Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging

Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, Rolfe Schmidt

Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging

Karthikeyan Bhargavan[1], Charlie Jacomme[2], Franziskus Kiefer[1], and Rolfe Schmidt[3]

[1]Cryspen
[2]Inria Nancy Grand-Est, Université de Lorraine, LORIA, France
[3]Signal Messenger

Abstract

The Signal Messenger recently introduced a new asynchronous key agreement protocol called PQXDH (Post-Quantum Extended Diffie-Hellman) that seeks to provide post-quantum forward secrecy, in addition to the authentication and confidentiality guarantees already provided by the previous X3DH (Extended Diffie-Hellman) protocol. More precisely, PQXDH seeks to protect the confidentiality of messages against harvest-now-decrypt-later attacks.

In this work, we formally specify the PQXDH protocol and analyze its security using two formal verification tools, PROVERIF and CRYPTOVERIF. In particular, we ask whether PQXDH preserves the guarantees of X3DH, whether it provides post-quantum forward secrecy, and whether it can be securely deployed alongside X3DH. Our analysis identifies several flaws and potential vulnerabilities in the PQXDH specification, although these vulnerabilities are not exploitable in the Signal application, thanks to specific implementation choices which we describe in this paper. To prove the security of the current implementation, our analysis notably highlighted the need for an additional binding property of the KEM, which we formally define and prove for Kyber.

We collaborated with the protocol designers to develop an updated protocol specification based on our findings, where each change was formally verified and validated with a security proof. This work identifies some pitfalls that the community should be aware of when upgrading protocols to be post-quantum secure. It also demonstrates the utility of using formal verification hand-in-hand with protocol design.

1 Introduction

The impending advent of large-scale quantum computers poses an existential threat to the security of modern cryptography. As quantum algorithms can efficiently break all widely used asymmetric cryptographic schemes, a paradigm shift towards post-quantum cryptography (PQC) is required to ensure the confidentiality, integrity, and authenticity of sensitive data and communications.

The Signal messenger uses two protocols: X3DH [32] and Double Ratchet [33] (DR) to provide mutual authentication, forward secrecy, post-compromise security, and a form of deniability. These protocols are widely considered as the benchmark for end-to-end encrypted messaging and are also used in other messengers like WhatsApp, Facebook Messenger, and Skype. Both X3DH and DR rely heavily on the Diffie-Hellman (DH) construction, instantiated in the implementation using the Curve25519 elliptic curve [31]. However, this construction can be broken by a quantum computer that uses Shor's algorithm [36] to compute discrete logarithms. Consequently, any message sent using Signal today could be eventually decrypted using a quantum computer, significantly reducing the expected forward secrecy threshold.

To mitigate this risk, Signal proposed a new protocol, called PQXDH (Post-Quantum Extended Diffie-Hellman) as a first step in the post-quantum transition of the Signal messenger. The protocol uses a post-quantum key encapsulation mechanism (PQ-KEM) to fortify X3DH against future quantum adversaries. In the Signal implementation, the PQ-KEM is instantiated using Kyber [17], which is currently undergoing standardization as ML-KEM [2]. The goal of the protocol is to preserve the existing security of X3DH while also providing forward secrecy against harvest-now-decrypt-later (HNDL) adversaries who collect encrypted messages sent today with a view to decrypting them later using a quantum computer.

PQXDH is already deployed, alongside X3DH, in the Signal messenger and is used by tens of millions of users. It is one of the first post-quantum secure channel protocols to be deployed at this scale and serves as an early case study of how to integrate post-quantum crypto in real-world systems.

In this paper, we formally specify and analyze the security of the PQXDH protocol against both classical and quantum adversaries. The security properties we consider are mutual authentication and forward secrecy, but we do not analyze deniability. We use the automated symbolic protocol analysis tool PROVERIF [14] to find flaws and potential attacks in the first version of PQXDH. We suggest fixes to the protocol and then develop a security proof for the updated protocol

using the CryptoVerif prover [15]. In collaboration with Signal developers and designers, we helped incorporate these changes into the now published second version of PQXDH.

Contributions. We provide the first formal security analysis of the PQXDH [29] protocol deployed in Signal, uncovering a public key confusion attack, a KEM re-encapsulation attack, and two additional weaknesses in the protocol. We propose fixes for these attacks, resulting in a second version of the PQXDH specification [30], and we present formal security theorems for this updated version.

As a side contribution, our re-encapsulation attack also brings additional light to the current discussion over the correct security definitions for KEMs [6, 22].To prove that the instantiation of PQXDH in the current Signal implementation is secure, we needed an extra security property for KEMs, which we prove is met by Kyber, and under which PQXDH is secure. This definition is of independent interest.

To the best of our knowledge, this paper provides the first machine-checked security analysis for a real-world post-quantum cryptographic protocol. Our work identifies pitfalls in transitioning cryptographic protocols to post-quantum secure versions, and demonstrates how they can be avoided by the use of formal analysis tools during protocol design, and by deeper collaborations between verification researchers in academia and cryptographic developers in industry.

Models and reproducibility. Our formal models are available online at [12]. Standard versions of the CryptoVerif and Proverif tools are sufficient and not provided at [12].

2 PQXDH: Post-Quantum Extended Diffie-Hellman Key Agreement Protocol

The classic Signal Protocol is the composition of the X3DH handshake protocol [32] with the Double Ratchet protocol [33] for continuous key agreement. In 2023, the X3DH protocol was replaced by PQXDH [29], which we detail in this section, along with the security properties expected from the protocol. Our description is based on the standalone specifications for X3DH and PQXDH.

2.1 X3DH: Extended Triple Diffie-Hellman

We begin with a brief description of the legacy X3DH protocol before detailing the changes made by PQXDH.

The Signal Protocol is designed for asynchronous communication between agents who upload their public keys to a key distribution server. For simplicity, we present X3DH as a synchronous key exchange between two agents Alex and Blake, where Blake may be refered to as the initiator of the exchange, even though in practice, it is the key distribution server that prepares the initial protocol message.

Key Generation Each agent generates multiple key pairs:

- A long term identity key IK.
- A medium term key, called a "Signed PreKey", SPK. This key will be signed when used in the protocol and will be regularly rotated, e.g. every two days.
- A short term "One Time Prekey", OPK, that is deleted after each use, and short term ephemeral keys, EK, used within each session.

For each key pair K, we use K^{sk} to refer to the private key and K^{pk} for the public key. For the agents Alex and Blake, let IK_A and IK_B respectively denote their long term identity key pairs; Similarly, for all other keys, the subscript A denotes a key controlled by Alex and the subscript B denotes a key controlled by Blake. We assume a public-key infrastructure (PKI) so that all identity keys are known by all parties.[1]

Initiation To initiate the protocol, Blake prepares a Pre-key bundle message containing $\mathsf{SPK}_B^{\mathsf{pk}}, \mathtt{sign}(\mathsf{SPK}_B^{\mathsf{pk}}, \mathsf{IK}_B^{\mathsf{sk}})$ and optionally some One-Time Prekey $\mathsf{OPK}_B^{\mathsf{pk}}$.

Session Secret Generation After receiving those pre-keys, Alex verifies the signatures, generates their own ephemeral DH key pair EK_A, and computes some DH shared secrets:

$$DH_1 = (\mathsf{SPK}_B^{\mathsf{pk}})^{\mathsf{IK}_A^{\mathsf{sk}}} \qquad DH_2 = (\mathsf{IK}_B^{\mathsf{pk}})^{\mathsf{EK}_A^{\mathsf{sk}}}$$
$$DH_3 = (\mathsf{SPK}_B^{\mathsf{pk}})^{\mathsf{EK}_A^{\mathsf{sk}}} \qquad DH_4 = (\mathsf{OPK}_B^{\mathsf{pk}})^{\mathsf{EK}_A^{\mathsf{sk}}}$$

DH_4 is optional, and only computed if $\mathsf{OPK}_B^{\mathsf{pk}}$ was provided. Intuitively, DH_1 will authenticate Alex to Blake, and the rest are for increasing levels of confidentiality towards Blake. These three or four DH values are then concatenated and used inside a Key Derivation Function (KDF) to obtain a session key SK_A, used to encrypt a first message with an Authenticated Encryption with Additional Data (AEAD), where the associated data includes an encoding of $\mathsf{IK}_A^{\mathsf{pk}}, \mathsf{IK}_B^{\mathsf{pk}}$. Alex then sends their ephemeral DH public key $\mathsf{EK}_A^{\mathsf{pk}}$, along with the AEAD ciphertext, back to Blake.

Completing the Handshake Upon receiving Alex's message, Blake performs the symmetric DH computations, mutatis mutandis, and passes the concatenated values to the KDF to obtain the final secret, SK_B. Blake then uses this to decrypt the AEAD ciphertext. This decryption is successful if $\mathsf{SK}_A = \mathsf{SK}_B$, and the protocol session is complete.

2.2 PQXDH Design Rationale

The X3DH protocol becomes entirely insecure in the presence of an adversary with a quantum computer or an improved classical algorithm capable of computing discrete logarithms in the underlying group. Because of this, the engineers at Signal Messenger designed an update to X3DH, called PQXDH, with the following design goals:

[1] In practice, this assumption is met if we either trust the server to distribute correct identity keys, or if agents check key fingerprints out-of-band.

