Hybrid Signal protocol for post-quantum email encryption

Sara Stadler[1], Vitor Sakaguti[1], Harjot Kaur[2], and Anna Lena Fehlhaber[3]

[1]Tutao GmbH, pqmail@tutao.de
[2]Leibniz University Hannover, kaur@sec.uni-hannover.de
[3]annalenafehlhaber@tutanota.com

June 24, 2021

Abstract

The Signal protocol is used in many messaging applications today. While it is an active research topic to design a post-quantum variant of the protocol, no such variant is currently realized in the real world. In the following document we describe a hybrid version of the Signal protocol, that will be implemented to achieve post-quantum security for Tutanota's end-to-end encrypted e-mails.

1 Introduction

The Signal protocol, developed by Open Whisper Systems, is a cryptographic protocol for end-to-end encrypted communication. It is used in many messaging applications today. The protocol works asynchronously, which makes it also well suited for e-mails where we cannot expect both communication participants to be online at the same time. In this document we describe a hybrid version of the Signal protocol. The protocol design is the result of a joint research project between the secure email service Tutanota and the the Leibniz University Hanover. It will be implemented to achieve post-quantum security for Tutanota and to add forward and future secrecy to our end-to-end encrypted e-mails. A major design requirement therefore was to rely only on post-quantum key encapsulation mechanisms (KEM) and digital signature schemes which are currently undergoing standardization by NIST. Currently we have already implemented a prototype proving it's real world applicability.
In Section 2 we briefly describe the original Signal protocol. Afterwards we introduce our modifications in Section 3. In Section 4 we summarize how our protocol differs from the original Signal protocol.

2 Preliminaries: The Signal Protocol

While the Signal is in fact an amalgamation of the Extended Triple Diffie-Hellman (X3DH) key exchange protocol [14] and the Double Ratchet algorithm [13] we will treat it as one protocol in the following and only distinguish sub-protocols if needed.

2.1 Key features of the protocol

The protocol offers the following key features (see e.g. [16]):

1. **Asynchrony**: In the X3DH key exchange protocol prekeys are used so that there is no need for both parties to be online at the same time.

2. **Confidentiality**: All messages are encrypted and therefore not made available or disclosed to unauthorized parties.

3. **Integrity**: The consistency, accuracy, and trustworthiness of the data in transit is ensured by the use of authenticated encryption with associated data (AEAD), where messages are authenticated with MAC tags.

4. **Authentication**: The communicating parties authenticate each other by the means of long-term identity keys. Trust in these keys is established via fingerprint comparison (manually or using qr-codes). Any other mechanism for trust-establishment can, however, be plugged-in easily.

5. **Perfect Forward Secrecy**: As each message key is freshly derived from the previous one via a key derivation function (KDF), a key compromise does not enable the adversary to decrypt past messages.

6. **Future Secrecy**: Frequently contributing fresh shared secrets to the KDF facilitates the recovery of the key chain and ensures that an adversary in possession of one encryption key cannot keep listening to the conversation for long.

7. **Message Unlinkability** Different messages exchanged in one conversation cannot be linked as they are encrypted and authenticated with one-time keys. Therefore, if one message could for some reason be linked to a specific party, there would still be no proof that same party authored any other message.

8. **Offline Deniability**: The protocol does not provide any means to prove that a message was authored by a specific party after it has been sent. However, if one of the communicating parties collaborates with a third party during a protocol run, such proof can be provided.

2.2 Building blocks

The protocol contains the following building blocks:

1. **Publishing keys**: The X3DH protocol uses prekeys to achieve asynchrony. Therefore each party X must generate the following key pairs:

   - n one-time prekey pairs $\{(ospk_X^i, oppk_X^i)\}i$, where i is a unique id.
   - A semi ephemeral key pair $\{(sspk_X^j, sppk_X^j)\}j$, where j is a unique id
   - A long-term identity key pair (isk_X, ipk_X)

   From these keys, so called key bundles are generated and uploaded on a key server. A key bundle contains:

   - The user's registration id (id_X)
   - A public one-time key preceded by a unique id $(i, oppk_X^i)$
   - The semi-ephemeral public key signed with the long-term identity key and also identified by a unique id $(j, sppk_X^j, sig_X^j)$, where $sig_X^j \leftarrow sign(isk_X, sppk_X^j)$
   - the public long-term identity key (ipk_X)

   When Alice wants to send a message to Bob, with whom she did not communicate before, she first needs to fetch his key bundle from a server.

2. **Authenticated Key Exchange (AKE)**: The AKE is carried out by the X3DH protocol. Once Alice received Bob's key bundle from the server, she verifies the signature on the signed prekey:

   $$verify(ipk_B, sppk_B^j, sig_B^j) \tag{2.1}$$

   She then generates a base key pair

   $$(bsk_A, bpk_A) \leftarrow gen() \tag{2.2}$$

   and performs four Diffie-Hellman (DH) calculations using different combinations of her base key's and identity key's secret key and Bob's public keys from the server. This yields the following keys:

   $$\begin{aligned} k_1 &\leftarrow DH(isk_A, sppk_B^j), \\ k_2 &\leftarrow DH(bsk_A, ipk_B), \\ k_3 &\leftarrow DH(bsk_A, sppk_B^j), \\ k_4 &\leftarrow DH(bsk_A, oppk_B^i). \end{aligned} \tag{2.3}$$

   Note that the calculation of k_4 is optional, as all one-time prekeys sent to the server may have been fetched already. All results are concatenated to derive a master secret

$$sk \leftarrow KDF(k_1||k_2||k_3(||k_4)). \tag{2.4}$$

This master secret can be used to start a session with Bob. However, in the combined signal protocol a ratchet step (as described below) is performed beforehand to derive the actual message key (mk).

