{D}'$ succeeds as well and can distinguish between the current and the previous game. By the PA1 assumption, the PA1 distinguisher succeeds with an advantage $\leq \epsilon_{\mathcal{D}^{PA}}$, which also limits the advantage of the deniability distinguisher.

Game 2 = Fake (EKDH assumption). We replace the honest computations of DH_1, DH_2, DH_3 with the EKDH extractor in lines 67, 68, and 69, finally resulting in Fake as described above. Following Theorem 4.3, we bound the advantage difference introduced by this step by the $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$-EKDH assumption.

If the EKDH extractor succeeds (in all three cases), then Fake has computed the session key in the same way as Run, and, hence, indistinguishably. If the extractor fails (in any of the three cases), then the distinguisher succeeds in extracting one of the shared DH secrets with a maximum probability of $\epsilon_{\mathcal{D}^{DH}}$ while running in time $t_{\mathcal{D}^{DH}}$. Hence, the distinguisher can distinguish the real session key from the simulated session key, i.e. the previous game from the current game, with a probability of at most $\epsilon_{\mathcal{D}^{DH}}$. The runtime of the Fake algorithm is dominated by the runtimes of the EKDH extractors and the PA1 extractor, i.e. $t_{\mathsf{F}} \approx t_{\mathcal{E}^{PA}} + 3 \cdot t_{\mathcal{E}^{DH}}$. As for Theorem 4.3, the second and third extractor call refer to the same output of the adversary, i.e. they are not independent. Conservatively, we bound with $\epsilon_{\mathcal{D}^{DH}}$ and not $(\epsilon_{\mathcal{D}^{DH}})^3$. □

For responder deniability we obtain identical guarantees for $\mathsf{X3DH}$ and PQXDH. Alice can deny (*responder deniability*) participation in a protocol run against a malicious Bob that honestly generated his keys (*against malicious adversaries with honest keys*, Theorem 4.8), and against a malicious Bob (*against malicious adversaries*, Theorem 4.9). She does so by showing that Bob (1-*out-of*-2, i.e. using Bob's secret keys, Theorem 4.8), or anybody (1-*out-of*-∞, Theorem 4.9) could have produced the transcript and session key himself. This holds even against a big brother distinguisher (*in the big brother model*, Theorem 4.8) and does not rely on eavesdropped info ($\mathsf{AuxPrep} = \bot$, Theorems 4.8 and 4.9).

Theorem 4.8. *Both the* $\mathsf{X3DH}$ *and* PQXDH *protocols as shown in Figure 3 are* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{RegHon}, \textsc{Init}, \textsc{ChallResp}\}, \{SK\}, \{SKs\})$*-oracles and auxiliary inputs sampled with* $\mathsf{AuxPrep} = \bot$*, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{O_A}, q_{O_F}, q_{O_D}$*,* $t_{\mathcal{A}}, t_{\mathcal{D}}$ *are arbitrary,* t_{F} *is small, and* $\epsilon_{\mathcal{D}} = 0$*.*

