Kyber KEX with 2-way Handshake

April 16, 2024

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.

This is a documentation of the implementation provided by the CRYSTALS Kyber authors supplemented with a use case illustrating how Kyber KEX is used.

Contents

List of Figures

List of Tables

1 Kyber Key Exchange Mechanism

The Kyber key exchange (KEX) mechanism is implemented by the CRYSTALS Kyber authors with the file `kex.c` provided with their code distribution. Yet, it is undocumented and its usage may not be immediately obvious to the reader.

The following abstract provides a specification of the Kyber KEX mechanism.

1.1 Implicit Goals of Kyber KEX

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.
- 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.
- 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.
- 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).
- User identification and authentication: By using the Kyber key exchange mechanism (KEX) with pre-shared keys, user authentication is achieved. Only when a user uses the correct pre-shared key with the KEX mechanism, a common shared secret is established.

1.2 Key Exchange

1.3 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.

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` Receipt of `pk_r`	Send public key `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`	

Table 1: Basic Kyber Key Exchange Steps

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 1.3.5 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.

1.3.1 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

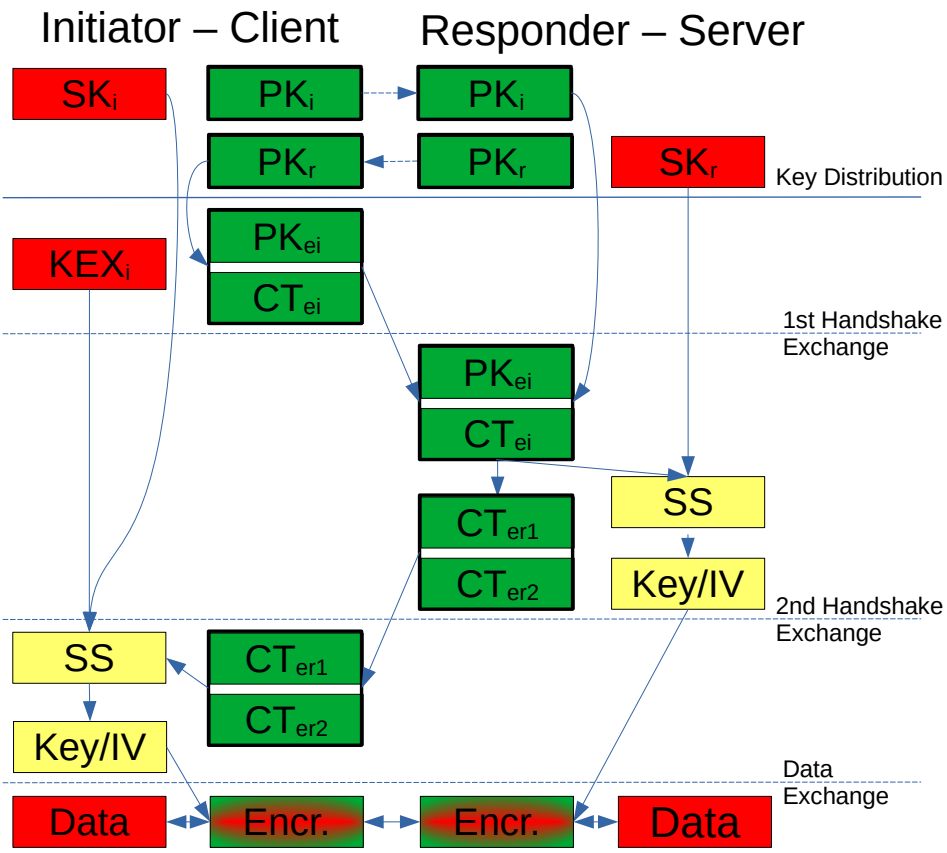

Figure 1: 2-way Kyber KEX

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.

1.3.2 First Operation: Client — Initiator Handshake

The client has the following information available or generated before:

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

The client generates the following information for every handshake anew:

1. Perform a Kyber KEX encapsulation initiation operation with `pk_r` and obtain the public Kyber KEX data `pk_e_i` and `ct_e_i` and the secret Kyber KEX data of `tk` and `sk_e` to be retained for the third operation outlined in section 1.3.4. This Kyber KEX encapsulation operation is implemented with the following steps:
   (a) Kyber key generation to obtain the client ephemeral Kyber public key pair `pk_e_i` and `sk_e`,
   (b) Kyber KEM encapsulation operation deriving the Kyber ciphertext `ct_e_i` and the Kyber shared secret `tk` from server's Kyber public key `pk_r`.