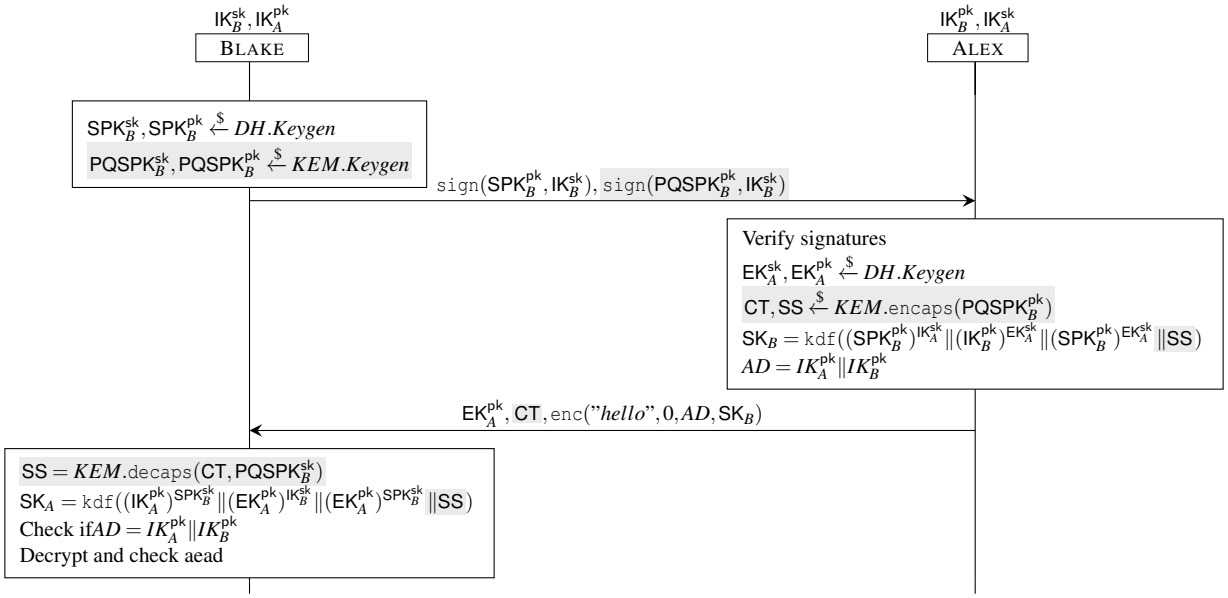

Figure 1: The PQXDH protocol (simplified).
It behaves like the X3DH protocol, with additional (PQ-)KEM computations, highlighted with a light gray background.

1. **Passive quantum adversary.** The protocol must provide security against passive attackers capable of computing discrete logarithms in the underlying group. In particular, it must prevent HNDL attacks.
2. **No security loss.** The protocol must not remove any security guarantees. In particular, the protocol must provide the same DH-based confidentiality and authentication guarantees as X3DH so that even if attacks are found on post-quantum primitives in the protocol, Signal users are no less secure.
3. **Efficient.** When the above security requirements are met, a more efficient protocol for Signal Messenger's usage patterns is preferred. This is a soft design goal, since when performance is reasonable, Signal will trade some efficiency for a security level that exceeds the design goals.

We emphasize that protection against active quantum adversaries is not a goal for PQXDH, although Signal engineers intend to provide this in a future protocol version. Also, since the Signal Protocol and Signal Messenger's implementation of it have seen extensive scrutiny they add to these a fourth, informal guideline: the new protocol should minimize changes to the existing X3DH protocol and its codebase.

Fully Post-quantum Handshake Protocols There is a growing body of research into fully post-quantum replacements for X3DH [19, 20, 25], but these protocols do both more and less than what Signal Messenger requires. They offer more by providing full post quantum security against active quantum adversaries. They offer less since they rely on entirely different cryptographic assumptions and lose all DH-based security guarantees, so these protocols do not meet the requirements without some modification.

Hybrid Protocols One way to reclaim the DH-based security would be to design a hybrid protocol that mixes DH operations into a fully post-quantum handshake protocol. While this is an interesting area of research, the costs of the post-quantum handshake protocols are significant. Since security against active quantum adversaries is not a requirement, any protocol that gives this up in exchange for more lightweight communication and computation will be preferred.

Post-quantum Primitive Selection As seen in Section 2.3, PQXDH uses a PQ-KEM to add post-quantum secure entropy into the classical protocol. While the specification does not require a particular PQ-KEM, Signal selected Kyber1024 [17] for deployment. The Kyber family of KEMs was selected because its standards-track status has both put it under extensive scrutiny and leads them to expect high quality library support in the future. In the Kyber family, the 1024-bit variant was selected as the most conservative option. Since PQXDH is designed to provide security decades into the future, they judged that a marginal additional cost in message size and client computation is a worthwhile hedge against future advances in lattice algorithms.

2.3 Protocol outline

Signal chose an approach that made minimal changes to the X3DH protocol and to their existing codebase. At a high level, PQXDH can be seen as injecting a post-quantum secure shared key into the classical X3DH protocol using a PQ-KEM. In Fig. 1 we present a simplified execution of PQXDH, where we highlight the additions to X3DH. The changes are as follows:

1. Agents also generate a KEM key pair, PQSPK.
2. In the initial message to Alex, Blake includes $\mathsf{PQSPK}_B^{\mathsf{pk}}$ and $\mathtt{sign}(\mathsf{PQSPK}_B^{\mathsf{pk}}, \mathsf{IK}_B^{\mathsf{sk}})$.
3. When computing the session secret, Alex also computes $\mathsf{CT}, \mathsf{SS} \xleftarrow{\$} \mathit{KEM}.\mathtt{encaps}(\mathsf{PQSPK}_B^{\mathsf{pk}})$ and concatenates SS to the X3DH Key Derivation Function input.
4. Alex includes the KEM ciphertext, CT, in its message.
5. Blake uses their private key to compute $\mathsf{SS} = \mathit{KEM}.\mathtt{decaps}(\mathsf{CT}, \mathsf{PQSPK}_B^{\mathsf{sk}})$ and also concatenates it to the X3DH Key Derivation Function input.

Simplifications For clarity we made several simplifications to our description of PQXDH. We provide a full version in Appendix with Fig. 4. In particular, Fig. 1 omits:

- the asynchronous behaviour of the protocol and its untrusted key distribution server;
- the encoding functions for public keys;
- the key identifiers sent in Alex's final message of the asynchronous protocol;
- the DH_4 computation based on the optional OPK;
- the distinction between short-term (or "one time") KEM keys PQOPK and medium-term (or "last resort") KEM keys PQSPK.

However, our formal models of PQXDH include all of the above except the last, and we discuss their security impact. For one-time KEM keys, we note that in the PQXDH specification, Alex has no way to distinguish between a short-term or medium-term KEM key, and an untrusted server can always choose to send medium-term keys and hold back the short-term ones. Thus, in our models and analysis we assume that all KEM keys are medium term.

2.4 Desired Security Properties

The specification does not formally define which security guarantees are meant to be provided by the protocol. However, it provides a high-level discussion on security considerations, including authentication, secrecy, identity binding, deniability, and post-quantum resistance.

We of course would like to prove resistance to harvest-now-decrypt-later (HNDL) adversaries, as it is the main design goal of PQXDH. However, since we expect that PQXDH must attain the same level of security as X3DH against classical adversaries, we also consider traditional authenticated key exchange security properties in our analysis:

- **Mutual Authentication**: if a party completes the protocol with a given public key, it is indeed communicating with the owner of this public key.
- **Forward secrecy**: keys computed between honest participants are secret, unless the long term secrets of a party were compromised before the exchange took place.
- **Resistance to key compromise impersonation**: even if the long term keys of an agent are leaked, sessions initiated by this agent are still secure.
- **Session independence**: the compromise of the short-term material of a session does not affect the security of other sessions and the compromise of medium-term material does not affect the security of sessions using different medium-term keys.
- **HNDL protection**: if an attacker suddenly has access to a quantum computer, all previously completed key exchange are still secure.

Note that we in fact consider a slightly stronger notion of HNDL protection than the one discussed in the "passive quantum adversaries" consideration of [29, Section 4.7]. Indeed, rather than a fully passive quantum attacker, we consider the security against an active attacker that at some point gets access to a quantum computer, and can then try to break the security of previously completed key-exchanges.

The PQXDH security considerations explicitly give some compromise cases leading to insecurity, while stating incompleteness. In our work, we will strive to capture precisely under which conditions the security holds.