Proof. We give the Fake algorithm in Figure 7 on the left-hand side. The Fake algorithm considers the KEM parts (obtaining the KEM public key, verifying the signature on the KEM public key, encapsulating against the KEM public key, and sending a KEM ciphertext) only for PQXDH.

```
$\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m = (B, \mathsf{ssid}, \sigma_{\mathsf{DH}}^{\mathsf{ssid}}, epk_B, \sigma_{\mathsf{KEM}}), \mu, \mathsf{aux}, r, Q)$:
$\pi.\mathsf{pid} \leftarrow B$
$sssk_B^{\mathsf{ssid}} \leftarrow$ extract from $\mathrm{SK}(\pi)$
$(sssk_B^{\mathsf{DH}}, sssk_B^{\mathsf{KEM}}) \leftarrow sssk_B^{\mathsf{ssid}}$
$(epk_A^{\mathsf{DH}}, esk_A^{\mathsf{DH}}) \leftarrow\$ \mathsf{DH.KGen}()$
$(sspk_B^{\mathsf{DH}}, sspk_B^{\mathsf{KEM}}) \leftarrow sspk_B^{\mathsf{ssid}}$
$(epk_B^{\mathsf{DH}}, epk_B^{\mathsf{KEM}}) \leftarrow epk_B$
if $\mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, sspk_B^{\mathsf{DH}}, \sigma_{\mathsf{DH}}) = \mathsf{false}$
    return $(\pi, \epsilon, \epsilon)$
$DH_1 \leftarrow \mathsf{DH}(ltpk_A^{\mathsf{DH}}, sssk_B^{\mathsf{DH}})$
$DH_2 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, ltpk_B^{\mathsf{DH}})$
$DH_3 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, sspk_B^{\mathsf{DH}})$
if $epk_B^{\mathsf{DH}} \neq \perp$ //ephemeral DH key present
    $DH_4 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, epk_B^{\mathsf{DH}})$
else
    $DH_4 \leftarrow \epsilon$
if $epk_B^{\mathsf{KEM}} \neq \perp$ //ephemeral KEM key present
    if $\mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, epk_B^{\mathsf{KEM}}, \sigma_{\mathsf{KEM}}) = \mathsf{false}$
        return $(\pi_A, \epsilon, \epsilon)$
    $(ct_{\mathsf{KEM}}, ss) \leftarrow\$ \mathsf{KEM.Enc}(epk_B^{\mathsf{KEM}})$
else //no ephemeral KEM key present
    if $\mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, sspk_B^{\mathsf{KEM}}, \sigma_{\mathsf{KEM}}) = \mathsf{false}$
        return $(\pi_A, \epsilon, \epsilon)$
    $(ct_{\mathsf{KEM}}, ss) \leftarrow\$ \mathsf{KEM.Enc}(sspk_B^{\mathsf{KEM}})$
$\mathsf{ms} \leftarrow DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$
$\pi.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms})$
$AD \leftarrow ltpk_A \| ltpk_B$
$ct_{\mathsf{AE}} \leftarrow \mathsf{Enc}(\pi_A.\mathsf{K}, AD, \mu)$
return $(\pi, (A, epk_A^{\mathsf{DH}}, ct_{\mathsf{AE}}, ct_{\mathsf{KEM}}), \epsilon)$
```

```
$\mathsf{Fake}(\vec{\mathsf{pk}}, \pi, m = (B, \mathsf{ssid}, \sigma_{\mathsf{DH}}^{\mathsf{ssid}}, epk_B, \sigma_{\mathsf{KEM}}), \mu, \mathsf{aux}, r, Q)$:
$\pi.\mathsf{pid} \leftarrow B$
$(r_{\mathcal{A}}, \vec{r_C}, \vec{r_R}) \leftarrow r$
$(epk_A^{\mathsf{DH}}, esk_A^{\mathsf{DH}}) \leftarrow\$ \mathsf{DH.KGen}()$
$(sspk_B^{\mathsf{DH}}, sspk_B^{\mathsf{KEM}}) \leftarrow sspk_B^{\mathsf{ssid}}$
$(epk_B^{\mathsf{DH}}, epk_B^{\mathsf{KEM}}) \leftarrow epk_B$
if $\mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, sspk_B^{\mathsf{DH}}, \sigma_{\mathsf{DH}}) = \mathsf{false}$
    return $(\pi, \epsilon, \epsilon)$
$DH_1 \leftarrow \mathcal{E}^{DH}_{\mathcal{A};(Q,\vec{r_R})}(ltpk_A^{\mathsf{DH}}, \mathsf{aux}, r_{\mathcal{A}})$
$DH_2 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, ltpk_B^{\mathsf{DH}})$
$DH_3 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, sspk_B^{\mathsf{DH}})$
if $epk_B^{\mathsf{DH}} \neq \perp$ //ephemeral DH key present
    $DH_4 \leftarrow \mathsf{DH}(esk_A^{\mathsf{DH}}, epk_B^{\mathsf{DH}})$
else
    $DH_4 \leftarrow \epsilon$
if $epk_B^{\mathsf{KEM}} \neq \perp$ //ephemeral KEM key present
    if $\mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, epk_B^{\mathsf{KEM}}, \sigma_{\mathsf{KEM}}) = \mathsf{false}$
        return $(\pi_A, \epsilon, \epsilon)$
    $(ct_{\mathsf{KEM}}, ss) \leftarrow\$ \mathsf{KEM.Enc}(epk_B^{\mathsf{KEM}})$
else //no ephemeral KEM key present
    if $\mathsf{SIG.Vf}(ltpk_B^{\mathsf{SIG}}, sspk_B^{\mathsf{KEM}}, \sigma_{\mathsf{KEM}}) = \mathsf{false}$
        return $(\pi_A, \epsilon, \epsilon)$
    $(ct_{\mathsf{KEM}}, ss) \leftarrow\$ \mathsf{KEM.Enc}(sspk_B^{\mathsf{KEM}})$
if $DH_1 = \perp$
    $\pi.\mathsf{K} \leftarrow\$ \{0,1\}^{256}$
else
    $\mathsf{ms} \leftarrow DH_1 \| DH_2 \| DH_3 \| DH_4 \| ss$
    $\pi.\mathsf{K} \leftarrow \mathsf{KDF}(\mathsf{ms})$
$AD \leftarrow ltpk_A \| ltpk_B$
$ct_{\mathsf{AE}} \leftarrow \mathsf{Enc}(\pi_A.\mathsf{K}, AD, \mu)$
return $(\pi, (A, epk_A^{\mathsf{DH}}, ct_{\mathsf{AE}}, ct_{\mathsf{KEM}}), \epsilon)$
```

Figure 7: The Fake algorithms for producing Alice's message (responder deniability) in Theorem 4.8 (on the left-hand side) and Theorem 4.9 (on the right-hand side).

The Fake algorithm uses the freshly sampled ephemeral key esk_A^{DH} to compute DH_1 and Bob's long-term secret keys from the SK oracle $ltsk_B^{\mathsf{DH}}$ to compute DH_2, DH_3, DH_4. It learns the KEM shared secret from encapsulating against Bob's public key. If either signature (on the semi-static DH key or on the KEM key) does not verify, it aborts. Using those secrets, Fake computes the session key and encrypts the user message with the AEAD scheme under the session key.

Since Fake produces a transcript and session key in the same way as Run, the distinguisher cannot have an advantage in winning his game, even if the distinguisher has access to all long-term and semi-static secret keys. □

Theorem 4.9. *If the* $(t_{\mathcal{A}^{DH}}, t_{\mathcal{E}^{DH}}, t_{\mathcal{D}^{DH}}, \epsilon_{\mathcal{D}^{DH}})$*-Extended Knowledge of DH (EKDH) Assumption holds for* AuxPrep $= \perp$ *and* KDF *is a random oracle, then both the* X3DH *and the* PQXDH *protocols as shown in Figure 3 are* $(n_p, n_{ss}, q_{O_A}, q_{O_F}, q_{O_D}, t_{\mathcal{A}}, t_{\mathsf{F}}, t_{\mathcal{D}}, \epsilon_{\mathcal{D}})$*-deniable with respect to* $(\{\textsc{Reg}, \textsc{Init}, \textsc{ChallResp}\}, \emptyset, \emptyset)$*-oracles and auxiliary inputs sampled with* AuxPrep $= \perp$*, where* n_p *is the number of parties and* n_{ss} *the number of semi-static keys per party,* $q_{O_A}, q_{O_F}, q_{O_D}$ *are arbitrary,* $t_{\mathcal{A}} \approx t_{\mathcal{A}^{DH}}$*,* $t_{\mathsf{F}} \approx t_{\mathcal{E}^{DH}}$*,* $t_{\mathcal{D}} \approx t_{\mathcal{D}^{DH}}$*, and* $\epsilon_{\mathcal{D}} \leq \epsilon_{\mathcal{D}^{DH}}$*.*

Proof. We give the Fake algorithm in Figure 7 on the right-hand side. The Fake algorithm considers the KEM parts (obtaining the KEM public key, verifying the signature on the KEM public key, encapsulating against the KEM public key, and sending a KEM ciphertext) only for PQXDH.