Note that Bob can calculate the same master secret as Alice using Alice's base key and identity public key (which he will receive with Alice's initial message) and the secret keys corresponding to the public keys from his key bundle:

$$\begin{aligned} k_1 &\leftarrow DH(sspk_B^j, ispk_A), \\ k_2 &\leftarrow DH(isk_B, bpk_A), \\ k_3 &\leftarrow DH(sspk_B^j, bpk_A), \\ k_4 &\leftarrow DH(ospk_B^i, bpk_A). \end{aligned} \tag{2.5}$$

3. **Ratchet steps** : Before Alice can start sending messages to Bob she derives the message key mk using the Double Ratchet algorithm.
This algorithm allows Alice and Bob to continuously agree on new encryption keys to achieve forward and future secrecy. For this purpose three KDF chains are implemented: a root chain, a sending chain and a receiving chain. These chains are advanced in two different ratchets.

   For the asymmetric (or Diffie-Hellman)-Ratchet (see Figure 1) Alice and Bob continuously generate new Diffie-Hellman key pairs and send the respective public keys to the receiving party. This way both parties can calculate a new DH output (k_n). For the initial ratcheting Alice uses the private key of a newly generated ephemeral keypair and Bob's signed prekey:

$$\begin{aligned} (esk_A, epk_A) &\leftarrow gen() \\ k_n &\leftarrow DH(esk_A, sppk_B). \end{aligned} \tag{2.6}$$

   After having received Alice's message containing epk_A, Bob calculates the same secret:

$$k_n \leftarrow DH(sspk_B, epk_A). \tag{2.7}$$

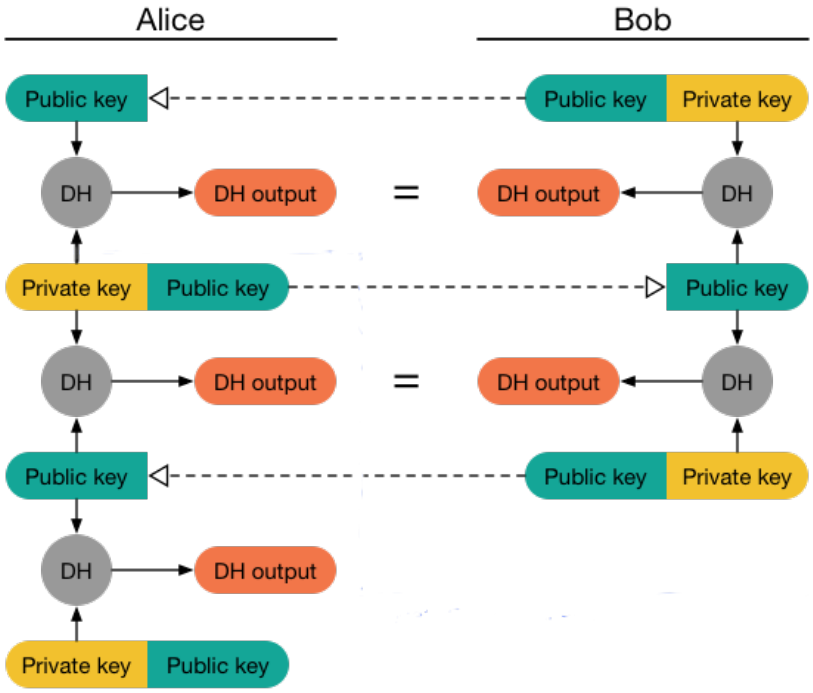

Figure 1: Continuous Diffie-Hellman key exchange in the Double Ratchet Algorithm [13].

The DH output becomes the input to a KDF on the root chain (see Figure 2) together with the root key.

$$(rk, ck) \leftarrow KDF(rk, k_n). \tag{2.8}$$

The master secret derived by the $X3DH$ protocol serves as initial root key. The output of this KDF is a new root key and a chain key. This step is performed by Alice and Bob analogously with the only difference that Alice uses the new chain key to start a sending chain while Bob uses it to start a receiving chain.