2.5 Threat Model and Crypto Assumptions

We consider an active attacker that can intercept and modify all messages sent over the network. The security of PQXDH is expected to hold under a set of standard cryptographic assumptions about the algorithms used in the protocol:

1.A) Either gapDH is intractable for the elliptic curve X25519, which typically implies that any of the DH_x value computed in X3DH cannot be computed if the attacker as only seen the corresponding public shares;

1.B) Or the PQ-KEM (Kyber1024) is IND-CCA [16], which means that only Alex and Blake can derive the shared secret computed by the KEM;

2) And the signature Sig (XEdDSA), is EUF-CMA [18], which means that signatures are unforgeable, and ensures the authentication of the pre-key bundle, which in turn ensures that Blake knows that only Alex will be able to compute the same SK;

3) And the final AEAD (AES256 in CBC mode with HMAC and Encrypt-Then-Mac) is IND-CPA (plaintexts are secrets) and IND-CTXT (ciphertexts are integrity protected) [35], which is of course needed to ensure the secrecy of the first sent message, and is also used to authenticate Alex and Blake together;

4) And the KDF function (HKDF) is a Random Oracle;

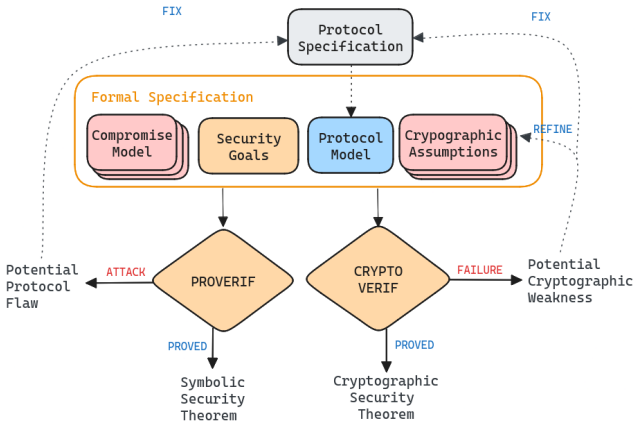

Figure 2: Protocol Verification Workflow

which ensures that if any part of the input is secret, the final key is a strong secret that is indistinguishable from a fresh random bitstring.

We also consider that the attacker may compromise some short, medium and long term keys, depending on the scenario and security property under consideration.

Assumptions 1.A) and 2) must hold against classical attackers, while 1.B), 3) and 4) are also for quantum attackers. Hence, the intuition behind the hybrid design is that 1.A+2+3+4 implies classical security today, and 1.B+2+3+4 implies long term security against future quantum attackers.

We choose the gapDH and ROM assumptions above following prior formal analyzes of X3DH [21]. However, in the quantum setting, we will not rely on the ROM, but rather on the simpler Pseudo Random Function (PRF) assumption.

3 Formal Verification Methodology

To formally analyze whether PQXDH meets its security goals, we employ a multi-faceted approach using both a symbolic protocol analysis tool and a computational proof framework. Our workflow is depicted in Figure 2. We describe each step of this workflow in sequence.

3.1 Modeling the Protocol

We begin by taking the English-language protocol specification document [29] and formalizing it in a protocol description language. Different verification tools accept different input languages. For example, the ProVerif and CryptoVerif tools accepts protocols specified in the applied pi calculus, where each role of the protocol is specified as a *process* that communicates with other protocol participants and the adversary over public *channels*. CryptoVerif also accepts protocols specified using an *oracle* syntax that is more similar to pen-and-paper cryptographic proofs.

For PQXDH, we model two roles (Alex and Blake) and specify how they process and construct the two messages in the protocol using a library of cryptographic constructions. Once the protocol messages are modeled, we define the overall scenario including the public key infrastructure, and the servers that hold the pre-keys. For PQXDH, we model:

- an arbitrary number of agents communicating together;
- a trusted PKI that provides the identity keys of clients to each other (modeled as a public database with tables);
- each device uploads Curve and KEM keys to a server which is fully untrusted (modeled as a public channel);
- each connection consists of one encrypted message from the Alex to Blake (with no follow-up messages);
- each connection can optionally use an OPK;
- we consider that all KEM keys are medium-term keys.

3.2 Threat Model

We add processes to the model that allow for certain keys to be compromised. In our model, identity keys IK may be compromised (dynamically, adaptively) at any time, allowing the adversary to then maliciously sign other keys like SPK and PQSPK and upload them to servers. For certain security goals, we also consider the compromise of short- and medium-term secrets such as OPK, EK, PQSPK, and SPK.

We also model the quantum adversary, which is done by:

- explicitly giving the attacker the power to break the non post-quantum secure primitives (i.e., allow it to compute discrete logarithms and signatures);
- or equivalently, leaking the secrets used by non post-quantum secure primitives;
- or even stronger, not making any cryptographic assumption about the elliptic curve or signature at all.

Essentially, we mark the timestamp for each compromise or quantum attack, and then consider the security for key exchanges completed before this point in time.

Note that depending on the tool and the complexity of the analysis, adding more compromise scenarios may make the analysis infeasible. For example, in CryptoVerif, we only consider the dynamic compromise of the long term identity keys. In ProVerif, thanks to its high level of automation, we are able to consider the dynamic compromise of any cryptographic material of any agent.

3.3 Security Goals

To state the security goals of the protocol we annotate the protocol model with *events* that indicate the current protocol status of each participant. Events do not affect the concrete execution of the trace and are only used to reason about it. For instance the process for Blake triggers the event:

```
event(BlakeDone(A,B,spk,pqpk,sk))
```

to indicate that an agent `B` (playing the role of Blake) has completed a PQXDH session with `A` (playing Alex), where it

used the keys corresponding to the public keys `spk` ($\mathsf{SPK}_B^{\mathsf{pk}}$) and `pqspk` ($\mathsf{PQSPK}_B^{\mathsf{pk}}$) to compute the session key `sk` (SK_B).

Mutual Authentication For authentication, we state that whenever Alex completes the protocol, Blake must also have completed the protocol with matching parameters, which is expressed in PROVERIF (simplified) as:

```
query A,B,spk,pqpk,sk,i,j;
  event(BlakeDone(A,B,spk,pqpk,sk))@i
    ⟹ event(AlexDone(A,B,spk,pqpk,sk))@j & j<i
```

This read as, for all values of the given variables, whenever the event `BlakeDone` was raised with the corresponding values at some point in time `i`, then another agent concluded with the event `AlexDone` with the same parameters and at some time point `j` before `i`. Note that in a query of this form, the variables occurring before the implication are universally quantified, and the other ones are existentially quantified. We state a similar authentication goal in the reverse direction to get mutual authentication.

Confidentiality We can state a range of confidentiality properties for PQXDH. The weakest property is that the attacker should not be able to compute the session key. Since we allow the compromise of long term keys, this query would be trivially false on our models. We then typically write a stronger query corresponding to forward secrecy, covering the fact that the session key is secret unless we compromised the long-term key *before* the session was completed. Then, also taking into account the fact that the attacker may at some point have access to a quantum computer, the post-quantum forward secrecy query is written in PROVERIF as:

```
query A,B,spk,pqpk,sk,i,j;
  event(BlakeDone(A,B,spk,pqpk,sk))@i
          ⟹ not(attacker(sk))
             | (event(LongTermComp(A))@j & j<i)
             | (event(QuantumComp)@j & j<i)
```

Here, the time point `j` represents either the compromise of the long term key of `A`, or the arrival of quantum computers. And this read as, if any agent Blake completed a session and computed key `sk`, then either the attacker cannot compute the key, or the long term key of `A` was compromised before, or the attacker had access to a quantum computer before. If the query is true, it implies that at least one of the cases must be true, which in turn implies forward secrecy, as all session keys computed before the compromise at time `j` is secret.

We can also state an even stronger property that the session key is indistinguishable from a freshly generated random key, which is what we typically prove in CRYPTOVERIF.

3.4 Symbolic Verification using PROVERIF

PROVERIF is a fully automated tool in the symbolic model, where after specifying a protocol and some security properties, it either gives back an attack or a security claim. The attacker model is weaker than the computational model, as the cryptography is assumed to be perfect. PROVERIF does attack finding, and also allows the analyst to quickly explore a set of possible scenarios or protocol variants, as well as scale better to bigger protocols, or more complex compromise scenarios.

Symbolic Cryptographic Model In the symbolic setting for PROVERIF, cryptographic primitives are assumed to be perfect by default. However, we can explicitly give additional capabilities to the attacker. To model the arrival of a sufficiently powerful quantum computer or major advances in classical algorithms, we include in our models the fact that an attacker may at some point be able to compute discrete logarithms. Likewise, to model that advances in cryptanalysis may render a KEM insecure, we include in our models the fact that an attacker may at some point be able to decapsulate KEM ciphertexts without access to the secret key. In PROVERIF, we will thus have a single monolithic attacker model, that may at some point have access to quantum computers, or may at some point be able to compromise the KEM. We will then prove security properties stating the security unless some bad case happens, either a compromise of some keys, or the compromise of one of the primitive.

Finding Attacks with PROVERIF Consider the post-quantum forward secrecy confidentiality query we stated above. If we drop the second and third cases in the disjunction, it just models a basic form of confidentiality.

In our models, this property is of course false: typically, the identity of `A` may be compromised, and in fact fully controlled by the attacker. That is, if we ask PROVERIF to prove this query, it will come up with an attack trace against it where `A` was compromised. As this is a normal and expected insecure case, we then adapt and fix the property we are trying to prove by adding this normal case to the insecure case disjunctions.

Our methodology is to rely on the high automation level of PROVERIF to iteratively do this process, and add the possible compromises that are "normal" cases, until we reach a point where either PROVERIF proves the security by validating the query, or we find an attack that does not correspond to a normal compromise case, and in fact invalidate the security.

By following this informal methodology, one can obtain a query where the disjunction of compromise cases are all independent, and where removing a single one of the cases falsifies the query. When such a query is true, it can be seen as a correct and maximal formulation of the security property: it is correct in the sense that it is met by the protocol, and maximal in the sense that removing any part of the security

property yields an attack and it thus captures that maximal variant of this security property. Note that in our methodology, one has to indeed manually check that all the cases in the disjunction are indeed independent, and that removing a single one leads to an attack. This maximality notion is not a notion of completeness with respect to the real world or the computational model, but it is one with respect to our models and the Dolev-Yao attacker. As our models do include the possible dynamic compromise of any cryptographic material used by any party, this gives us a high level of confidence in the analysis, and in the end we obtain security properties that give us an exhaustive understanding of in which cases security holds.

