Kyber KEX with 2-way Handshake

Stephan Müller

2023-09-04

Abstract

This document specifies the use of Kyber KEX with a 2-way handshake. The goal is to allow Kyber KEX as a replacement of Diffie-Hellman without changing existing protocol handshakes. A reference implementation of the algorithms is given with leancrypto.

Contents

List of Figures

List of Tables

1 Secure Connection Design

The secure connection is a protocol that is similar to the SSHv2 protocol with the difference that it uses quantum-computing-safe cryptographic algorithms.

The handshake as well as the processing of the payload data, however, is similar to the protocol specified in RFC4253.

The following sections outline the protocol specification.

1.1 Goal of Secure Connection Support

The following goals are achieved with the secure connection support:

- Rollback-protection: This is achieved for the handshake by the following properties:
  - Kyber implements the IND-CCA2 property which implies that every Kyber ciphertext is created using a 256 bit random number. By using fresh random numbers for the Kyber operation, a replay protection is provided. As such Kyber ciphertext is created by the client and the server, the replay protection is ensured by both sides of the communication link.
  - A client/server nonce is created once during session creation which is a 128 bit random value. The concatenation of the client and server nonce serves as a session identifier that is is retained even over a rekey operation. To validate the session identifier, it is inserted into the Kyber KDF operation that generates the shared secret.
  - At runtime, the use of an AEAD algorithm ensures non-repudiation for the bulk data transfer.
  - Finally, the session identifier is used as AAD for the protection of bulk data transfer.
- Perfect forward secrecy: This is achieved by using an ephemeral Kyber keypair for each connection. The client as well as the server both create a new Kyber keypair during each handshake. During rekey, a complete new handshake is performed which does not rely on any information negotiated during preceding handshakes.
- Confidentiality: This is achieved by using a symmetric cryptographic algorithm with a security strength of 256 bits.
- Integrity: This is achieved during bulk data transfer by using an AEAD algorithm with a 128 bit tag. During handshake, this is achieved by using the bi-directional authenticated Kyber key exchange (KEX) which rests on pre-shared keys which are considered to be exchanged in a trustworthy manner. Note, other cryptographic network protocols implement the handshake integrity (and the resulting authentication of the remote peer) using a signature in a separate step after the KEX handshake is completed. This difference does not alter the KEX handshake operation outlined in this section. The reader may ignore the signature operation when only

assessing the KEX operation. When ignoring the signature verification part, the KEX handshake may be reused in other protocols.

- Authenticity / non-repudiation: This is achieved during handshake with a bi-directional authentication based on the authenticated Kyber KEX operation. This authentication is intrinsic to the Kyber operation at a similar level as authentication is part of an AEAD symmetric algorithm.
- Freshness: After at latest the expiry of 1 hour or after the encryption / decryption of 1GB of data (whichever occurs earlier), a complete new re-key operation must be performed.
- Quantum-computing-resistant: This is achieved by only using quantum-computing-resistant algorithms.
- Lack of entropy on either side is harmless: Both sides of the connection contribute equally to the security strength of the shared secret. If one side lacks sufficient entropy, this is offset by the respective other side as each side individually ensures the security strength of the shared secret. This is ensured by using the Kyber key exchange mechanism (KEX) as opposed to the Kyber key encapsulation mechanism (KEM).

1.2 Key Agreement

1.2.1 Kyber KEX - Basic Concept

The following table outlines the general stages of an bi-directional authenticated Kyber Key Exchange (KEX) which are also all followed by the secure connection implementation.

Table 1: Basic Kyber Key Exchange Steps

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` ;	Send public key `pk_r` ;
	Receipt of `pk_r`	Receipt of `pk_i`
3	Initiate key exchange - generate: ephemeral public key `pk_e_i`, ephemeral ciphertext `ct_e_i`, KEM shared secret `tk`, ephemeral secret key `sk_e`.	
4	Send KEX data `pk_e_i`, `ct_e_i`	Receipt of `pk_e_i`, `ct_e_i`

Step	Alice (Inititiator)	Bob (Responder)
5		Calculate: ephemeral ciphertext `ct_e_r_1`, ephemeral ciphertext `ct_e_r_2`, Shared Secret `ss`
6	Receipt of `ct_e_r_1`, `ct_e_r_2`	Send `ct_e_r_1`, `ct_e_r_2`
7	Calculate: Shared Secret `ss`	

Note that steps 1 and 2 are to be performed once to generate a set of static key pairs and to securely exchange the public keys which is therefore marked as gray as it is not directly part of a given Kyber KEX operation. See section Implied Authentication for details about the implications of those keys and the used exchange method.

The basic Kyber KEX steps are compressed such that only two network exchanges are required to perform a Kyber KEX handshake. By combining several steps into one, the 2-way handshake for a Kyber KEX operation is achieved which is illustrated with Figure [1].