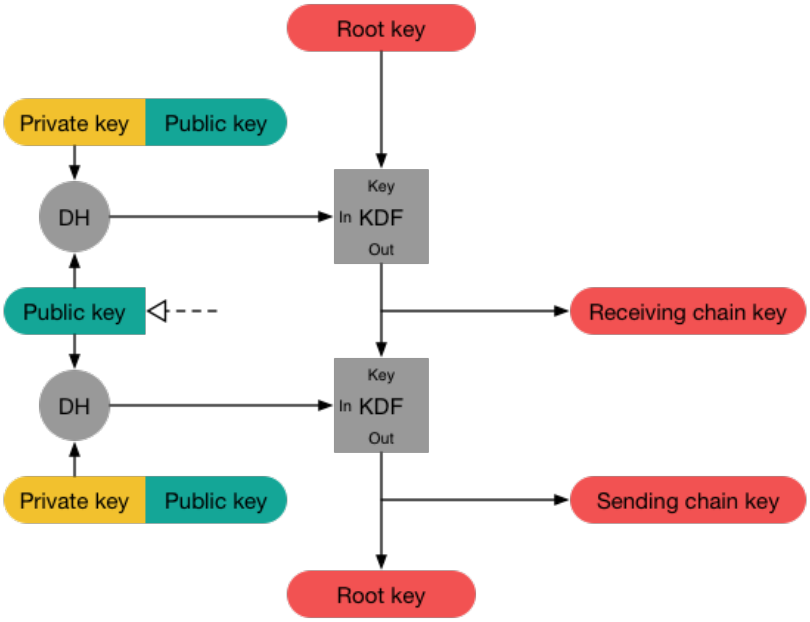

Figure 2: Key derivation in the asymmetric Ratchet of the Double Ratchet Algorithm [13].

Alice's sending and Bob's receiving chain are advanced analogously in the symmetric ratchet (see Figure 3) where a key derivation is performed using the newly generated chain key as input

$$(ck, mk) \leftarrow KDF(ck). \tag{2.9}$$

The output is a new chain key and the message key. Note that those keys will be identical for Alice and Bob.

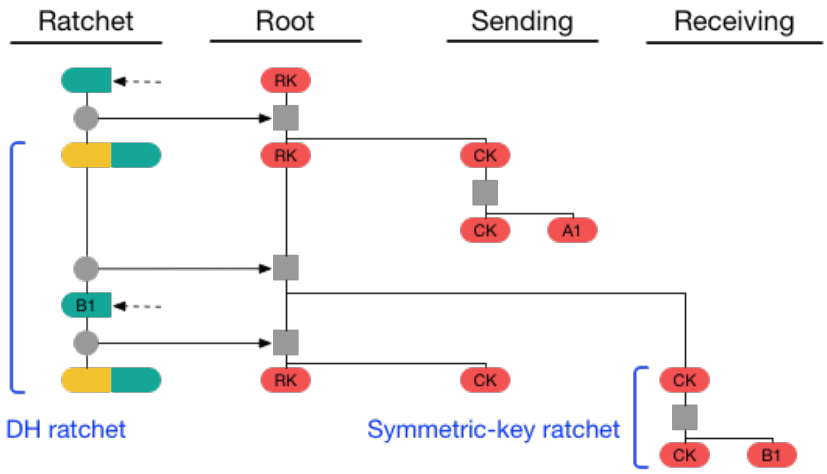

Figure 3: Key derivation in the symmetric ratchet of the Double Ratchet Algorithm [13].

The Ratchet steps are repeated in the same way whenever Alice or Bob receive a new public key from the other party. Whenever they want to send a message without having received before they only advance the sending chain in the symmetric ratchet without advancing the root chain before.

4. **Send steps**: Once Alice generated the message key, she can finally compose the initial message and send it to Bob. The initial message contains:
   - only for the initial (X3DH) message:
     - Alice's registration id (id_A)
     - Alice's identity public key (ipk_A)
     - Alice's public base key (bpk_A)
     - Identifiers stating which of Bob's one-time keys and signed prekeys she used (i, j)
   - The header (h), containing:
     - The new asymmetric ratchet public key (epk_A)
     - The message number (n)
     - The previous chain length (pn)
   - the ciphertext (c)
   - and the respective tag(t)

The message m is encrypted using a symmetric AEAD algorithm which in addition to the ciphertext outputs a tag that guarantees the integrity of the message and associated data. The associated data (ad) consists of the message header h and the public identity keys of both parties:

$$\begin{aligned} h &\leftarrow header(epk_A, n, pn) \\ ad &\leftarrow h||encode(ipk_A)||encode(ipk_B) \\ (t, c) &\leftarrow enc(mk, m, ad); \end{aligned} \tag{2.10}$$

5. **Receive steps**: Bob, who generated the same message key mk, can decrypt the message and verify the MAC tag.

$$ad \leftarrow h||encode(ipk_A)||encode(ipk_B) \tag{2.11}$$

$$m \leftarrow dec(mk, c, t, ad) \tag{2.12}$$

If the message decrypts and verifies successfully the communication between Alice and Bob has been established, otherwise the protocol is aborted.

3 A hybrid Signal protocol

In this section we describe a hybrid version of the Signal protocol, combining the original protocol with a post quantum variant. This post-quantum variant differs from the original protocol in some aspects that will be described in more detail in this section. These differences are due to the fact that the Diffie-Hellman (DH) key exchange has to be replaced by a key encapsulation mechanism (KEM) as there is currently no suitable post-quantum replacement for DH. We discussed using SIDH and CSIDH. However, the former is not secure to be used with non-ephemeral keys [5, 9] while the latter has still little maturity as well as some performance issues.

A KEM variant of the Double Ratchet protocol was first described by Duits [8]. Brendel et al. [5] discuss the construction of X3DH using a KEM. However, they point out that standard KEMs are inadequate to achieve asynchrony for an authenticated key exchange (AKE) as the encapsulating party cannot contribute non-ephemeral input (and thus cannot authenticate by contributing her identity key). They discuss so called split KEMs where both parties contribute keys to the KEM as a possible solution but are not able to provide a strongly-secure instance of such a KEM. They point out that while many passively-secure lattice-based KEMs are split KEMs, they become insecure when keys are used more than once (which is essential to X3DH). To make key reuse secure they have to be transformed via the the Fujisaki-Okamoto (or a similar) transform with the consequence of losing their compliance with the split KEM design.