The Fake algorithm uses its freshly sampled ephemeral secret key esk_A^{DH} to compute DH_2 through DH_4. It extracts DH_1 from the adversary under the EKDH assumption. As before, we interpret the adversary's query to the Reg oracle as its output and apply the extractor $\mathcal{E}_{\mathcal{A}}^{DH}$ accordingly. It learns the KEM shared secret from encapsulating against Bob's public key. If either signature (on the semi-static DH key or on the KEM key) does not verify, it aborts. If the extraction fails, Fake sets a random key as session key. Otherwise, it computes the session key honestly. It encrypts the user message with the AEAD scheme under the session key.

The Fake algorithm produces the transcript in the same way as Run and, hence, indistinguishably. If the extractor succeeds, then Fake has computed the session key in the same way as Run, and, hence, indistinguishably. If the extractor fails, then the distinguisher succeeds in extracting the shared DH secret with a maximum probability of $\epsilon_{\mathcal{D}^{DH}}$. The runtime of the Fake algorithm is dominated by the runtime of the EKDH extractor. Hence, the distinguisher can distinguish the real session key from the simulated session key with a probability of at most $\epsilon_{\mathcal{D}^{DH}}$. □

Remark 4.5 (Deniability against malicious adversaries in the big brother model)**.** *In Remark 4.2 we have pointed out challenges for showing deniability against malicious adversaries* in the big brother model *for* X3DH*. They directly apply to* PQXDH *as well.*

5 Conclusion and Future Work

Table 2 summarizes that in comparison to X3DH, PQXDH keeps the deniability guarantees for the responder but not for the initiator: simulating a transcript requires a signature that the distinguisher never sees. It remains an open question if deniability against malicious adversaries holds in the big brother model for both X3DH and PQXDH.

Observe that the AEAD ciphertext influences the assumptions needed to show deniability: Without the AEAD ciphertext, we can show that the DH assumption (and for PQXDH also plaintext awareness of the KEM) is necessary to show deniability against malicious adversaries.[18] When including the AEAD

[18] A deniability adversary can use any ciphertext creator to obtain a KEM ciphertext, add a freshly sampled DH public key, and send this as Alice's message to the challenge oracle. We can build a PA extractor by observing the RO queries of the Fake algorithm to learn the KEM shared secret, which must be indistinguishable due to deniability.

ciphertext, we cannot show the necessity of these assumptions anymore. Also, the KEM used in Signal's implementation, Kyber, does not fulfill plaintext awareness, as pointed out in [KS23].

One can consider a key awareness assumption (similar to the case of SIGMA [DGK06]) that allows extracting the AEAD key from a malicious Alice. However, on the one hand, the Fake algorithm cannot tell if an extracted AEAD key is the real session key. On the other hand, the distinguisher can tell them apart (since it knows what the adversary does). Observing the adversary's queries to the random oracle does not help either since Fake cannot compute the master secret ms. (Nor could the challenger verify if the adversary queries the random oracle on the actual master secret.) Hence, it appears this proof strategy does not work.

Our analysis treats Bob's long-term DH key and signing key separately, which is not the case in practice. A formal proof for the protocol taking this combined primitive into account is still open.

We see it as an interesting problem to examine if our deniability model is directly applicable to messaging protocols or if any modifications are needed. Cremers and Zhao [CZ24] have extended the deniability notion of [BFG+22] for key exchange to messaging protocols. Furthermore, one can extend our deniability model to group chats, see e.g. [HW21] and the efforts around the MLS standardization process. An extended deniability model for secure messaging can then be used to analyze the deniability guarantees of the whole Signal protocol, i.e. including the Double Ratchet.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback, and Marc Fischlin, Ehren Kret, Trevor Perrin, Olga Sanina, and Rolfe Schmidt for helpful discussions and feedback.

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. C.J. was (partially) funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297.

References

[BFG+20] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. Towards post-quantum security for Signal's X3DH handshake. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 404–430, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Cham, Switzerland. (Cited on page 4.)

[BFG+21] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. Cryptology ePrint Archive, Report 2021/769, 2021. (Cited on pages 2, 4, 9, 37, 39, and 40.)

[BFG+22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. Post-quantum asynchronous deniable key exchange and the Signal handshake. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 3–34, Virtual Event, March 8–11, 2022. Springer, Cham, Switzerland. (Cited on pages 2, 4, 5, 9, 14, 15, 16, 31, 37, 39, and 40.)

[BGB04] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. Off-the-record communication, or, why not to use PGP. In Vijay Atluri, Paul F. Syverson, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pages 77–84. ACM, 2004. (Cited on page 3.)

[BJK23] Karthikean Barghavan, Charlie Jacomme, and Franziskus Kiefer. Formal analysis of the PQXDH protocol, 2023. https://github.com/Inria-Prosecco/pqxdh-analysis. (Cited on page 4.)

[BJKS23] Karthikean Barghavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. An analysis of Signal's PQXDH. https://cryspen.com/post/pqxdh/, October 2023. (Cited on pages 4 and 16.)

[BMP03] Colin Boyd, Wenbo Mao, and Kenneth G. Paterson. Deniable authenticated key establishment for internet protocols. In *Security Protocols Workshop*, pages 255–271, 2003. (Cited on page 3.)

[BMP04] Colin Boyd, Wenbo Mao, and Kenneth G. Paterson. Key agreement using statically keyed authenticators. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 2004: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 248–262, Yellow Mountain, China, June 8–11, 2004. Springer Berlin Heidelberg, Germany. (Cited on page 3.)

[BMS20] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment, Second Edition*. Information Security and Cryptography. Springer, 2020. (Cited on page 3.)

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62, Jeju Island, Korea, December 5–9, 2004. Springer Berlin Heidelberg, Germany. (Cited on page 8.)

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 9–12, 1995. Springer Berlin Heidelberg, Germany. (Cited on page 8.)

[CCD$^+$17] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy*, pages 451–466, Paris, France, April 26–28, 2017. IEEE Computer Society Press. (Cited on pages 4 and 16.)

[CCH23] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real world deniability in messaging. Cryptology ePrint Archive, Report 2023/403, 2023. (Cited on page 2.)

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. (Cited on page 4.)