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

1. Kyber KEX data consisting of:
   (a) ephemeral Kyber public key `pk_e_i`,
   (b) ephemeral Kyber ciphertext `ct_e_i`.

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

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` Receipt of `pk_r`	Send public key `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`	

Table 2: First Operation of Kyber Key Exchange

1.3.3 Second 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 static public key `pk_i`.

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

1. Perform a Kyber KEX responder operation with the following input to generate a shared secret using the Kyber KEX data received from the client. The Kyber KEX responder operation is defined as:
   (a) Kyber KEM encapsulation of the client ephemeral Kyber public key `pk_e_i` to generate the ephemeral Kyber ciphertext `ct_e_r_1` and the private Kyber shared secret `ss_0`.
   (b) Kyber KEM encapsulation of the client's static Kyber public key `pk_i` to generate the ephemeral Kyber ciphertext `ct_e_r_2` and the private Kyber shared secret `ss_1`.
   (c) Kyber KEM decapsulation of the client ephemeral Kyber ciphertext `ct_e_i` and the server's static Kyber secret key `sk_r` to yield the Kyber shared secret `ss_2`.
2. The different Kyber shared secrets generated in the preceding step are concatenated in the given order. The concatenated data is inserted into a

KDF, such as one from the KDFs specified in [1, 2, 3] to generate a key and IV for a symmetric algorithm, and possibly a MAC key for an integrity algorithm instantiated to protect the actual user data to be communicated.

3. destroy Kyber ephemeral shared secret data.

The server sends the following data to the client:

1. Kyber KEX data consisting of:
   (a) ephemeral Kyber ciphertext `ct_e_r_1`,
   (b) ephemeral Kyber ciphertext `ct_e_r_2`

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

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` Receipt of `pk_r`	Send public key `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`	

Table 3: Second Operation of Kyber Key Exchange

1.3.4 Third Operation: Client - Initiator Handshake Completion

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

1. Perform a Kyber KEX initiator completion operation with the following input to generate a shared secret using the Kyber KEX data received from the server. The Kyber KEX initiator finalization operation is defined as:
   (a) Kyber KEM decapsulation of the server ephemeral Kyber ciphertext `ct_e_r_1` and the client's ephemeral Kyber secret key `sk_e` to yield the Kyber shared secret `ss_0`.

(b) Kyber KEM decapsulation of the server ephemeral Kyber ciphertext `ct_e_r_2` and the client's static Kyber secret key `sk_i` to yield the Kyber shared secret `ss_1`.

2. The different Kyber shared secrets generated in the preceding step are concatenated in the given order supplemented by concatenating the `tk` Kyber shared secret value generated during the first step. The concatenated data is inserted into a KDF, such as one from the KDFs specified in [1, 2, 3] to generate a key and IV for a symmetric algorithm, and possibly a MAC key for an integrity algorithm instantiated to protect the actual user data to be communicated. The selected KDF must be identical to the one used by the server.

3. destroy ephemeral initiator Kyber KEX data received from the server and obtained during the 1st operation that was used for this handshake.

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

Step	Alice (Inititiator)	Bob (Responder)
1	Key gen: `pk_i`, `sk_i`	Key gen: `pk_r`, `sk_r`
2	Send public key `pk_i` Receipt of `pk_r`	Send public key `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`	

Table 4: Third Operation of Kyber Key Exchange

1.3.5 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. 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. 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 implied authentication of KEX only results in different "shared" secrets on either side of the communication channel if the wrong pre-shared keys are used (e.g. the client uses a different key than present on the server). There is no error detected at during the KEX operation. However, when using an AEAD algorithm with the "shared" secret data, the authentication error is enforced: The authentication effectively is provided by the authenticating part of the AEAD decryption operation, i.e. when one side (commonly the client as it sends first the data) sends data, the respective other side cannot decrypt it can returns a decryption failure. That failure is now detected by the receiving end. As the failure is detected with the first data chunk (and any following data chunk that may be sent), the failure is detected as soon as its detection is required, i.e. when the communication commences.

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

References

[1] Lily Chen. *NIST SP 800-108r1, Recommendation for Key Derivation Using Pseudorandom Functions*. Revistion 1 edition, August, 2022.

[2] Richard Davis Elaine Barker, Lily Chen. *NIST Special Publication 800-56C Recommendation for Key-Derivation Methods in Key-Establishment Schemes*. Revistion 2 edition, August, 2020.

[3] P. Eronen H. Krawczyk. *RFC5869 HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. May, 2010.