In our proposal for a post-quantum variant we therefore modified the X3DH protocol to use a signature to replace the Diffie-Hellman calculation involving the initial senders identity key. In Section 4 we discuss how this impacts the protocol's security features. The most obvious impact is the loss of deniability

for the X3DH message. In their most recent paper [6] Brendel et al. show how to design a similar post-quantum secure variant of the X3DH protocol using designated verifier signature schemes instead. Their design looks promising but seems to require further analysis before it can be used in real-world applications.

3.1 Building blocks

In the following we describe the building blocks of the hybrid protocol. As the pre-quantum variant does not differ from the one already described in Section 2 we only describe the post-quantum secure variant and the way both variants are combined in the hybrid approach. Note that in the following all keys and signatures used or generated in the pre-quantum protocol are marked as ^{pre} while the same values for the post-quantum variant are marked as ^{post}, an asterisk (*) indicates that both values are included separately and two asterisks (**) mark their concatenation.

1. **Publishing keys**: For the hybrid protocol the users have to generate the following post-quantum pre-keys in addition to the keys used in the original protocol:

   - n one-time prekey pairs $\{(^{post}ospk_X^i,\ ^{post}oppk_X^i)\}i$, where i is a unique id.
   - A semi ephemeral key pair $\{(^{post}sspk_X^j,\ ^{post}sppk_X^j)\}j$, where j is a unique id
   - A long-term identity key pair $(^{post}isk_X,\ ^{post}ipk_X)$

   They then publish key bundles of the following form on the server.

   - The user's registration id (id_X), which will be the e-mail address
   - Pre- and post-quantum public one-time keys preceded by a unique id $(i,\ ^*oppk_X^i)$
   - Pre- and post-quantum semi-ephemeral public keys signed with the long-term identity keys, identified by a unique id $(j,\ ^*sppk_X^j,\ ^*sig_X^j)$, where $^*sig_X^j$ consists of $^{pre}sig_X^j \leftarrow sign(^{pre}isk_X,\ ^{pre}sppk_X^j)$ and $^{post}sig_X^j \leftarrow sign(^{post}isk_X,\ ^{post}sppk_X^j)$.
   - Pre- and post-quantum public long-term identity keys ($^*ipk_X$). Note that the post-quantum identity key consists of 2 keys: a signing key and a key encapsulation key. For simplicity's sake we, however, treat it as one key in the following.

   Note that all pre- and post quantum keys are included as separate data structures. We consider it sufficient to have a pre-quantum only signature on the pre-quantum and a post-quantum signature on the post quantum key, so that each key is securely authenticated as long as the respective signature scheme remains secure.

2. **Authenticated Key Exchange (X3DH)**: For the post-quantum variant some modifications have to be made to the X3DH AKE as there is no suitable Diffie-Hellman alternative and the respective calculations have to be replaced with key encapsulations.

   Definition 1 *A key encapsulation* $(c, k) \leftarrow encaps(pk_X)$ *takes a public key as input and outputs a symmetric key and a corresponding ciphertext. The decapsulating party obtains the symmetric key by calculating* $k \leftarrow decaps(sk_X, c)$.

   After fetching Bob's key bundle from the server Alice has to verify an additional post-quantum signature on Bob's post-quantum signed pre-key:

$$verify({}^{post}ipk_B,\ {}^{post}sppk_B^j,\ {}^{post}sig_B^j) \tag{3.13}$$

   Alice does not generate a post-quantum equivalent to the base key pair as KEMs do not provide the possibility to contribute particular secret keys but instead (differing) randomness is used inside the key encapsulation function. For the same reason she cannot contribute her non-ephemeral identity key.

   Using key encapsulations instead of DH calculations yields the following steps:

$$\begin{aligned}(c_2,\ {}^{post}k_2) &\leftarrow encaps(\ {}^{post}ipk_B),\\ (c_3,\ {}^{post}k_3) &\leftarrow encaps(\ {}^{post}sppk_B^j),\\ (c_4,\ {}^{post}k_4) &\leftarrow encaps(\ {}^{post}oppk_B^i).\end{aligned} \tag{3.14}$$

   Note that Alice cannot calculate an equivalent to k_1 and therefore, does not achieve authentication during the key exchange.

   The ${}^{pre}k_i$ and ${}^{post}k_i$ resulting from the pre- and post-quantum key agreement are concatenated to derive the master key:

$$sk = KDF({}^{pre}k_1||\ {}^{**}k_2||\ {}^{**}k_3||\ {}^{**}k_4) \tag{3.15}$$

   To achieve the authentication of the encapsulating party, the encapsulated keys are signed:

$$\begin{aligned}data \leftarrow c_2||c_3||c_4||H({}^{post}ipk_B)||H({}^{post}sppk_B^j)||(H(\ {}^{post}oppk_B^i))\\ {}^{post}sig_A \leftarrow sign({}^{post}isk_A, data),\end{aligned} \tag{3.16}$$

   where H is a cryptographically secure hash function. Note that we include the hash of the used encryption keys in the signature to prevent any attack where an adversary can find a different combination of plaintext and encryption key that yields the same ciphertext and claim that the new plaintext is the one originally signed by the victim (see [2]).