[CF11] Cas Cremers and Michele Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. (Cited on pages 2, 4, 13, 37, and 38.)

[CHN$^+$24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. K-waay: Fast and deniable post-quantum X3DH without ring signatures. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association. (Cited on pages 4, 37, and 40.)

[CK02] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161, Santa Barbara, CA, USA, August 18–22, 2002. Springer Berlin Heidelberg, Germany. (Cited on page 14.)

[CZ24] Cas Cremers and Mang Zhao. Secure messaging with strong compromise resilience, temporal privacy, and immediate decryption. In *2024 IEEE Symposium on Security and Privacy*, pages 2591–2609, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press. (Cited on pages 5 and 31.)

[Den06] Alexander W. Dent. The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 289–307, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany. (Cited on page 8.)

[DFG$^+$13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A cryptographic analysis of OPACITY - (extended abstract). In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 345–362, Egham, UK, September 9–13, 2013. Springer Berlin Heidelberg, Germany. (Cited on pages 2, 4, 37, and 39.)

[DG05] Mario Di Raimondo and Rosario Gennaro. New approaches for deniable authentication. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 2005: 12th Conference on Computer and Communications Security*, pages 112–121, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press. (Cited on page 3.)

[DG22] Samuel Dobson and Steven D. Galbraith. Post-quantum Signal key agreement from SIDH. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*, pages 422–450, Virtual Event, September 28–30, 2022. Springer, Cham, Switzerland. (Cited on page 4.)

[DGK05] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Secure off-the-record messaging. In Vijay Atluri, Sabrina De Capitani di Vimercati, and Roger Dingledine, editors, *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005*, pages 81–89. ACM, 2005. (Cited on page 3.)

[DGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 400–409, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. (Cited on pages 2, 3, 4, 12, 14, 16, 31, 37, 38, and 39.)

[DKSW09] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. Composability and on-line deniability of authentication. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptogra-*

phy Conference, volume 5444 of *Lecture Notes in Computer Science*, pages 146–162. Springer Berlin Heidelberg, Germany, March 15–17, 2009. (Cited on pages 3 and 4.)

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th Annual ACM Symposium on Theory of Computing*, pages 409–418, Dallas, TX, USA, May 23–26, 1998. ACM Press. (Cited on page 3.)

[FG24] Rune Fiedler and Felix Günther. Security analysis of Signal's PQXDH handshake. Cryptology ePrint Archive, Report 2024/702, 2024. (Cited on page 4.)

[FL25] Rune Fiedler and Roman Langrehr. On deniable authentication against malicious verifiers. In *Advances in Cryptology – CRYPTO 2025, Part I*, Lecture Notes in Computer Science, Santa Barbara, CA, USA, August 2025. Springer, Cham, Switzerland. (Cited on page 15.)

[HKK$^+$02] Dan Harkins, Charlie Kaufman, Tero Kivinen, Stephen Kent, and Radia Perlman. Design rationale for ikev2. Internet-Draft draft-ietf-ipsec-ikev2-rationale-00.txt, IETF Secretariat, Aug 2002. (Cited on page 3.)

[HKKP22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for Signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable. *Journal of Cryptology*, 35(3):17, July 2022. (Cited on pages 2, 4, 8, 14, 23, 26, 27, 37, and 39.)

[HLLC11] Lein Harn, Chia-Yin Lee, Changlu Lin, and Chin-Chen Chang. Fully deniable message authentication protocols preserving confidentiality. *Comput. J.*, 54(10):1688–1699, 2011. (Cited on page 14.)

[HW21] Andreas Hülsing and Fiona Johanna Weber. Epochal signatures for deniable group chats. In *2021 IEEE Symposium on Security and Privacy*, pages 1677–1695, San Francisco, CA, USA, May 24–27, 2021. IEEE Computer Society Press. (Cited on page 31.)

[JCL$^+$22] Shaoquan Jiang, Yeow Meng Chee, San Ling, Huaxiong Wang, and Chaoping Xing. A new framework for deniable secure key exchange. *Inf. Comput.*, 285(Part):104866, 2022. (Cited on pages 2, 3, 4, 13, 37, and 41.)

[Jia14] Shaoquan Jiang. Timed encryption with application to deniable key exchange. *Theor. Comput. Sci.*, 560:172–189, 2014. (Cited on pages 2, 3, 4, 13, 37, 40, and 41.)

[JS08] Shaoquan Jiang and Reihaneh Safavi-Naini. An efficient deniable key exchange protocol (extended abstract). In Gene Tsudik, editor, *FC 2008: 12th International Conference on Financial Cryptography and Data Security*, volume 5143 of *Lecture Notes in Computer Science*, pages 47–52, Cozumel, Mexico, January 28–31, 2008. Springer Berlin Heidelberg, Germany. (Cited on pages 2, 3, 4, 13, 37, and 40.)

[JW10] Shaoquan Jiang and Huaxiong Wang. Plaintext-awareness of hybrid encryption. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 57–72, San Francisco, CA, USA, March 1–5, 2010. Springer Berlin Heidelberg, Germany. (Cited on page 8.)

[Kat03] Jonathan Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 211–228, Warsaw, Poland, May 4–8, 2003. Springer Berlin Heidelberg, Germany. (Cited on page 3.)

[Kra96] Hugo Krawczyk. SKEME: a versatile secure key exchange mechanism for internet. In James T. Ellis, B. Clifford Neuman, and David M. Balenson, editors, *ISOC Network and Distributed System Security Symposium – NDSS'96*, pages 114–127, San Diego, CA, USA, February 22–23, 1996. IEEE Computer Society. (Cited on page 3.)

[KS23] Ehren Kret and Rolfe Schmidt. The PQXDH key agreement protocol, September 2023. https://signal.org/docs/specifications/pqxdh/. (Cited on pages 1, 2, 4, 7, 16, 22, and 31.)

[LLPM07] Meng-Hui Lim, Sanggon Lee, Youngho Park, and Sangjae Moon. Secure deniable authenticated key establishment for internet protocols. Cryptology ePrint Archive, Report 2007/163, 2007. (Cited on page 3.)

[MMP$^+$23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. (Cited on page 4.)