Interestingly, formulating this kind of property captures at once multiple flavours of the previously mentioned classical security properties for key exchanged. We typically cover in a single query the forward secrecy, session independence and HNDL resistance.

We will detail attacks found when doing this iterative process of query definition, as well as the final queries we proved on the fixed version of the protocol, in the next sections.

3.5 Cryptographic Proofs in CryptoVerif

CryptoVerif is used to make cryptographic proofs of protocols in the computational model. Such proofs are closer to the reasoning used by cryptographers, and model attackers more precisely by considering that they can be any computer running in a reasonable time. It thus gives guarantees similar to cryptographic proofs under common assumptions such as gapDH, IND-CCA or EUF-CMA. It allows an analyst to specify the original protocol as a cryptographic game as well as security properties, and then tries to automatically apply a set of valid game transformations in order to obtain a proof. It supports manual guidance of the proof, which is often needed in complex applications. Importantly, CryptoVerif was recently updated to be post-quantum sound [15], and we can use it to provide valid computational guarantees for post-quantum attackers. Overall, CryptoVerif has a stronger attacker model than ProVerif, it is less automated and does not do attack finding. Further, while the security guarantees of CryptoVerif are stronger, as they correspond to the computational model, making proofs in this model is more difficult. We sometimes have to consider fewer compromise scenarios than is possible in the highly automated ProVerif. All those elements highlight the value of combining the two approaches.

Cryptographic Assumptions In CryptoVerif, the goal is to carry out the proofs under the cryptographic assumptions from Section 2.5, one in the classical setting and another in the post-quantum one. Importantly, in the post-quantum proof, we simply do not make any assumption over the curve which is thus completely insecure, and conversely for the classical proof, no assumption is made over the KEM.

The first proof under the classical security assumptions of gapDH for the curve, EUF-CMA for Sig, IND-CPA+INT-CTXT for the AEAD and the ROM for the KDF should ensure that PQXDH is as secure as X3DH was.

The second proof, this time under classical EUF-CMA for Sig, and post-quantum IND-CCA for the KEM, and post-quantum PRF for the KDF, ensures that PQXDH does bring a new post-quantum resistance. We rely on the PRF assumption for the KDF, both because the post-quantum soundness of CryptoVerif does not capture the quantum ROM, and second because the PRF assumption is in fact more canonical and we can use it thanks to the new KEM shared secret.

Security Proofs For authentication goals, CryptoVerif follows a similar syntax to ProVerif. Consider Alex's authentication query. If we can prove this, it means that whenever Alex accepts in the cryptographic game, then with overwhelming probability Blake did accept and agrees on the given SK. A core difference with ProVerif is that we cannot consider whether some agents are compromised inside the query. Instead, we modify the protocol so that we only raise the corresponding events when the other party is honest and not yet compromised.

For secrecy, CryptoVerif enables us to check that a set of variables will always contain strongly secret values, we then similarly only set these variables in the protocol with a derived key when the two parties are honest and not yet corrupted.

Note that those approaches are more rigid than ProVerif, and in addition coming up with a proof is significantly harder in CryptoVerif, we thus focus on a smaller set of simpler properties, trying to prove the core data authentication and forward secrecy properties.

3.6 Using both ProVerif and CryptoVerif

To summarize, our global approach is twofold:

- in ProVerif we iteratively derive precise and exhaustive security properties, but in a slightly weaker attacker model than classical cryptographic proofs;
- in CryptoVerif we obtain strong guarantees in the computational model , but on a subset of the properties and possible compromises considered in ProVerif.

Our final methodology is then to use both tools in parallel. As we shall see in the following sections, each part of the analysis provided some feedback that could be used for the other part, and using both tools in combination allowed us to discover strictly more weaknesses than using then individually.

4 Protocol Flaws in the PQXDH version 1

The revision 1 of PQXDH [29] allowed for a number of implementations which would have suffered from attacks.

We classify our attacks into two categories, one for the concrete and immediate issues that developers working on post-quantum protocol migration should always have in mind, and one for the more generic design considerations and pitfalls. The immediate issues are two main attacks on the design that we uncovered, along with the justification of why it is avoided on the concrete Signal implementation. The more generic design considerations are made of additional feedback on weaknesses of the specification, which do not directly translate as attacks on one of our target security properties, but impact the overall resistance of the design.

The models used to find those attacks are available at [12]. All experiments were executed on a common laptop with 32Gb of RAM and an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz CPU. All attacks are found in a few minutes so we do not detail more the runtimes here. Expected runtimes are provided in each corresponding modeling file.

4.1 Attacks on the design

4.1.1 Public Key Confusion Attack

We discovered that nothing in the specification ensures that it is possible to distinguish between a KEM and a X25519 public key. This in turn leads to an attack where computations with one algorithm are made with public keys of another algorithm, thus falling outside its secure usages.

Attack description This theoretical attack runs as follows:

1. An attacker swaps the two signatures from Blake before forwarding them to Alex;
2. Alex computes $\mathsf{CT}, \mathsf{SS} \xleftarrow{\$} \mathit{KEM}.\mathtt{encaps}(\mathsf{SPK}_B^{\mathsf{pk}})$, using an elliptic curve public key, and then computes multiple DH exponentiation over a KEM public key.
3. Given CT and EK, the attacker can recompute all the secret values, are they all stem from insecure computations over invalid public keys for the given algorithm.

This is a theoretical attack over the design of the protocol, as it makes it impossible to prove its security under the classical IND-CCA or gapDH assumptions. Indeed, those assumptions do not explicitly specify what happens when invalid public keys are used, and in fact they imply that there is no security guarantees in such a case.

Discovery methodology Trying to prove the secrecy of the key in CryptoVerif led to a failure, where we were in fact unable to carry out the proof under the standard assumptions. We were thus forced to add a domain separation between the public-key signatures to carry out the proof. Note at this point that no attack is confirmed, as failing to do a proof does not imply that this proof cannot be made.

To fully confirm the attack, we then explicitly stated in ProVerif that KEM and DH public keys could be confused together and lead to the insecurity of the encapsulations. ProVerif then produced a concrete attack trace, where the attacker was able to compute the KEM shared secret and thus break the post-quantum security guarantees.

Practical consequences In the Signal implementation, this attack is prevented, since the key sizes of Kyber1024 and X25519 do not match, and in addition each key is prefixed by a byte identifying the algorithm. Adding this format disjointedness as an explicit condition to the specification prevents any future problems with other KEM/DH combinations, or for other independent implementations.

Interestingly, this issue illustrates how adding an extra, provably-secure component to an already-secure protocol may in fact downgrade the guarantees of the protocol. This serves as a cautionary tale for other protocols seeking to use hybrid constructions for post-quantum security.

4.1.2 KEM Re-Encapsulation Attack

The attack is based on the fact that under the classical IND-CCA assumption for KEMs, it is possible to perform so-called re-encapsulations. Consider an honest KEM encapsulation result $(\mathsf{SS}, \mathsf{CT})$ for some key pair $(\mathsf{sk}, \mathsf{pk})$, and an additional key pair $(\mathsf{sk}', \mathsf{pk}')$. If the first key pair was compromised and the attacker knows $(\mathsf{CT}, \mathsf{sk}, \mathsf{pk}')$, it may in fact be able to compute CT' such that $\mathtt{decaps}(\mathsf{CT}', \mathsf{sk}') = \mathsf{SS}$, that is, produce a valid encapsulation for pk' of the same shared secret SS. This scenario is compatible with the IND-CCA assumption, and typically arises when we use IND-CCA secure public key encryption scheme to build an IND-CCA secure KEM. This class of attacks was introduced in [22], which we find a new variant here.

Attack description Consider the following execution, fully described in Fig. 3:

1. An attacker is able to compromise some PQSPK_B of some Blake, while another PQSPK'_B of the same agent is uncompromised.
2. The attacker makes Alex use PQSPK_B, and obtain a ciphertext CT, from which it can learn the shared secret SS, as PQSPK_B was compromised.
3. Now, the attacker, not violating IND-CCA of the KEM, comes up with a new ciphertext CT, valid for PQSPK'_B, such that the decapsulation of CT' is also SS
4. The attacker then forwards to Blake the message from Alex, but swaps CT by CT', and the key identifier of PQSPK_B by PQSPK'_B.
5. Blake succeeds in computing the key using PQSPK'_B.