Bob can calculate the same master secret as Alice using the secret keys corresponding to the public keys from his key bundle. Note that before performing this steps Bob verifies the signature on the encapsulated secrets using Alice's identity key, which he receives with the initial message:

$$\begin{gathered} data \leftarrow c_2||c_3||c_4||H({}^{post}ipk_B)||H({}^{post}sppk_B^j)||(H({}^{post}oppk_B^i)) \\ verify({}^{post}ipk_A, data, {}^{post}\, sig_A) \end{gathered} \tag{3.17}$$

Bob then decapsulates all keys:

$$\begin{aligned} {}^{post}k_2 &\leftarrow decaps({}^{post}isk_B, c_2) \\ {}^{post}k_3 &\leftarrow decaps({}^{post}sspk_B^j, c_3) \\ {}^{post}k_4 &\leftarrow decaps({}^{post}ospk_B^i, c_4) \end{aligned} \tag{3.18}$$

3. **Ratchet steps**: For this step we use the Double Ratchet algorithm as described in Section 2. For post-quantum security we use a variant of the Double Ratchet protocol where Diffie-Hellman calculations are replaced with key encapsulations in the asymmetric Ratchet (see Figure 4).

   For the initial ratcheting Alice uses Bob's signed prekey. She generates a new key pair as well but cannot contribute her private key to the key encapsulation. Her new key pair will only be used for the next message that Bob sends to her.

$$\begin{aligned} ({}^{post}esk_A,\ {}^{post}epk_A) &\leftarrow gen() \\ (c_n,\ {}^{post}k_n) &\leftarrow encaps({}^{post}sppk_B^j) \end{aligned} \tag{3.19}$$

   The output of the key encapsulation is a tuple, containing a newly generated secret and a matching ciphertext. After having received Alice's message containing the ciphertext c_n, Bob calculates the same secret:

$${}^{post}k_n \leftarrow decaps({}^{post}sspk_B^j, c_n). \tag{3.20}$$

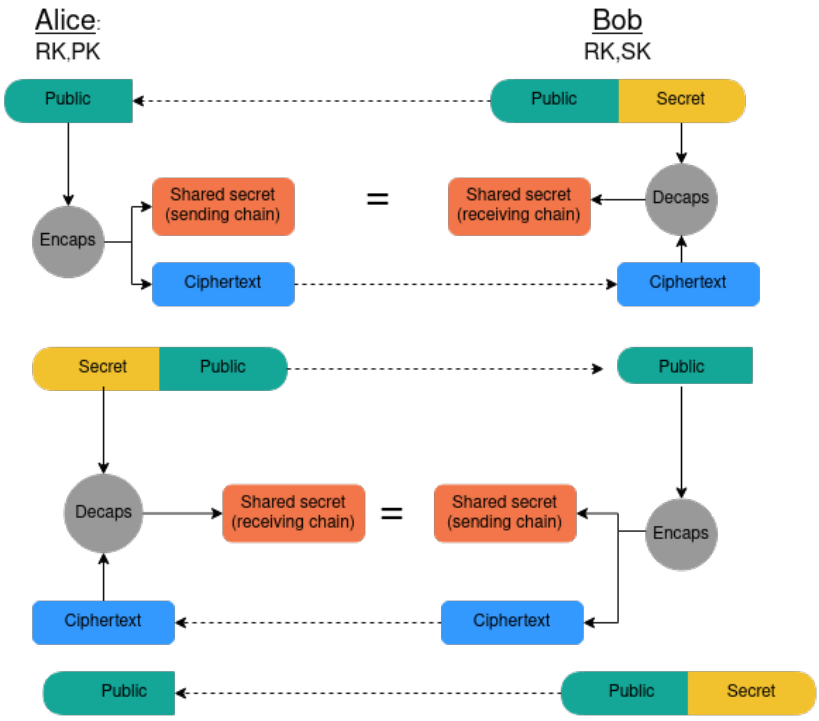

Figure 4: The key exchange in the Double Ratchet Algorithm using key encapsulation instead of DH calculations.

The root chain and the sending/receiving chain are advanced the same way as in the original Double-Ratchet-Algorithm. The only difference relates to the input to the key derivation on the root chain which consists in the concatenation of the DH shared secret and the shared secret from the key encapsulation:

$$(rk, ck) \leftarrow KDF(sk, \ ^{**}k_n)). \tag{3.21}$$

4. **Send steps**: In this step the header and associated data (ad) are generated and the message is encrypted and sent. The message now has the following form (e.g. for Alice):

   - only for the initial (X3DH) message:
     - Alice's registration id (id_A)
     - Alice's identity public key ($^{*}ipk_A$)
     - Alice's public base-key (pbk_A)
     - The encapsulated keys from the AKE step (c_2, c_3, c_4)
     - The respective signature ($^{post}sig_A$)

  - Identifiers stating which of Bob's prekeys and signed prekeys she used (i, j)
- The header (h), containing:
  - The encapsulated asymmetric chain key (c_n)
  - The new asymmetric ratchet public keys $({}^{*}epk_A)$
  - The message number (n)
  - The previous chain length (pn)
  - The chain number for the sender chain(nc)[1]
- the ciphertext (c)
- and the respective tag(t)