[MP02] Wenbo Mao and Kenneth G. Paterson. On the plausible deniability feature of internet protocols. unpublished, https://web.archive.org/web/20220818192033/http://www.isg.rhul.ac.uk/~kp/IKE.ps, 2002. (Cited on pages 3 and 15.)

[MP16a] Moxie Marlinspike and Trevor Perrin. The double ratchet algorithm, November 2016. https://www.signal.org/docs/specifications/doubleratchet/. (Cited on page 1.)

[MP16b] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol, November 2016. https://signal.org/docs/specifications/x3dh/. (Cited on pages 1, 2, 4, 7, and 16.)

[Nao02] Moni Naor. Deniable ring authentication. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498, Santa Barbara, CA, USA, August 18–22, 2002. Springer Berlin Heidelberg, Germany. (Cited on page 15.)

[OTR20] OTR team. OTR version 4, 2020. https://github.com/otrv4/otrv4/blob/master/otrv4.md. (Cited on page 2.)

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337, Santa Barbara, CA, USA, August 17–21, 2003. Springer Berlin Heidelberg, Germany. (Cited on pages 3, 4, and 16.)

[Per16] Trevor Perrin. The XEdDSA and VXEdDSA signature schemes, October 2016. https://signal.org/docs/specifications/xeddsa/. (Cited on page 16.)

[RMA$^+$23] Nathan Reitinger, Nathan Malkin, Omer Akgul, Michelle L. Mazurek, and Ian Miers. Is cryptographic deniability sufficient? non-expert perceptions of deniability in secure messaging. In *2023 IEEE Symposium on Security and Privacy*, pages 274–292, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press. (Cited on page 5.)

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. (Cited on page 4.)

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002: 9th Conference on Computer and Communications Security*, pages 98–107, Washington, DC, USA, November 18–22, 2002. ACM Press. (Cited on page 8.)

[SAB$^+$22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`. (Cited on page 8.)

[UDB$^+$15] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press. (Cited on pages 3, 5, and 15.)

[UG15] Nik Unger and Ian Goldberg. Deniable key exchanges for secure messaging. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1211–1223, Denver, CO, USA, October 12–16, 2015. ACM Press. (Cited on pages 3, 4, 15, and 16.)

[UG18] Nik Unger and Ian Goldberg. Improved strongly deniable authenticated key exchanges for secure messaging. *Proceedings on Privacy Enhancing Technologies*, 2018(1):21–66, January 2018. (Cited on pages 3, 4, and 16.)

[Ung21] Nik Unger. *End-to-End Encrypted Group Messaging with Insider Security.* PhD thesis, University of Waterloo, Ontario, Canada, 2021. `https://hdl.handle.net/10012/17196`. (Cited on page 15.)

[VGIK20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the Signal protocol. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 2020: 18th International Conference on Applied Cryptography and Network Security, Part II*, volume 12147 of *Lecture Notes in Computer Science*, pages 188–209, Rome, Italy, October 19–22, 2020. Springer, Cham, Switzerland. (Cited on pages 3, 4, 6, 7, 14, 18, 21, and 37.)

[YGS23] Tarun Kumar Yadav, Devashish Gosain, and Kent E. Seamons. Cryptographic deniability: A multi-perspective study of user perceptions and expectations. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023: 32nd USENIX Security Symposium*, pages 3637–3654, Anaheim, CA, USA, August 9–11, 2023. USENIX Association. (Cited on page 5.)

[YZ10] Andrew Chi-Chih Yao and Yunlei Zhao. Deniable internet key exchange. In Jianying Zhou and Moti Yung, editors, *ACNS 2010: 8th International Conference on Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 329–348, Beijing, China, June 22–25, 2010. Springer Berlin Heidelberg, Germany. (Cited on page 3.)

[YZ13] Andrew Chi-Chih Yao and Yunlei Zhao. OAKE: a new family of implicitly authenticated Diffie-Hellman protocols. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 1113–1128, Berlin, Germany, November 4–8, 2013. ACM Press. (Cited on pages 2, 3, 37, and 40.)

Type of Deniability	O_A	O_F	O_D	Comment
concurrent deniability [DGK06]	Reg, Init, Chall	$\emptyset$	aux	
partial deniability [DGK06]	Reg, Init, Chall	$\text{User}_{x,y}$	aux	$q_U = 1$
peer deniability [CF11]	Reg, RegHon, Init, Chall	$\text{User}_{x,y}$, SK	$\emptyset$	AuxPrep $= \perp$, session keys not deniable
peer-and-time deniability [CF11]	Reg, RegHon, Init, Chall	SK	$\emptyset$	AuxPrep has transcripts, session keys not deniable
deniability (against malicious adv.) [HKKP22]	Reg, Init, Chall	$\emptyset$	$\emptyset$	AuxPrep $= \perp$
deniability (against semi-honest adv.) [HKKP22]	RegHon, ChallHon	$\emptyset$	$\emptyset$	AuxPrep $= \perp$
outsider deniability [DFG+13]	ChallPassive	$\emptyset$	$\emptyset$	AuxPrep has valid transcripts
deniability [BFG+22, BFG+21]	ChallPassiveResp	SK	SKs	AuxPrep $= \perp$
HP-deniability [YZ13]	ChallPassive	$\emptyset$	$\emptyset$	AuxPrep $= \perp$
$DENY^{\mathsf{false}}$ and $DENY^{\mathsf{true}}$ [CHN+24]	ChallPassiveResp	SK	SKs	AuxPrep $= \perp$
deniability [JS08]	Reg, Init, Chall	$\emptyset$	aux	AuxPrep has secret keys
deniability [Jia14]	Reg, Init, Chall	$\emptyset$	aux	AuxPrep has secret keys, valid transcripts
deniability [JCL+22]	Reg, Init, Chall	$\emptyset$	aux	AuxPrep has secret keys, valid transcripts

Table 3: Relation between our model and prior work: How to achieve deniability notions similar to those in the literature. Challenge oracles not suffixed by a role refer to the oracles for both roles. Deniability [BFG+22, BFG+21], outsider deniability [DFG+13], $DENY^{\mathsf{false}}$ and $DENY^{\mathsf{true}}$ [CHN+24] are game-based definitions; the others are simulation-based.