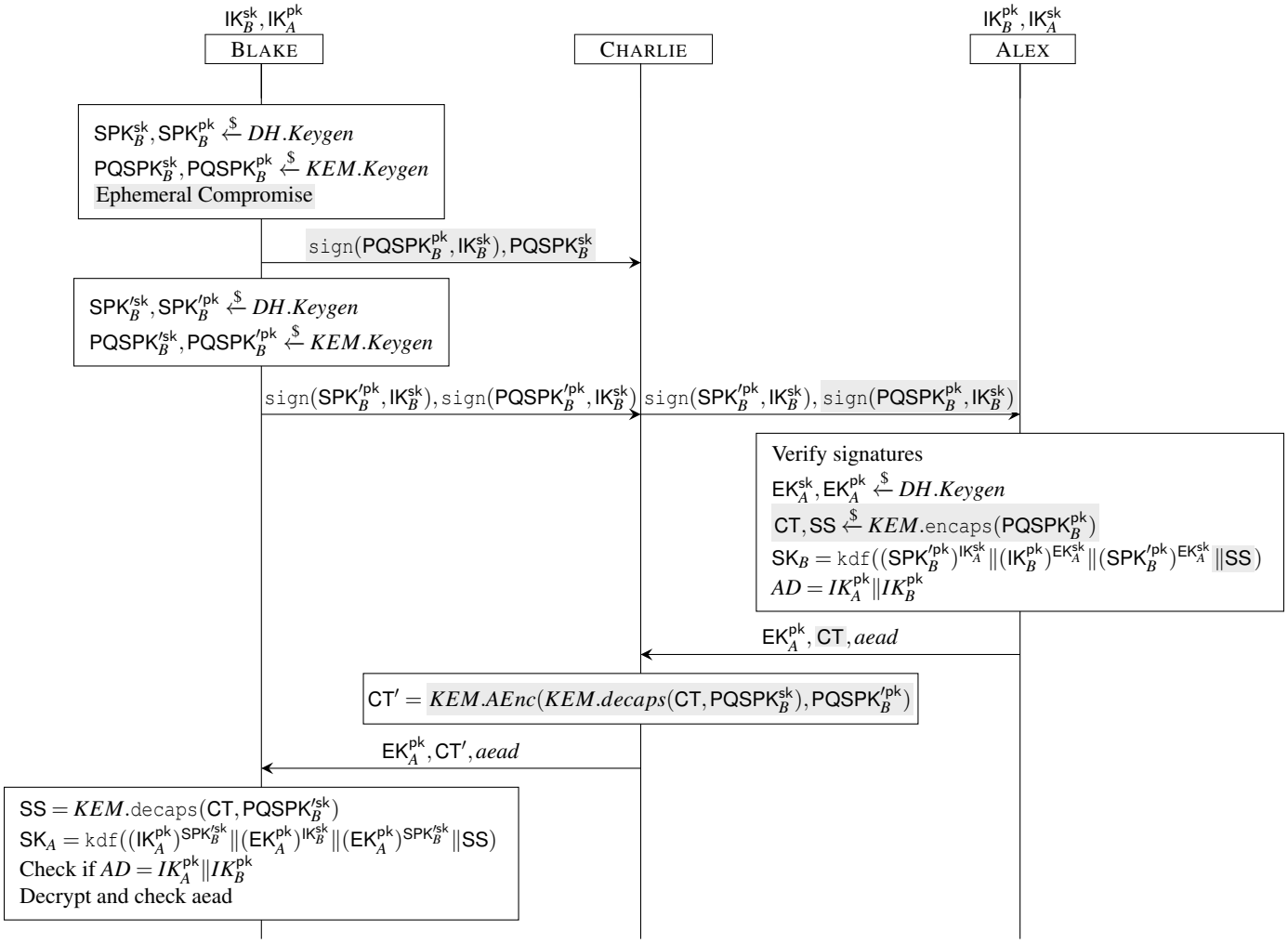

Figure 3: Re-encapsulation attack

Elements not following the standard execution are highlighted with a light gray background.

The main issue here is that the compromise of a single PQSPK_B enables an attacker to compromise all future KEM shared secrets of Blake, and this even after Blake deleted its compromised PQSPK_B. Formally, it is an attack on the session independence, and also imply that there is no mutual agreement on the KEM public key used.

Discovery As stated previously, our CRYPTOVERIF models do not include possible compromise of ephemeral values (this is a limitation to be lifted). We were thus able to complete the proof of the protocol under the classical hypotheses. However, by using PROVERIF, where in fact the basic way to model a KEM is to use an asymmetric encryption to encrypt a fresh shared secret, we could find this attack automatically. It is a surprising corner case, where the classical way to model something in the symbolic model already captures a non-trivial potential attacker capability.

Practical consequences We checked whether Kyber allowed for such re-encapsulations, which turns out not to be the case. So, while it was only formally proven to be IND-CCA secure, Kyber in fact guarantees an additional security assumptions, where the secret is tied to the corresponding public key and ciphertext.

To verify the current implementation, we had to define this additional security property, prove that it is met by Kyber, and prove that there is under this assumption agreement on the public key used in a session, effectively disabling our attack. This is detailed in Section 5.3.

Importantly, implementations of the specification that would have used a IND-CCA asymmetric encryption to build the KEM would have suffered from this. In addition, there are schemes for which the security is unclear: both BIKE and HQC do not directly tie the shared secret to the public key, but only to the ciphertext. Finally, the McEliece KEM is such that "modifying one bit in a public key has a significant chance of not affecting any particular ciphertext" [1, Section 3], and its designer conclude that "Application designers are encouraged to assume solely the standard IND-CCA2 property".

4.2 Design Considerations

4.2.1 KDF Input Confusion

At first glance, it may appear that PQXDH also has an encoding ambiguity in the format of keys used in various calls to the key derivation function (KDF) in the protocol. In fact, the 'info' field contains enough information though to disambiguate these KDF calls.

In our first PROVERIF model, we ignored the 'info' field and the tool found several cross-protocol attacks. Ignoring such seeming low-level details when modeling a protocol can sometimes miss attacks, and at other times yield false attacks.

Even after accounting for the 'info' field, there remains a concern that the format of the KDF input key material between X3DH and PQXDH can be confused with each other. For example, X3DH may use 3 or 4 DH shared secrets concatenated with each other as the KDF input, while PQXDH concatenates either 3 DH and 1 KEM secret, or 4 DH and 1 KEM secret. Within PQXDH, the 'info' field prevents confusions, but there could still theoretically be a cross-protocol attack between PQXDH (3DH+SS) and X3DH (4DH) depending on what info field the application chooses in X3DH, since the X3DH specification does not mandate the contents of its 'info' fields.

This is another interesting instance where the composition of two protocols can potentially result in attacks even if both protocols are individually secure.

4.2.2 Weak Post-Quantum Forward Secrecy

A part of the post-quantum forward secrecy guarantees of PQXDH relies on Alex receiving a signed one-time KEM public key PQOPK and the corresponding Blake deleting the corresponding private key after receiving the PQXDH message. However, since both the PQOPK and the signed last-resort KEM key PQSPK are signed with the same key, and both have the same encoded format, it is impossible for Alex to know whether they are using a last resort or one-time key.

As a consequence, PQXDH offers a weaker forward secrecy guarantee than Alex may expect, since Alex cannot know if the decryption key will be immediately deleted. In practice, however, the last-resort key PQSPK is a medium-term secret, and will be deleted by Blake after some time.

5 Formal Verification of PQXDH version 2

We detail here our provably secure update of the revision that was published by Signal. As the existing implementation does not directly follow some of our recommendations, we also verified that the existing implementation is secure by introducing a dedicated KEM assumption. Looking ahead, we also reflect on the limitations of our model, as well as propose a few possible protocol changes for a revision 3 to improve its future resilience.

The models used to obtain the proofs are also available at [12]. Similarly to the attack finding case, all proofs are completed in a few minutes.

5.1 Protocol Changes in version 2

To address all the previous points, we recommended multiple fixes that were integrated inside a new revision [30]. We provide a diff between revision 1 and revision 2 at [28] for ease of reading, with the following updates:

- To address the key-encapsulation attack, we mentioned that the PQSPK should be included inside the Authenticated Data of the AEAD, or the KEM should in fact meet an additional security property and bind together the secret and the public key ([30, Section 3.3]). As the exact security definition needed for the KEM does not have for the moment a clear consensus, we did not put any clear definition in the draft. We however in this work define our own sufficient security property, see Section 5.3. This is also detailed in the revision ([30, Section 4.12]).
- To address the key confusion attacks, we now explicitly state that the range of the encoding functions for public keys must be pairwise disjoint ([30, Section 2.1]).
- As an intermediate step (as we do not provide guarantees over this in the untrusted server setting), We added key identifiers for the PQSPK and PQOPK that enables Alex to know whether it is a one time or not pre shared key ([30, Section 2.5]).
- Our security theorems are integrated at the beginning of the Security Considerations ([30, Section 4]).
- We made explicit the missing hypothesis that the AEAD must be IND-CPA+INT-CTXT even against quantum attackers ([30, Section 4.7]).
- We added a discussion on key identifiers, not included in our presentation of the protocols, but that are used by Alex to tell Blake which key is used. Including them in our models highlighted that they are in fact not security relevant ([30, Section 4.13]).

5.2 Security Theorems proved with PROVERIF and CRYPTOVERIF

The previous modifications of the revision were proposed only after we tried them in our models. In particular, on the version where we explicitly assume that signatures of SPK and PQSPK cannot be confused and that the PQSPK_B is included inside the AD, we were able to obtain a number of security theorems.

The iterative process described previously in order for us to obtain the most precise possible security queries in PROVERIF yields complicated queries, with many possible disjunctions. Recall that such queries are correct in the sense that PQXDH does satisfy it, and it is complete in the sense that removing a single case of the disjunction leads to an attack trace, which is however a normal insecurity case. This means that this query does capture precisely the full security guarantees of PQXDH in presence of many possible compromises.

For ease of reading and clarity, we only provide a high-level theorem for all our symbolic results here. The detailed queries and corresponding exhaustive theorems are provided in Appendix A.

Theorem 1. PQXDH *in the symbolic model provides peer-authentication, forward secrecy, resistance to key compromise impersonation, session independence and resistance to "harvest now decrypt later" attacks in case of a DH break down.*

In addition, it also provides data agreement over the shared pre-key used.