The associated data (ad) now consists of both parties' public identity keys, the message header h and the signed encapsulated ciphertexts from the AKE step $(c_2, c_3, c_4, \ {}^{post}sig_A)$. The authenticated encryption of the message m is performed as in the original Signal protocol:

$$\begin{gathered} h \leftarrow header({}^{*}epk_A, c_n, n, pn, nc) \\ ad \leftarrow h||H({}^{**}ipk_A)||H({}^{**}ipk_B)||c_2||c_3||c_4||{}^{post}sig_A \\ (t, c) \leftarrow enc(mk, m, ad) \end{gathered} \tag{3.22}$$

Note that the encapsulated keys (c_i) and the respective signature $({}^{post}sig_X)$ are additionally used as associated data for the AEAD to prevent an adversary from replacing the original signature with their own (see [2]) and to prevent mix-and-match attacks on KEM combiners as described by Bindel et al. [3].

5. **Receive steps**: In this step the described message is decrypted and verified:

$$ad \leftarrow h||H({}^{**}ipk_A)||H({}^{**}ipk_B)||c_2||c_3||c_4||{}^{post}sig_A \tag{3.23}$$

$$m \leftarrow dec(mk, c, t, ad) \tag{3.24}$$

[1] This is an addition to the original protocol we introduced due to [1]. It facilitates calculating the sending chain key only in the send step, thereby reducing the lifetime of the new key and increasing security.

3.2 Protocol Flow

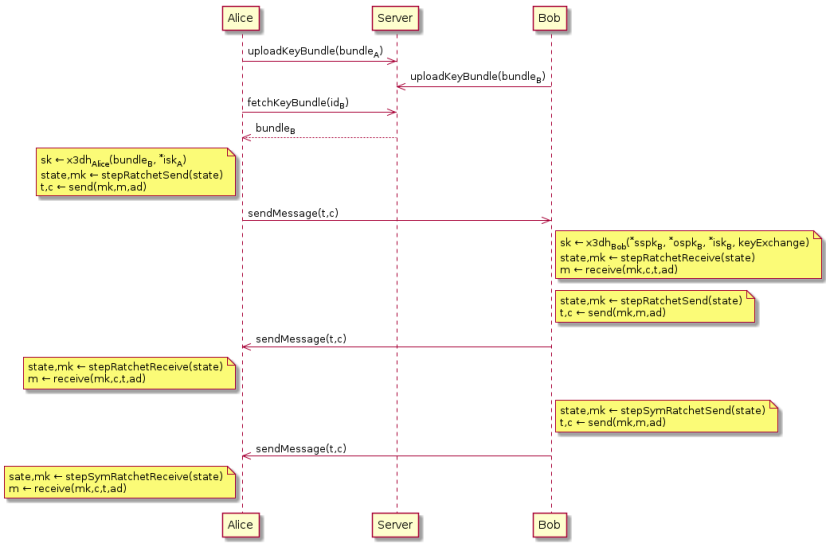

Figure 5: Protocol flow for the hybrid protocol.

Figure 5 depicts the flow of the hybrid protocol.When registering their Tutanota accounts, Alice and Bob generate their identity key pairs as well as n one-time key pairs and a signed ephemeral key. They then upload the key bundles to the server. When Alice wants to send an e-mail to Bob, with whom she did not communicate earlier, she first fetches his key bundle from the server. She then performs the authenticated key exchange by calling the function

$$x3dh_{Alice}(Bundle_B, {}^*isk_A).$$

This function expects Bob's key bundle and Alice's secret identity key and performs the steps from Equations 2.1,2.2, 2.3 as well as 3.13, 3.14 and 3.15.

After having derived the master secret Alice performs the ratchet steps. We assume that Alice (and Bob) maintains a state containing, among others[2]:

- Their own most recent pre- and post-quantum asymmetric ratchet key pairs $({}^*esk_{own}, {}^*epk_{own})$
- The other parties most recent pre- and post-quantum asymmetric ratchet public keys $({}^*epk_{remote})$
- The most recent encapsulated secret that was sent to the other party c_n

[2]We reduce this overview to the most important values needed for the core protocol and leave out values required e.g. to deal with out of order messages or to resend X3DH messages.

- The most recent root key (rk)
- The most recent chain keys for the sending and receiving chain (ck_s, ck_r)

This state is initialized with the sk from the X3DH protocol as rk and Bob's public signed prekey as $({}^{*}epk_{remote})$. The function

$$stepRatchetSend(state)$$

performs all ratchet steps as in Equations 2.6 as well as 3.19 and 3.21, for the asymmetric and 2.9 for the symmetric ratchet (sending chain). In addition to an updated state, the ratchet step outputs the message key for Alice's first message to Bob.

Alice now performs the send steps as in Equations 3.22 by calling

$$send(mk, m, ad)$$

and sends the resulting message and tag to Bob.

When Bob receives Alice's e-mail he extracts all (encapsulated) keys from the message and calls

$$x3dh_{Bob}({}^{*}sspk_B,\ {}^{*}ospk_B, {}^{*}isk_B, keyExchange)$$

using this $keyExchange$ and his private keys corresponding to the public keys from the key bundle as parameters. This executes all steps from Equations 2.5 as well as 3.18 and 3.15.