1.2.2 Prerequisites

The following operations must be performed *before* any handshake operation commences. These operations are therefore outside of this protocol specification:

- Exchanging the Kyber static public keys `pk_i` and `pk_r` generated during step 1 to authenticate the respective remote peer in a way that establishes or maintains trust, and
- Giving each peer a unique name to allow resolving the exchanged Kyber public key. Usually this can be achieved by using the server's full qualified domain name or IP address used to reach the server whereas the client uses an arbitrary name that is sent to the server during handshake.

1.2.3 1st Operation: Client - Initiator Handshake

The client has the following information available or generated before:

1. an arbitrary client name which is sent to the server to allow the server finding the proper corresponding client public key `pk_i`,
2. a Kyber keypair `pk_i` and `sk_i` where the public key `pk_i` is registered with the server, and
3. the server's Kyber public key `pk_r`.

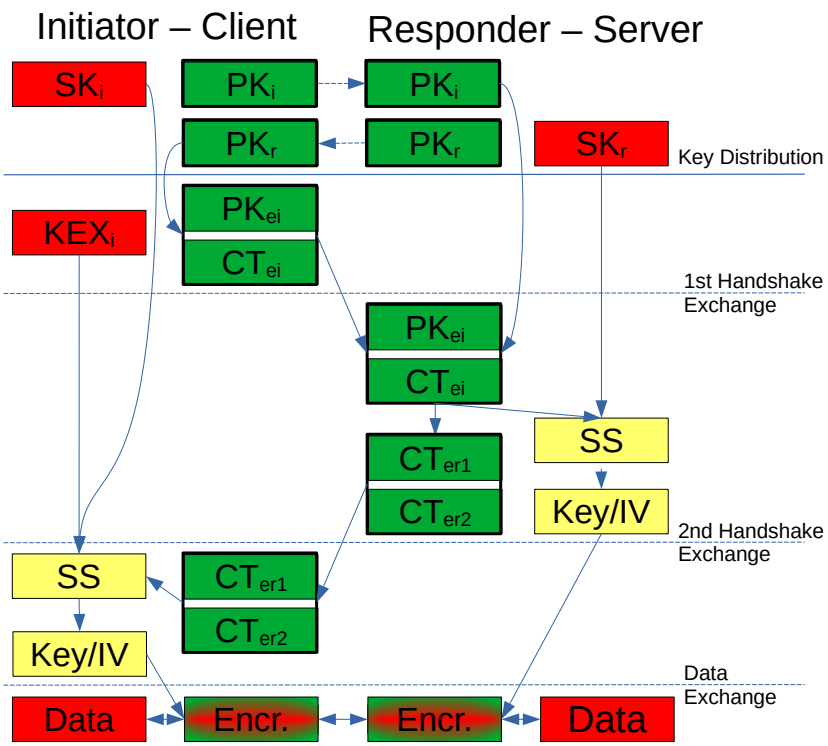

Figure 1: 2-way Kyber KEX

The client generates the following information for every handshake anew:

1. During the first handshake, a 128 bit client nonce using a random number generator. This nonce is retained for all subsequent handshakes to identify a session.
2. perform a Kyber KEX encapsulation initiation operation with `pk_r` and obtain the Kyber KEX data `pk_e_i` and `ct_e_i`

The client sends the following data to the server as part of the initiator handshake:

1. protocol version (version number 1 is used)
2. an arbitrary client name,
3. the client nonce,
4. Kyber KEX data `pk_e_i`,
5. Kyber KEX data `ct_e_i`

The red-marked parts of the table are covered by this operation:

Table 2: 1st Operation of Kyber Key Exchange

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` ;	Send public key `pk_r` ;
	Receipt of `pk_r`	Receipt of `pk_i`
3	Initiate key exchange - generate:	
	ephemeral public key `pk_e_i`,	
	ephemeral ciphertext `ct_e_i`,	
	KEM shared secret `tk`,	
	ephemeral secret key `sk_e`.	
4	Send KEX data `pk_e_i`, `ct_e_i`	Receipt of `pk_e_i`, `ct_e_i`
5		Calculate: ephemeral ciphertext `ct_e_r_1`, ephemeral ciphertext `ct_e_r_2`, shared secret `ss`
6	Receipt of `ct_e_r_1`, `ct_e_r_2`	Send `ct_e_r_1`, `ct_e_r_2`
7	Calculate: Shared Secret `ss`	

1.2.4 2nd Operation: Server - Responder Handshake

The server has the following information available or generated before:

1. a Kyber keypair `sk_r` and `pk_r` where the public key `pk_r` is registered with the client, and
2. the client's Kyber public key `pk_i`.

The server generates the following information for every handshake anew:

1. During the first handshake, a 128 bit server nonce using a random number generator. This nonce is retained for all subsequent handshakes to identify a session.