A Comparing prior definitions to our model

This section discusses in more detail how our model captures the idea of prior definitions from the literature (sketched by Table 3). Keep in mind that whenever we write Chall we refer to the challenge oracles for both roles (i.e. ChallInit, ChallResp), and similarly for ChallHon and ChallHonInit, ChallHonResp.

The following deniability models are not concerned with user messages. Though, we expect that they can be adapted similar to our model to handle key exchange protocols with user messages.

A.1 Concurrent and partial deniability [DGK06]

Deniability in the context of key exchange was introduced by Di Raimondo, Gennaro and Krawczyk [DGK06], namely concurrent deniability and partial deniability. Let us first focus on *concurrent deniability* [DGK06, Definition 2]. Both definitions are simulation-based definitions, i.e. a distinguisher has to decide if it is presented with values sampled from the real distribution or the simulated distribution. In either case, key pairs for all honest users and auxiliary info (which may depend on the keys) are sampled first. On a technical note, the probability of the distinguisher succeeding is taken over the randomness in generating the keys (and consequentially the sampling of the auxiliary info) and the random coins of the adversary. Note that the distinguisher gets access to the auxiliary info as well.

In the real case, the adversary may then interact with oracles representing the honest users in an arbitrary manner; the output consists of the previously sampled public keys and auxiliary info, the randomness that the adversary used, the transcripts between the adversary and all oracles, and the session keys of all completed sessions. The astute observer notices the similarity to our game-based model with $b = 0$ for $\mathrm{O}_A = \{\text{Reg}, \text{Init}, \text{Chall}\}$ (with syntactic changes). In the simulated case, a simulator (who depends on the adversary) is given the same input as the adversary (i.e. the public keys, the auxiliary info, and the randomness that the adversary used) and needs to produce indistinguishable output. Note that the distinguisher receives the auxiliary info as well.

The proof technique for this definition suggested by [DGK06] and used thenceforth [CF11, VGIK20, HKKP22, JCL+22] is that the simulator executes the adversary and answers in place of the oracle. Subsequent work [JS08, Jia14] follows this approach and directly tasks the simulator to answer in place of the oracles. The advantage of this technique is that the final transcript then contains identically distributed messages by the adversary. Our game-based definition covers this case with $b = 1$ by using the Fake algorithm for simulation. To be more precise, the Fake algorithm of our game-based definition depends only on the currently challenged session and does not have a shared state over all invocations. The authors believe that this is also intended by [DGK06], who focus on concurrent sessions, where the simulation needs to happen regardless of other sessions. However, they need the simulator to have an overall state to manage the queries of the adversary.

Hence, we capture the idea of their concurrent deniability with $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{Chall}\}, \mathrm{O}_F = \emptyset$, $\mathrm{O}_D = \{\textsc{aux}\}$, and arbitrary $\mathsf{AuxPrep}$.

Their *partial deniability* [DGK06, Definition 3] allows the simulator to access an oracle acting as the challenged user partnered with another user. So we capture the idea of their partial deniability with $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{Chall}\}, \mathrm{O}_F = \{\textsc{User}_{x,y}\}, \mathrm{O}_D = \{\textsc{aux}\}$, arbitrary $\mathsf{AuxPrep}$, and $q_U = 1$, i.e. per session for which the Fake algorithm is queried it can start one session with the target user (who is then partnered with a new, honest user).

A.2 Peer deniability and peer-and-time deniability [CF11]

Peer deniability and peer-and-time deniability [CF11] are simulation-based notions: The former is based on partial deniability [DGK06] and the latter is a strengthening of peer deniability. Neither definition uses auxiliary info.

A.2.1 Peer deniability

Starting from partial deniability [DGK06], peer deniability [CF11, Definition 9] is conceptually close with a few important differences:

- Peer deniability does not guarantee deniability of session keys. Hence, in line 8 of Figure 2 the distinguisher does not get access to the session keys computed by the $\textsc{Chall}$ oracle.
- The simulator can access polynomially many sessions to create messages, i.e. q_U is not fixed to 1.
- They give the attacker access to a corruption oracle, which allows the attacker *and the simulator* to learn the secret key(s) of the corrupted parties. We model the corruption oracle with the $\textsc{RegHon}$ and SK oracles: The former allows the adversary to learn the secret key of a honestly generated key pair, and the latter for the Fake algorithm.[19] However, we retain the $\textsc{Reg}$ oracle from partial deniability [DGK06]. In consequence, the Fake algorithm cannot rely on the SK oracle, just like the simulator cannot rely on learning the peer's secret key from the set of corrupted parties (maybe the adversary does not corrupt the party in question).

Hence, we capture the idea of their definition by setting $\mathrm{O}_A = \{\textsc{Init}, \textsc{Reg}, \textsc{RegHon}, \textsc{Chall}\}, \mathrm{O}_F = \{\textsc{User}_{x,y}, \mathrm{SK}\}, \mathrm{O}_D = \emptyset$, and $\mathsf{AuxPrep} = \perp$. Additionally, we remove K as argument to $\mathcal{D}$ in line 8 of Figure 2.

A.2.2 Peer-and-time deniability

Peer-and-time deniability [CF11, Definition 10] strengthens peer deniability (above) by restricting the simulator (i.e. the Fake oracle): The simulator has to finish interacting with the $\textsc{User}_{x,y}$ oracle before interacting with the adversary. This ensures that the victim doesn't produce incriminating messages dependent on the adversary's input.

[19]Strictly speaking, they give all corrupted secret keys to the simulator, while we give only the peer's secret key to the Fake algorithm. However, the authors do not see how a third party's secret key would be of help.

For our game-based approach we split up the simulator into many calls to Fake, each producing a single message. Hence, we cannot directly adapt the notion of Fake first interacting with a user oracle as preparation, and replying to the adversary only afterwards. We note, however, that peer-and-time deniability does not use auxiliary info, allowing us to repurpose the auxiliary info in our game-based definition: Originally, the simulator uses some strategy to query its oracle in an arbitrary manner. We encode this strategy into the $\mathsf{AuxPrep}$ algorithm and give aux only to Fake (and not to $\mathcal{A}$).