He initializes the state with the sk from the X3DH protocol as rk, his signed prekey pair as $({}^{*}esk_{own},\ {}^{*}epk_{own})$ and Alice's ephemeral key from the message as $({}^{*}epk_{remote})$. He calls

$$stepRatchetReceive(state),$$

which performs all steps from Equations 2.7 as well as 3.20 and 3.21, for the asymmetric and 2.9 for the symmetric ratchet (receiving chain). The function outputs the updated state and the message key.

Bob finally performs the receive steps as in Equations 3.23 by calling

$$receive(mk, c, t, ad).$$

If the message decrypts and verifies successfully the communication between Alice and Bob has been established, otherwise the protocol is aborted.

To exchange further messages Alice and Bob do not have to perform the authenticated key exchange again. When Bob replies to Alice he only executes

$$stepRatchetSend(state)$$

and

$$send(mk, m, ad)$$

as defined above. Alice calls

$$stepRatchetReceive(state)$$

and

$$receive(mk, c, t, ad).$$

In our example Bob sends another message without having received a reply from Alice. In this case the asymmetric ratchet cannot be stepped. Bob now calls

$$stepSymRatchetSend(state)$$

performing only Equation 2.9 and subsequently

$$send(mk, m, ad)$$

using the new message key. Alice calls

$$stepSymRatchetReceive(state)$$

again performing Equation 2.9 and

$$receive(mk, c, t, ad).$$

3.3 Instantiation

Table 1 gives an overview of the cryptographic algorithms we use in the hybrid Signal protocol for the pre- and post-quantum variant respectively.

Algorithm	**Pre-quantum instantiation**	**Post-quantum instantiation**
Asymmetric algorithms		
Diffie-Hellman / KEM	Curve25519 ECDH [12]	Kyber 786 [4]
Signatures	XEdDSA [15]	Dilithium 1280x1024 (deterministic) [7]
Symmetric algorithms		
AEAD	AES 256 CBC and HMAC [10]	AES 256 CBC and HMAC
KDF (sending/receiving chain)	HMAC	HMAC
KDF (root chain)	HKDF [11]	HKDF
Hash algorithm	Sha256	Sha256

Table 1: Cryptographic algorithms used in the hybrid protocol.

4 Deviation from the original signal protocol

Our hybrid protocol achieves the same security properties as the original Signal protocol for pre-quantum and almost the same security properties for post-quantum security. The necessary deviations described in the previous section lead to some restrictions. In this section we summarize these deviations and point out their consequences.

Replacing DH calculations by key encapsulations

- **Randomness contribution** Replacing DH calculations by key encapsulations usually implies that only one party provides randomness. However, it depends on the used algorithm. We are using Kyber KEM [4], where both parties contribute randomness to the generation of the shared secrets.

Replacing the initiator's DH calculation using the identity key with a signature As pointed out by the X3DH specification [14] if DH-based mutual authentication is replaced by signatures "this reduces deniability, increases the size of initial messages, and increases the damage done if ephemeral or prekey private keys are compromised, or if the signature scheme is broken." This holds for our proposal as well, though only partly as we only replace one DH calculation. We discuss this in the following:

- **Deniability** Because the message is signed, we provide cryptographic proof that the initiator did send the encapsulated keys included in the first message. They can therefore not deny the authorship of this message. For all following messages we, however, yield the same offline deniability as the original protocol. [3]

- **Security depends on signatures scheme instead of DH calculations** If the signature scheme is broken, the initiator can be impersonated. In the original version the security of the authentication relies on the DH calculation using the secret identity key.[4]

- **Damage on key compromise** The X3DH specification [14] mentions increased damage "if ephemeral or prekey private keys are compromised" as a consequence of replacing DH calculations with signatures. This is founded in the fact that if the DH calculations with the identity keys are left out, the compromise of ephemeral and signed private keys only would yield the combined shared secret. In our case, however, we still encapsulate with the other parties identity key. Therefore, there is no security reduction in our case.

- **Increased initial message size** The initial message becomes larger as it includes an additional signature. This adds up to the message growth caused by the additional symmetric keys and encapsulated secrets that have to be included in the hybrid variant. However, for e-mail keeping message sizes small is of less importance, as compared to instant messaging.

[3] Note that online deniability is explicitly not given in neither variant (see [14]).

[4] The receiver, who authenticates via the knowledge of the KEMs private identity key, cannot be impersonated in the post-quantum variant even if the signature scheme is broken.