The server receives the client data and performs the following operations:

1. verify that the protocol version is 1
2. establish the session identifier which concatenates client nonce and server nonce
3. perform a Kyber KEX responder operation with:
   a. `pk_i`,
   b. the initiator Kyber KEX data, and
   c. the session identifier is inserted into the Kyber KDF after the Kyber-internal data was added to the KDF

   and obtain the new Kyber KEX data `ct_e_r_1`, `ct_e_r_2` as well as the shared secret of 768 bits.
4. initiate 2 AEAD algorithm instances:
   a. client-to-server AEAD instance using the 384 leftmost bits from the shared secret which is segmented into the 256 leftmost bits used as key and the 128 rightmost bits as IV,
   b. server-to-client AEAD instance using the 384 rightmost bits from the shared secret which is segmented into the 256 leftmost bits used as key and the 128 rightmost bits as IV.
5. destroy Kyber ephemeral shared secret data.

The server sends the following data to the client:

1. the server nonce,
2. Kyber KEX data `ct_e_r_1`,
3. Kyber KEX data `ct_e_r_2`, and
4. a return code of the server's operation.

The red-marked parts of the table are covered by this operation:

Table 3: 2nd Operation of Kyber Key Exchange

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` ;	Send public key `pk_r` ;
	Receipt of `pk_r`	Receipt of `pk_i`
3	Initiate key exchange - generate: ephemeral public key `pk_e_i`, ephemeral ciphertext `ct_e_i`, KEM shared secret `tk`, ephemeral secret key `sk_e`.	
4	Send KEX data `pk_e_i`, `ct_e_i`	Receipt of `pk_e_i`, `ct_e_i`
5		Calculate: ephemeral ciphertext `ct_e_r_1`, ephemeral ciphertext `ct_e_r_2`, shared secret `ss`
6	Receipt of `ct_e_r_1`, `ct_e_r_2`	Send `ct_e_r_1`, `ct_e_r_2`
7	Calculate: Shared Secret `ss`	

1.2.5 3rd Operation: Client - Initiator Handshake Completion

The client receives the server data and performs the following operations:

1. establish the session identifier which concatenates client nonce and server nonce
2. perform a Kyber KEX initiator completion operation with:
   a. the ephemeral initiator Kyber KEX data obtained during the 1st operation, and
   b. the session identifier is inserted into the Kyber KDF after the Kyber-internal data was added to the KDF

   and obtain the shared secret of 768 bits.
3. initiate 2 AEAD algorithm instances:
   a. client-to-server AEAD instance using the 384 leftmost bits from the shared secret which is segmented into the 256 leftmost bits used as key and the 128 rightmost bits as IV,

   b. server-to-client AEAD instance using the 384 rightmost bits from the shared secret which is segmented into the 256 leftmost bits used as key and the 128 rightmost bits as IV.

4. destroy ephemeral initiator Kyber KEX data obtained during the 1st operation that was used for this handshake.

The red-marked parts of the table are covered by this operation:

Table 4: 3rd Operation of Kyber Key Exchange

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i`;	Send public key `pk_r`;
	Receipt of `pk_r`	Receipt of `pk_i`
3	Initiate key exchange - generate: ephemeral public key `pk_e_i`, ephemeral ciphertext `ct_e_i`, KEM shared secret `tk`, ephemeral secret key `sk_e`.	
4	Send KEX data `pk_e_i`, `ct_e_i`	Receipt of `pk_e_i`, `ct_e_i`
5		Calculate: ephemeral ciphertext `ct_e_r_1`, ephemeral ciphertext `ct_e_r_2`, shared secret `ss`
6	Receipt of `ct_e_r_1`, `ct_e_r_2`	Send `ct_e_r_1`, `ct_e_r_2`
7	Calculate: Shared Secret `ss`	

1.2.6 Implied Authentication

In addition to the secure establishment of session keys, the KEX operation offers another key aspect: it authenticates the remote peer cryptographically. This implies that there is no need for a subsequent signature-based authentication step.

This authentication is based on the fact that static public keys are used as part of the KEX operation. Only when the static public keys are correct and the remote peer possesses the associated secret key, the shared secrets can be established. This fact is the implied authentication, i.e. the authentication by means of the static key.

These static public keys outlined in the tables above can be exchanged in the following ways and the following implications:

1. The public keys are exchanged in a separate exchange process, such as using them as pre-shared keys, where the process guarantees (a) the keys are generated with sufficient entropy, and (b) the public keys are transmitted with their integrity and authenticity preserved. In this case, the KEX not only offers an implied authentication, but also provides protection against weak entropy present by either the initiator or the responder. In case of a (temporary) weak entropy to generate the ephemeral key pair, the entropy present in the static key and its resulting cipher text is considered to offset the lack of entropy. Yet, such lack of entropy should only be temporary.

2. The public keys are exchanged immediately before the KEX handshake using the same network channel as the KEX. The recipient of the respective public key has a means to establish the authenticity of the key, such as X.509 operations. This approach provides the implied authentication with the KEX operation considering that the recipient of the public keys can unambiguously establish the remote peer's identity and thus verify the received key.

It is also permissible that only one side (e.g. the responder) submits a public key. In this case, a unilateral authentication is offered. This is akin to standard TLS usage where a client is able to verify the authenticity of the server. Yet, the server is commonly not able to verify the respective client.

The use of the implied authentication is further elaborated by AuthKEM.