Both approaches (the simulator having access to an oracle before interacting with the adversary and our tailored sampling of auxiliary info) provide auxiliary info to Fake that the distinguisher is not aware of. Therefore, the Fake algorithm can use that info without the distinguisher noticing.

Hence, we capture the idea of their definition by setting $\mathrm{O}_A = \{\textsc{Init}, \textsc{Reg}, \textsc{RegHon}, \textsc{Chall}\}, \mathrm{O}_F = \{\mathrm{SK}\}, \mathrm{O}_D = \emptyset$, and $\mathsf{AuxPrep}$ follows the simulator's strategy for querying $\textsc{User}_{x,y}$. Additionally, we remove K as argument to $\mathcal{D}$ in line 8 of Figure 2 and aux as argument to $\mathcal{A}$ in line 7 in Figure 2.

A.3 Deniability against malicious and semi-honest adversaries [HKKP22]

Deniability against malicious and semi-honest adversaries [HKKP22, Definition 7.1] are two simulation-based notions based on [DGK06]. Neither case allows any auxiliary information.

To capture the idea of their definition against malicious adversaries, we follow the same approach as for [DGK06] and set $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{Chall}\}, \mathrm{O}_F = \emptyset, \mathrm{O}_D = \emptyset$, and $\mathsf{AuxPrep} = \bot$. To capture the idea of their definition against semi-honest adversaries, we limit the adversary to accessing the oracles which enforce adherence to the protocol flow. Hence, we set $\mathrm{O}_A = \{\textsc{RegHon}, \textsc{ChallHon}\}, \mathrm{O}_F = \emptyset, \mathrm{O}_D = \emptyset$, and $\mathsf{AuxPrep} = \bot$.

A.4 Outsider deniability [DFG+13]

Outsider deniability [DFG+13, Definition 2.5] is a game based-notion. Their notion gives the adversary access to an execute oracle (which yields a transcript) and a test oracle, which - depending on the secret bit - yields either a real transcript and session key or simulated values. Unlike our Fake algorithm, their simulator produces the complete transcript at once. Their notion is concerned with a passive adversary and a transcript should be simulatable with public data only. They do not allow auxiliary info, nor does the distinguisher get the adversary's randomness.

Hence, to capture the idea of their definition we set $\mathrm{O}_A = \{\textsc{ChallPassive}\}, \mathrm{O}_F = \emptyset, \mathrm{O}_D = \emptyset$, and $\mathsf{AuxPrep}$ contains valid transcripts (substituting for the execute oracle). Additionally, we remove r as argument to $\mathcal{D}$ in line 8 of Figure 2.

A.5 Deniability [BFG+22, BFG+21]

Deniability [BFG+21, Definition 11] is a game-based notion of 1-out-of-2 deniability against passive adversaries in the big brother model. They tailor the deniability notion to their use case of a replacement for Signal's initial handshake: They argue that Bob's pre-key bundle does not need to be deniable since it is not bound to any peer or any session. Hence, it suffices for Alice's message to be simulatable by Bob to achieve 1-out-of-2 deniability.

Note that they consider Bob, who prepares the pre-key bundle, the responder, while we consider Bob the initiator.

Furthermore, they merge the adversary and the distinguisher into one algorithm $\mathcal{A}$. Syntactically, this has the consequence that the adversary-distinguisher already learns all secret keys while having access to the challenge oracle. Since the adversary-distinguisher is semi-honest, we are not aware how the knowledge of secret keys could help the adversary-distinguisher.

While they give a Fake algorithm, their Fake algorithm syntactically differs from ours: It directly produces a complete transcript and session key. Semantically this is equivalent for the use within our ChallPassiveResp oracle. They do not use auxiliary info.

All in all, to capture the idea of their definition we set $\mathrm{O}_A = \{\textsc{ChallPassiveResp}\}, \mathrm{O}_F = \{\mathrm{SK}\}$, $\mathrm{O}_D = \{\mathrm{SKs}\}$, and $\mathsf{AuxPrep} = \perp$.

A.6 HP-deniability [YZ13]

HP-deniability (short for honest-player-deniability) [YZ13, Definition 6.1] is a simulation-based definition that targets passive adversaries and exposes "pre-computed and stored session-states" to the distinguisher. In [YZ13] the authors stress that this ephemeral state is not necessarily equivalent to the random coins used for the session. We observe that some protocols, e.g. X3DH and PQXDH, instruct parties to delete their ephemeral secrets after use. Hence, it is not generally obvious which information should be included in this ephemeral session state, but appears to be protocol-specific.

All in all, to capture the idea of their definition we set $\mathrm{O}_A = \{\textsc{ChallPassive}\}, \mathrm{O}_F = \emptyset, \mathrm{O}_D = \emptyset$, and $\mathsf{AuxPrep} = \perp$. Additionally, we need an extra game variable to save the ephemeral session state in: It is initialized in line 1 of Figure 2, populated in line 101 at the end of the ChallPassive oracle in Figure 8, and is given to the distinguisher in line 8 of Figure 2.

A.7 $DENY^{\mathsf{false}}$ and $DENY^{\mathsf{true}}$ [CHN+24]

The $DENY^{\mathsf{false}}$ notion adapts the idea of deniability [BFG+22, BFG+21] to the formalization of [CHN+24]. Similarly, it is another game-based notion of 1-out-of-2 deniability against passive adversaries in the big brother model. The $DENY^{\mathsf{true}}$ notion extends $DENY^{\mathsf{false}}$ by additionally giving the adversary-distinguisher access to the responder's state. They remark that the $DENY^{\mathsf{false}}$ notion intuitively models the responder trying to frame the initiator, and the $DENY^{\mathsf{true}}$ notion adds that the responder, who frames the initiator, cooperates with the judge by handing over his or her ephemeral state after the protocol run. As for HP-deniability [YZ13], we remark that some protocols, e.g. X3DH and PQXDH, require users to delete ephemeral data after completing the protocol run. Hence, it appears that the ephemeral session state needs to be defined and included on a per-protocol basis. They do not use auxiliary info.