5 Notations

$({}^{pre}isk_X,\ {}^{pre}ipk_X)$	Pre-quantum Identity key pair of party X: DH key pair.
$({}^{post}isk_X,\ {}^{post}ipk_X)$	Post-quantum Identity key pair of party X: Dilithium and Kyber key pair.
$({}^{pre}sspk_X^j,\ {}^{pre}sppk_X^j)$	Pre-quantum signed prekey key pair of party X: DH key pair signed with the DH identity key.
$({}^{post}sspk_X^j,\ {}^{post}sppk_X^j)$	Post-quantum signed prekey key pair of party X: Kyber key pair signed with the Dilithium identity key.
$({}^{pre}ospk_X^i,\ {}^{pre}oppk_X^i)$	Pre-quantum one-time prekey pairs of party X: DH key pair.
$({}^{post}ospk_X^i,\ {}^{post}oppk_X^i)$	Post-quantum one-time prekey pairs of party X: Kyber key pair.
$({}^{pre}esk_X,\ {}^{pre}epk_X)$	Pre-quantum ephemeral key pair of party X:DH key pair.
$({}^{post}esk_X,\ {}^{post}epk_X)$	Post-quantum ephemeral key pair of party X: Kyber key pair.
$({}^{pre}bsk_X,\ {}^{pre}bpk_X)$	One-time DH base key pair of party X needed only for the initial message
$sign({}^{pre}isk_X, data)$	XEdDSA signature on provided data using the private identity key.
$sign({}^{post}isk_X, data)$	Dilithium signature on provided data using the private identity.
$verify({}^{pre}ipk_X, data, signature)$	Verify XEdDSA signature on provided data using the public identity key.
$verify({}^{post}ipk_X, data, signature)$	Verify Dilithium signature on provided data using the public identity key.
$gen()$	Key generation of Kyber or DH keys.
$(c, k) \leftarrow encaps({}^{post}pk_X)$	Kyber key encapsulation using some public key of party X. The result is a new key and a respective ciphertext.
$k \leftarrow decaps({}^{post}sk_X, c)$	Kyber key decapsulation using ciphertext and some private key.
$DH({}^{pre}sk_X,\ {}^{pre}pk_Y)$	Diffie-Hellman calculation involving the private key of party$X$ and the public key of party Y.
$(ck, mk) \leftarrow KDF(ck)$	HMAC using ck as key. The result is a new chain key (ck) and a message key (mk).
$(rk, ck) \leftarrow KDF(rk, k_n)$	HKDF using rk as salt and the output of the asymmetric ratchet as input key. The result is new a root key (rk) and a chain key (ck).
$(c, t) \leftarrow enc(mk, m, ad)$	Authenticated encryption with associated data. The result is a ciphertext and a tag.
$m \leftarrow dec(mk, c, t, ad)$	Authenticated decryption using message key, ciphertext, tag and the associated data.
$header(data)$	Generates a message header containing the given data.
$encode({}^{pre}ipk_X)$	Encodes a public key into a byte sequence. Encoded public identity keys are used as associated data (ad).

Table 2: Notations used in the protocol description. While secret and public keys are abbreviated with sk and pk respectively, we use spk and ppk for secret and public prekeys. Pre-quantum keys are marked with ^{pre} and post-quantum keys with ^{post} respectively, an asterisk (*) indicates that both values are included separately and two asterisks (**) mark their concatenation.

References

[1] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. "The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol". In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. Lecture Notes in Computer Science. Springer, 2019, pp. 129–158. URL: https://doi.org/10.1007/978-3-030-17653-2%5C_5.

[2] Ross J. Anderson and Roger M. Needham. "Robustness Principles for Public Key Protocols". In: *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*. Ed. by Don Coppersmith. Vol. 963. Lecture Notes in Computer Science. Springer, 1995, pp. 236–247. URL: https://doi.org/10.1007/3-540-44750-4%5C_19.

[3] Nina Bindel et al. "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange". In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 903. URL: https://eprint.iacr.org/2018/903.

[4] Joppe W. Bos et al. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*. IEEE, 2018, pp. 353–367. URL: https://doi.org/10.1109/EuroSP.2018.00032.

[5] Jacqueline Brendel et al. "Challenges in Proving Post-Quantum Key Exchanges Based on Key Encapsulation Mechanisms". In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 1356. URL: https://eprint.iacr.org/2019/1356.

[6] Jacquline Brendel et al. *Post-quantum asynchronous deniable key exchange and the Signal handshake*. Cryptology ePrint Archive, Report 2021/769. https://eprint.iacr.org/2021/769. 2021.

[7] Léo Ducas et al. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.1 (2018), pp. 238–268. URL: https://doi.org/10.13154/tches.v2018.i1.238-268.

[8] Ines Duits. "The Post-Quantum Signal Protocol. Secure Chat in a Quantum World". MA thesis. University of Twente, 2019.

[9] Steven D. Galbraith et al. "On the Security of Supersingular Isogeny Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 63–91. URL: https://doi.org/10.1007/978-3-662-53887-6%5C_3.

[10] Dr. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. Feb. 1997. URL: https://rfc-editor.org/rfc/rfc2104.txt.

[11] Dr. Hugo Krawczyk and Pasi Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. RFC 5869. May 2010. URL: https://rfc-editor.org/rfc/rfc5869.txt.

[12] Adam Langley, Mike Hamburg, and Sean Turner. *Elliptic Curves for Security*. RFC 7748. Jan. 2016. URL: https://rfc-editor.org/rfc/rfc7748.txt.

[13] Moxie Marlinspike and Trevor Perrin. *The Double Ratchet Algorithm*. 2016. URL: https://www.signal.org/docs/specifications/doubleratchet/.

[14] Moxie Marlinspike and Trevor Perrin. *The X3DH Key Agreement Protocol*. 2016. URL: https://www.signal.org/docs/specifications/x3dh/.

[15] Trevor Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. 2016. URL: https://whispersystems.org/docs/specifications/xeddsa/.

[16] Nik Unger et al. "SoK: Secure Messaging". In: *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 232–249. URL: https://doi.org/10.1109/SP.2015.22.