Using the exhaustive analysis approach, we have an exhaustive list of compromise cases that break the security of the protocol, which leads to a couple of interesting observations:

1. The OPK does not increase the security of the key derived by Blake as much as might be expected. This is a consequence of it being unauthenticated, the attacker can then always send a malicious one instead, or more importantly never send one. Developing a way to detect if when the server does not provide an OPK there are indeed no more OPK available is an interesting challenge.
2. Compromising the SPK_B of some agent Blake allows an attacker to impersonate Blake for any other agent that could want to talk to Blake. It arises from the fact that Alex only authenticates to Blake by using a long term DH share IK_A combined with the SPK_B, an attacker compromising SPK_B can then compute the corresponding DH secret and impersonate Alex. In practice, it means that PQXDH does not meet what could be called an Ephemeral Key Compromise Impersonation.

We also obtained security theorems providing computational guarantees on the protocol with CRYPTOVERIF.

Theorem 2 (PQXDH classical computational security)**.** *If X25519 satisfies the gapDH assumption, the KDF is a Random Oracle, if the AEAD is IND-CPA+INT-CTXT and if the signature scheme is EUF-CMA, then* PQXDH *guarantees both the secrecy of the sent message, as-well-as peer-authentication with agreement over identities, OPK and SPK used, modulo the subgroup elements of X25519, as well as agreement over the PQSPK used provided it was included in the AD.*

Importantly, compared to revision 1, we do have authentication of the KEM public key used, as soon as it is in the AD. Importantly, we stress that Signal does not do this, but Kyber meets an additional property ensuring this, which we discuss in Section 5.3.

Theorem 3 (PQXDH post-quantum computational security)**.** *Under IND-CCA for the KEM, if the KDF is a PRF and the final AEAD is IND-CPA+INT-CTXT, as long as the signature scheme was unforgeable when some key exchange was completed, secrecy of the derived key still holds in the future.*

Importantly, recall that we do get guarantees against a post-quantum attacker, as CRYPTOVERIF was recently updated in order to be able to provide such guarantees [15].

5.3 KEM Public Key Agreement for PQXDH

As the re-encapsulating attack can be carried out without violating the IND-CCA assumption, it turns out that the IND-

CCA security of the KEM scheme is not enough to show the full security of PQXDH. As mentioned in our modification for revision 2, we either need that the PQSPK_B KEM public key is tied inside the AD of the AEAD, or that the KEM secret is in fact bound in some way to the used public key.

The difficulty is that this binding between the secret and public key is not covered by classical cryptographic definitions. It is of course typically not covered by IND-CCA, but it is also not covered by the main previous definitions, as surveyed e.g. in [39, Fig. 2]. In all those definitions, the attacker is trying to come up with an attack against the KEM for honestly generated keys that are never compromised.

Crucially, in our setting and for re-encapsulation attacks, as outlined in [22], the attacker may know the key. We describe in the following a tailored KEM security assumption that enabled us to prove the security of PQXDH under this assumption, and also provide a proof that Kyber does in fact meet this assumption. Thus, we demonstrate that the re-encapsulation attack is not practical for Kyber, and we prove the security of the existing implementation which does not include the KEM in the AD, as required by Theorem 2.

5.3.1 Tailored KEM definition

The core issue of the re-encapsulation attack is that we don't have authentication over the PQSPK_B used by Alex. In the PQXDH setting, we can define a semi-honest collision resistance property over the KEM that enables us to prove this authentication.

Definition 1 (SH-CR)**.** *We define the advantage of an adversary $\mathcal{A}$ against a KEM (*$\mathtt{keygen}, \mathtt{encaps}, \mathtt{decaps}$*) semi-honest collision resistance as:*

$$\mathtt{Adv}^{SH\text{-}CR}_{\mathcal{A},KEM} = \Pr\left(\begin{array}{c|l} \mathtt{decaps}(ct',\mathsf{sk}) = ss \wedge & \mathsf{sk},\mathsf{pk} \xleftarrow{\$} \mathtt{keygen}() \\ (ct \neq ct' \vee \mathsf{pk} \neq \mathsf{pk}') & \mathsf{pk}' \xleftarrow{\$} \mathcal{A}(\mathsf{sk}) \\ & ss, ct \xleftarrow{\$} \mathtt{encaps}(\mathsf{pk}') \\ & ct' \xleftarrow{\$} \mathcal{A}(\mathsf{sk}, ss, ct) \end{array}\right)$$

Here,we have one agent that will decapsulate for its own honestly generated but compromised sk an attacker chosen ciphertext, another agent that will encapsulate against an attacker chosen key, and the attacker wins if both computes the same secret, while either not using the same public key or the same ciphertext. Intuitively, this corresponds to the minimal security property needed to prove KEM ciphertext and public key agreements in PQXDH.

Notice that this is logically independent of the CCR definition from [6], where the attacker tries to come up with two arbitrary ciphertexts that decapsulate to the same value for a given honestly generated but compromised secret key. The separations between the two can be proven as follows.

Theorem 4. *We have the following separations:*

1. *CCR $\not\Rightarrow$ SH-CR*
2. *SH-CR $\not\Rightarrow$ CCR*

Proof. The proof are direct constructions, though probably not practical.

For 1), consider a CCR secure KEM scheme $(\mathtt{keygen}, \mathtt{encaps}, \mathtt{decaps})$. We generate from it KEM', where $\mathtt{keygen}'$ appends 0 to all public keys, and $\mathtt{encaps}'$ simply ignores the last bit of the public key and runs $\mathtt{encaps}$. This is still a CCR secure scheme, but it is clearly not SH-CR as the attacker can simply set pk' to be pk but swapping the last bit from 0 to 1, and then win the game.

For 2) consider a SH-CR secure KEM scheme, and such that, without loss of generality, $\mathtt{encaps}$ returns with at most negligible probability for any input a ciphertext equal to just 0 and 1. We set $\mathtt{decaps}'$ to be $\mathtt{decaps}$, expect on input 0 and 1, where it arbitrarily returns 0. That is, for any key, $\mathtt{decaps}'(0,\mathsf{sk}) = \mathtt{decaps}'(1,\mathsf{sk})$. This means that KEM' is clearly not CCR secure. However, it is still SH-CR, because 0 and 1 may never be returned by the honest encapsulation yielding ct in the SH-CR definition. □

We cannot directly find our notion in [22] either, as they consider for all the definitions either two honestly generated keys (e.g. in HON-BIND-K-CT), two leaked keys (e.g. in LEAK-BIND-K-CT), or two maliciously generated keys (e.g. in MAL-BIND-K-CT), but not our semi-honest setting.

However, similarly to the fact that MAL-Bind-K-CT $\Rightarrow$ *CCR* as discussed in [6], we trivially have on our side that MAL-Bind-K,CT-PK $\Rightarrow$ SH-CR.

More generally, we do not claim that SH-CR is the ideal security definition that KEMs should target, but it is the ideal one for our setting and for use in CryptoVerif, enabling us to carry out all our proofs.

5.3.2 KEM public key agreement

Using CryptoVerif, we proved the following two theorems based on SH-CR.

Theorem 5 (Kyber is SH-CR)**.** *If the hash functions used in the Kyber design are modeled as Random Oracles, Kyber is SH-CR.*

For the reasons mentioned previously in the practical implications of the re-encapsulation attack, it is clear that McEliece does not meet SH-CR, and the situation is unclear for BIKE and HQC.

However, as Signal specifically uses Kyber, we can obtain guarantees over the current implementation.

Theorem 6 (PQXDH KEM public key agreement)**.** *Under similar hypothesis as Theorem 2,* PQXDH *also guarantees the agreement over the PQSPK used provided the KEM is SH-CR.*

5.4 Toward PQXDH version 3

Our analysis also suggest further improvements towards a future revision 3 of the protocol, which would also require changes to the implementation. While these changes do not thwart any known attack, we believe they would increase the trust in the overall design, as well as simplify the proofs. The envisioned changes are:

- to add a tag under the signature of the PQSPK, so that Alex can know whether it is using a one time or a last resort PQSPK. While this does not prevent a malicious server to only provide last resort PQSPK to the attacker, it would at least ensure that Alex knows the corresponding security level;
- to add all the public keys used inside the KDF input, in order to strengthen the data agreement and avoid relying on SH-CR.

5.5 Limitations of our Proofs

We note that our proofs only hold for our models, which do not capture all the details of how the protocol is deployed.

The first limitation is that Signal uses the long term identity keys IK for both curve operations using X25519 as-well-as signature operations XEdDSA. This is a joint key reuse between algorithms, where one would in fact need a joint security hypothesis on both primitives, in the vein of [37]. This issue was completely ignored in the pen-and-paper Signal analysis [21], where the signatures were simply dropped and the data assumed to be pre-authenticated. We take a more precise approach, by still including the signatures, but assuming that IK contains two key pairs, one for curve computations and one for signatures. Coming up with a precise model to capture this key reuse is still an open question, even without considering a formal CryptoVerif proof.

A second limitation is that the CryptoVerif model only considers the compromise of the long term keys, and not of the ephemeral keys. The difficulty here is that each new compromise implies to completely redo the proof, and notably also redefine variants of the cryptographic axioms that allow for those compromises.

A third limitation is that we do not model or prove anything about deniability, which we leave as an open question.