All in all, to capture the idea of their definition we set $\mathrm{O}_A = \{\textsc{ChallPassiveResp}\}, \mathrm{O}_F = \{\mathrm{SK}\}$, $\mathrm{O}_D = \{\mathrm{SKs}\}$, and $\mathsf{AuxPrep} = \perp$.

A.8 Deniability [JS08]

Deniability [JS08] is a simulation-based definition that allows adversarial corruptions. Note that adversary, simulator, and distinguisher all learn the corrupted secret keys. Hence, we model this non-adaptively by encoding the adversary's corruption strategy into AuxPrep, which then yields the corresponding secret keys to all three algorithms $\mathcal{A}$, Fake, and $\mathcal{D}$. While their definition leaves it implicit, we assume they allow the adversary to create keys maliciously.

To capture the idea of their definition we set $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}, \textsc{ChallResp}\}, \mathrm{O}_F = \emptyset$, $\mathrm{O}_D = \{\textsc{aux}\}$, and AuxPrep yields secret keys according to a given corruption strategy.

A.9 Deniability [Jia14]

Deniability [Jia14] is similar to the notion of [JS08] but additionally allows access to valid transcripts for all three algorithms $\mathcal{A}$, Fake, and $\mathcal{D}$.

```
$\textsc{ChallPassiveInit}(U, V, \mathsf{info}_{\mathsf{create}}, \vec{\mu})$ $\boxed{\textsc{ChallPassiveResp}(U, V, \mathsf{info}_{\mathsf{create}}, \vec{\mu})}$:
$\pi_U.\mathsf{oid} \leftarrow U$;    $\pi_U.\mathsf{pid} \leftarrow V$
$\pi_V.\mathsf{oid} \leftarrow V$;    $\pi_V.\mathsf{pid} \leftarrow U$
$\pi_U.\mathsf{role} \leftarrow \mathsf{initiator}$;    $\pi_V.\mathsf{role} \leftarrow \mathsf{responder}$
$m_1 \leftarrow (\mathsf{create}, \mathsf{info}_{\mathsf{create}})$
$(\mu_1, \ldots, \mu_{n_m}) \leftarrow \vec{\mu}$
for $i \in [1, 3, \ldots, n_m - 1]$ //until $n_m$ if $n_m$ is odd
  if $b = 0$
    $(\pi_U, m_{i+1}) \leftarrow\$ \mathsf{Run}(\mathsf{sk}_U, \vec{\mathsf{pk}}, \pi_U, m_i, \mu_i)$
    $(\pi_V, m_{i+2}) \leftarrow\$ \mathsf{Run}(\mathsf{sk}_V, \vec{\mathsf{pk}}, \pi_V, m_{i+1}, \mu_{i+1})$
  else
    $(\pi_U \boxed{\pi_V}, m_{i+1}) \leftarrow\$ \mathsf{Fake}^{\mathrm{O}_F}(\vec{\mathsf{pk}}, \pi_U, m_i, \mu_i, \mathsf{aux})$
    $(\pi_U \boxed{\pi_V}, m_{i+2}) \leftarrow\$ \mathsf{Fake}^{\mathrm{O}_F}(\vec{\mathsf{pk}}, \pi_U, m_{i+1}, \mu_{i+1}, \mathsf{aux})$ //all Fake invocations share a state
$Q[\pi_U] \leftarrow (m_i)_{i=1}^{n_m}$;    $K[\pi_U] \leftarrow \pi_U.K$
return $(Q[\pi_U], K[\pi_U])$
```

Figure 8: The challenge oracle for passive adversaries, split for initiators and responders.

To capture the idea of their definition we set $\mathrm{O}_A = \{\textsc{Reg}, \textsc{Init}, \textsc{ChallInit}, \textsc{ChallResp}\}$, $\mathrm{O}_F = \emptyset$, $\mathrm{O}_D = \{\textsc{aux}\}$, and AuxPrep yields secret keys according to a given corruption strategy and valid protocol transcripts.

A.10 Deniability [JCL+22]

Deniability [JCL+22] follows the ideas of [Jia14]. Additionally, they give the adversary access to a Reveal oracle, which yields the session key. We do not need this extra oracle since our model provides the session key to the adversary and distinguisher directly.

We can capture the idea of their definition as for [Jia14].

B Extending the model to passive adversaries

For the sake of completeness, we extend our model to passive adversaries by adding a ChallPassive oracle, which the adversary may have access to via O_A, given in Figure 8.

In consequence, we also need to adapt Definition 3.1 to allow ChallPassiveInit, ChallPassiveResp $\in \mathrm{O}_A$. We obtain *initiator deniability against passive adversaries* by setting $\{\textsc{ChallPassiveInit}\} = \mathrm{O}_A$, allowing at most q_C queries to the ChallPassiveInit oracle, and *responder deniability against passive adversaries* with $\{\textsc{ChallPassiveResp}\} = \mathrm{O}_A$, and simply *deniability against passive adversaries* if both hold. The ChallPassive oracles limit the adversary to passively observing transcripts, without the opportunity to actively participate in the protocol execution or learning a party's secret keys. The ChallPassive oracles semantically differ only in case the Fake algorithm gets an oracle: With $\mathrm{SK} \in \mathrm{O}_F$ the Fake algorithm gets different secret keys in the ChallPassiveInit and ChallPassiveResp oracles. Similarly, Fake gets access to different sessions with $\textsc{User}_{x,y} \in \mathrm{O}_F$.

It is easy to see that deniability against semi-honest adversaries implies deniability against passive adversaries.

C Summary of Major Changes

- **version 1.0 May 2024**: Initial release

- **version 1.1 - June 2025**:
  - give previous query-response pairs to the Fake algorithm within the CHALL oracle (malicious version only; see Definition 3.1) and to the PA1 extractor (Definition 2.7); the K2DH and EKDH extractors depend on the adversary and the previous query-response pairs (in Figures 5 to 7)
  - separate roles for the CHALLPASSIVE oracle in Appendix B
  - treatment of randomness for PA1 reductions in Theorems 4.6 and 4.7
  - explicit runtime for the Fake algorithm
  - explicitly state the auxiliary info for the K2DH and EKDH assumptions in Theorems 4.2, 4.3, 4.6 and 4.7