A fourth and final limitation, of both the CryptoVerif and ProVerif models, is that we only consider PQXDH, and not the full eco-system. Lifting this limitation however represents a major challenge for our tools.

6 Lessons learned

Our formal analysis in collaboration with Signal's protocol designers and developers highlighted several interesting points, both with respect to the design of future hybrid schemes, as-well-as how to carry out such formal verifications.

6.1 Designing Hybrid schemes

The core issues we identify that one should consider when updating a protocol to a post-quantum variant are:

- The added KEM public key and ciphertext should be mutually authenticated, otherwise IND-CCA is probably not enough to prove the protocol secure, and the protocol may in fact be insecure for some specific KEM instantiations.
- Domain separation within the protocol between DH public keys and KEM public keys is crucial, as confusion may lead to a downgrade of security.
- Domain separation between distinct protocols is important. This is a typical case where both the old classical version of the protocol and it's post-quantum version will exist in parallel, so derived keys and other materials should not be confused between the two.

6.2 Formal verification

Our approach highlighted the interest of using both ProVerif and CryptoVerif in combination to analyze protocols:

- The lack of domain separation makes it impossible to get a CryptoVerif proof, so we can quickly suspect the presence of a confusion attack. These attacks are not caught by default in ProVerif, since the standard definitions of cryptographic primitives in ProVerif embed domain separation. But once we suspect an attack, we can model it explicitly and then use ProVerif to confirm a concrete attack on the protocol.
- We can model more compromises in ProVerif, and thus catch some attacks not covered by a CryptoVerif proof. One such attack is found, we can fix the protocol, and then go back to the CryptoVerif model and formally prove that the attack is indeed removed.
- Our exhaustive methodology in ProVerif allows us to capture as many attacks as the tool can find. One should always first try to include all possible compromises in the model, and then refine the security queries until they can be proved.

7 Related Work

This work is the first to analyze PQXDH and led to an update of the specification within months of its release. As such, there is no previous related work precisely on PQXDH. In the following, we discuss closely related work.

Our work follows in the line of previous formal analysis of protocol standards using the same or similar verification tools, e.g. of TLS 1.3 [11], 5G-AKA [8], EMV [9, 34] or EDHOC [26], where those analyzes often led to updates of the protocol specifications and implementations.

A variant of X3DH was previously studied in [27]. Our CryptoVerif models were inspired by them but heavily

adapted, both to match the X3DH version as-well-as the PQXDH model. Our PROVERIF models were, on the other hand, rewritten from scratch.

The Signal application eco-system is wider than X3DH, as the derived key is used in combination with the Double-Ratchet (DR) protocol, and an additional layer is introduced by the Sesame multiple session management protocol. X3DH composed with the subsequent DR protocol was analyzed with a pen-and-paper computational proof in [21], but they made rather strong assumptions over the way public keys are authenticated, whereas we model the signatures over those keys, which are crucial to some of the weaknesses we found. More pen-and-paper proofs of the composed X3DH +DR protocol can be found in [3, 13]. In our work, we prove strong formal guarantees for the new PQXDH protocol in isolation.

Implementations of X3DH and DR were also analyzed using DY* [10], and their integration within Sesame was analyzed using the Tamarin prover [24]. Both these works are only inside the symbolic model and use less precise models and cryptographic assumptions for X3DH than our work.

Importantly, none of the results of the previous analyzes of X3DH carry over to PQXDH, as extending a secure protocol with new elements can break its security, even if all these elements are individually secure. Indeed, our work uncovers a potential attack on X3DH because of new key confusions introduced by PQXDH version 1.

A few other tools support post-quantum computational guarantees, such as Squirrel [4, 23] and EasyCrypt [5, 7, 38], but they have not yet been applied to protocols like PQXDH. For us, it was natural to rely on CRYPTOVERIF by building on the previously developed models of X3DH.

We also uncover an attack based on subtle definitions of KEMs, similar to the re-encapsulations attacks discussed in [22]. We discuss the actual KEM security assumption needed for our setting, and link it with recent works [6, 22].

8 Conclusion

We performed an extensive analysis of the recent PQXDH proposal, finding design flaws, contributing concrete updates to the protocol specification, and developing machine-checked security proofs for the fixed version.

This work was performed in collaboration with the Signal design team and demonstrates how formal methods can be efficiently used in a short time frame to improve the design of post-quantum hybrid protocols.

It also highlights the interesting synergy of using multiple tools, as the analysis results from both CRYPTOVERIF and PROVERIF mutually improved each other.

Much work remains to be done around the formal analysis of Signal and of post-quantum secure messaging in general. Important future goals include precisely modeling the full Signal eco-system, developing computational analyzes that can capture the long term identity key reuse for signatures and curve computations, and designing and analyzing a post-quantum secure Double Ratchet protocol.

Acknowledgments

This work was partly done while Charlie Jacomme was at Inria, Paris, France. This work benefited from funding managed by the French National Research Agency under the France 2030 programme with the references ANR-22-PECY-0006 (PEPR Cybersecurity SVP) and ANR- 22-PETQ-0008 (PEPR Quantic PQ-TLS).

References

[1] Classic mceliece. conservative code-based cryptography: design rationale. online, 2022. https://classic.mceliece.org/mceliece-rationale-20221023.pdf.

[2] Module-lattice-based key-encapsulation mechanism standard. Technical report, Gaithersburg, MD, 2023.

[3] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: security notions, proofs, and modularization for the signal protocol. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 129–158. Springer, 2019.

[4] David Baelde, Stéphanie Delaune, Adrien Koutsos, Charlie Jacomme, and Solène Moreau. An interactive prover for protocol verification in the computational model. In *42nd IEEE Symposium on Security and Privacy (S&P'21),*, pages 537–554, Los Alamitos, CA, May 2021. IEEE Computer Society Press.

[5] Manuel Barbosa, Gilles Barthe, Xiong Fan, Benjamin Grégoire, Shih-Han Hung, Jonathan Katz, Pierre-Yves Strub, Xiaodi Wu, and Li Zhou. Easypqc: Verifying post-quantum cryptography. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2564–2586, 2021.

[6] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, and Bas Westerbaan. X-wing: The hybrid kem you've been looking for. *Cryptology ePrint Archive*, February 2024. https://eprint.iacr.org/archive/2024/039/1707692113.pdf.

[7] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 71–90, Berlin, Heidelberg, August 2011. Springer.

[8] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1383–1396, 2018.

[9] David Basin, Ralf Sasse, and Jorge Toro-Pozo. The emv standard: Break, fix, verify. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1766–1781. IEEE, 2021.

[10] Karthikeyan Bhargavan, Abhishek Bichhawat, Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, and Tim Würtele. Dy*: A modular symbolic verification framework for executable cryptographic protocol code. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 523–542. IEEE, 2021.

[11] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *IEEE Symposium on Security and Privacy (S&P'17)*, pages 483–503, Los Alamitos, CA, May 2017. IEEE Computer Society Press.

[12] Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. Cryptoverif and proverif models, 2024. https://github.com/Inria-Prosecco/pqxdh-analysis/tree/2e676a009471f370dbbfad3ac7ab5d7d9518ab57.

[13] Alexander Bienstock, Jaiden Fairoze, Sanjam Garg, Pratyay Mukherjee, and Srinivasan Raghuraman. A more complete analysis of the signal double ratchet algorithm. In *Annual International Cryptology Conference*, pages 784–813. Springer, 2022.

[14] Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends® in Privacy and Security*, 1(1-2):1–135, 2016.

[15] Bruno Blanchet and Charlie Jacomme. Post-quantum sound CryptoVerif and verification of hybrid TLS and SSH key-exchanges. In *Proceedings of the 37th IEEE Computer Security Foundations Symposium (CSF'24)*. IEEE Computer Society Press, 2024.

[16] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[17] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *EuroS&P*, pages 353–367. IEEE, 2018.

[18] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of ed25519: theory and practice. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1659–1676. IEEE, 2021.

[19] 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, *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2022.

[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, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 404–430. Springer, 2020.

[21] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, 33:1914–1983, 2020.

[22] Cas Cremers, Alexander Dax, and Niklas Medinger. Keeping up with the kems: Stronger security notions for kems. *Cryptology ePrint Archive*, 2023.

[23] Cas Cremers, Caroline Fontaine, and Charlie Jacomme. A logic and an interactive prover for the computational post-quantum security of protocols. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 125–141. IEEE, 2022.

[24] Cas Cremers, Charlie Jacomme, and Aurora Naska. Formal analysis of session-handling in secure messaging: Lifting security from sessions to conversations. In *Usenix Security*, 2023.

[25] 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. *J. Cryptol.*, 35(3):17, 2022.

[26] Charlie Jacomme, Elise Klein, Steve Kremer, and Maïwenn Racouchot. A comprehensive, formal and automated analysis of the edhoc protocol. In *USENIX Security'23-32nd USENIX Security Symposium*, 2023.

[27] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2017 IEEE European symposium on security and privacy (EuroS&P)*, pages 435–450. IEEE, 2017.

[28] Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol - diff between rev 1 and 2, September 2023. https://github.com/Inria-Prosecco/pqxdh-analysis/blob/main/revision2/pqxdh-diff-rev-1-to-2.pdf.

[29] Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol - revision 1, September 2023. Archive: https://github.com/Inria-Prosecco/pqxdh-analysis/blob/main/revision1/pqxdh-rev1.pdf.

[30] Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol - revision 2, September 2023. Archive: https://github.com/Inria-Prosecco/pqxdh-analysis/blob/main/revision2/pqxdh-rev2.pdf.

[31] A. Langley, M. Hamburg, and S. Turner. Elliptic curves for security. RFC 7748, RFC Editor, January 2016.

[32] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol, 2016.

[33] Trevor Perrin and Moxie Marlinspike. The Double Ratchet Algorithm, 2016.

[34] Andreea-Ina Radu, Tom Chothia, Christopher JP Newton, Ioana Boureanu, and Liqun Chen. Practical emv relay protection. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1737–1756. IEEE, 2022.

[35] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 98–107, 2002.

[36] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[37] Erik Thormarker. On using the same key pair for ed25519 and an x25519 based kem. *Cryptology ePrint Archive*, 2021.

[38] Dominique Unruh. Quantum relational hoare logic. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–31, 2019.

[39] Keita Xagawa. Anonymity of nist pqc round 3 kems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 551–581, 2022.

A Formal Symbolic security Theorems

We proved in PROVERIF a set of three queries, modeling:

1. the exhaustive case disjunction under which the key computed by the responder Alex is secret;
2. the exhaustive case disjunction under which the key computed by the initiator Blake is secret;
3. the exhaustive case disjunction under which there is authentication.

In PROVERIF, the final correct and complete query we were able to prove for the secrecy of the responder key is:

```
query A,B:client, spk:point, pqpk:kempub, sk:symkey,
     i,j:time;
  event(AlexOK(A,B,spk,pqpk,ts))@i
    ⟹ not(attacker(sk))
       | (CompromisedIK(B)@j
            & (j<i
               | (event(CompromiseSPK(B,spk))
                  & (event(CompromisePQPK(B,pqpk))
                     | event(BrokenKEM))
                  )
               )
            )
         | (event(BrokenDH())@j
              & (j < i
                 | event(CompromisePQPK(B,pqpk))
     | event(BrokenKEM)
                 )
              )
```

This can be translated as a security theorem detailing the possible cases.

Theorem 7 (Symbolic Responder Secrecy)**.** *The key SK_A computed by the responder Alex is secret, unless:*

1. *IK_B was compromised before the completion of the key exchange (this is to be expected, this is essentially a malicious A case).*
2. *IK_B was compromised after the key exchange, as well as some SPK_b, and either KEMs are broken or the corresponding $PQSPK_B$ has been compromised (the attacker just need to send a malicious OPK here to be able to compute all the values).*
3. *DH was broken before the key exchange (this is similar to case 1).*
4. *DH was broken only after the key exchange, and either KEMs are fully broken or the $PQSPK_B$ (similar to case 2) .*

All those cases appear normal, and it is in all the possible compromise cases that still maintain security that we find the classical security notions to be implied.

Remark that using an OPK does not change anything on the responder side, as they are not authenticated. But of course, compromising the OPK makes it so that the honest responder will never receive and answer the message.

We can also prove some form of authentication and data agreement:

Theorem 8 (Symbolic authentication)**.** *Whenever an initiator Blake accepts (resp. with or without an OPK), then, there exists a responder Alex that also accepted (resp. with or without the same OPK) with the same SPK_B and $PQSPK_B$, unless:*

1. *IK_A was compromised before the exchange (allows the impersonation of Alex).*
2. *Some SPK_B was compromised before the exchange (knowing SPK_B allows to impersonate A).*
3. *DH has been broken before the exchange.*

The second compromise case here is the most interesting one. Indeed, it means that Signal is suceptible to a form of ephemeral key compromise impersonation: compromising the ephemeral SPK_B secret of Blake allows to impersonate anybody to Blake. This is natural with the design of Signal, as Alex only authenticates by using a long term DH share combined with the SPK_B.

Theorem 9 (Symbolic Initiator Secrecy)**.** *The key SK_B computed by the initiator Blake is secret, unless:*

1. *IK_A was compromised before the communication. (this allows a full impersonation of Alex)*
2. *Some SPK_B was compromised before the communication. (more surprisingly, but inevitably, this allows a full impersonation of any Alex)*
3. *IK_B, SPK_B and $PQSPK_B$ are compromised after the communication, and either no OPK was used or it was compromised. (all secret material of A is leaked here, so this is a very natural case)*
4. *IK_B and SPK_B are compromised after the communication, and KEMs are broken, and either no OPK was used or it was compromised. (similar to 3)*
5. *DH was broken before the exchange (naturally breaks everything)*
6. *DH was broken after the exchange, and the $PQSPK_B$ used by the responder was compromised (similar to 3)*
7. *DH was broken after the exchange, and KEMs are broken (similar to 5)*

In contrast, the OPK now plays a role and increases the security, and also need to be compromised in many cases.

Summing up the three theorems, we do have the higher level security properties mentioned in Theorem 1:

1. peer-authentication: directly implied by Theorem 8, where authentication holds unless there is a compromise.
2. forward secrecy: implied by Theorems 7 and 9, where we can see that compromising the identity key of any party after the completion of a key exchange is not one of the insecure case, and is thus not enough to break the security.
3. resistance to key compromise impersonation: implied by Theorem 8, as authentication holds even if IK_B was compromised.
4. session independence: implied by Theorems 7 and 9, where we can see that one must compromise precisely ephemerals secrets of the target session to break it, and thus that leaking the ephemeral keys of other sessions does not impact security.
5. resistance to "harvest now decrypt later": all three theorems give us that just breaking DH after completion is not enough, and that additional compromises are always needed.

B Full protocol description

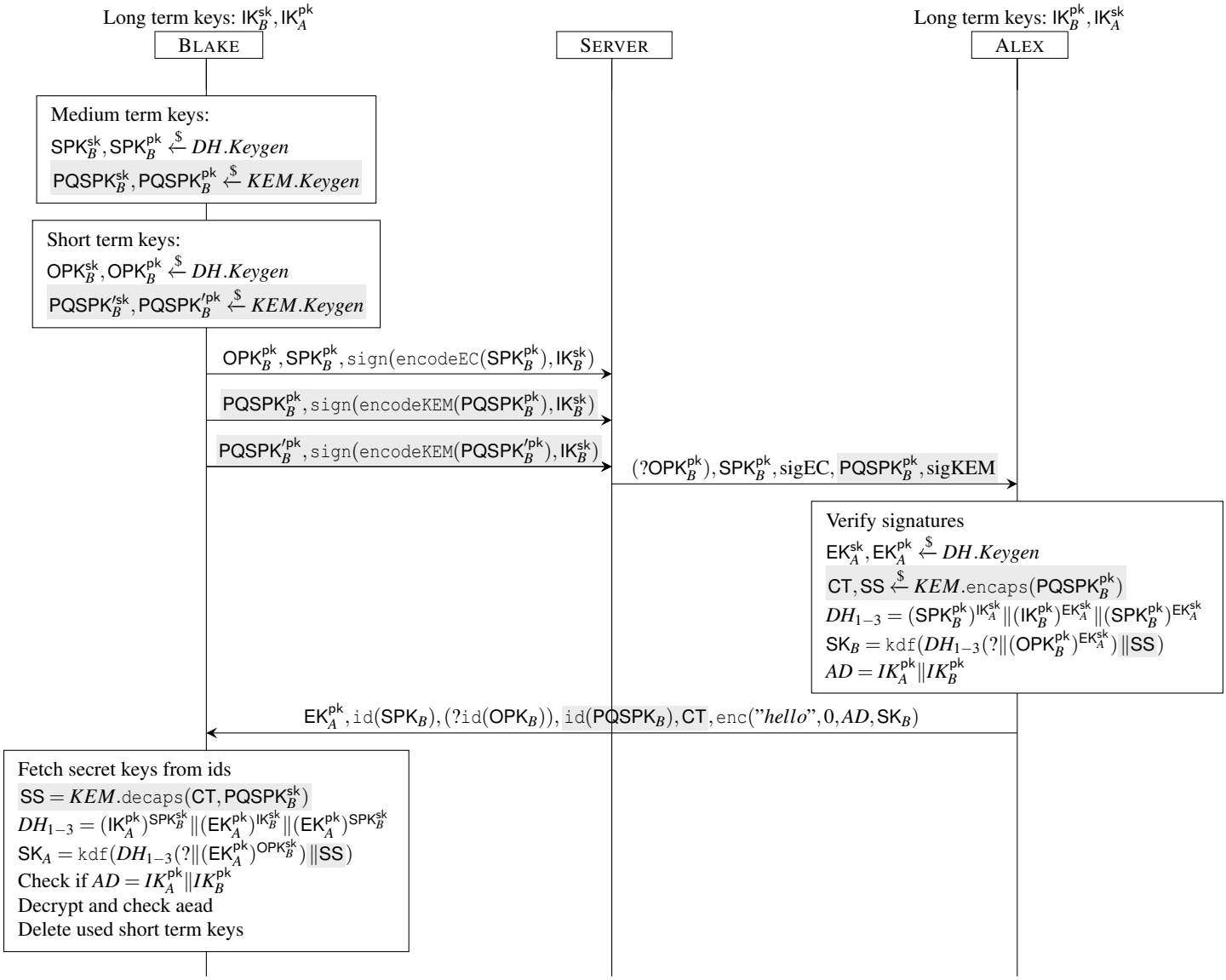

Figure 4: The PQXDH protocol, revision 1.

It is exactly the X3DH protocol, with additional KEM computations, highlighted with a light gray background. Question marks denote optional elements.