Symbolic analysis and verification of Hybrid Post-Quantum TLS 1.2

Duong Dinh Tran, Kazuhiro Ogata, Santiago Escobar, Sedat Akleylek, and Ayoub Otmani

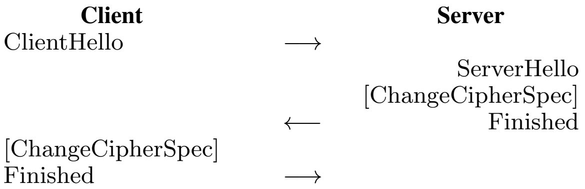

Fig. 3. Messages exchanged in an abbreviated handshake of Hybrid PQ TLS

Appendix A
Abbreviated handshake mode

When client A and server B want to resume a previously established session, instead of exchanging the complete handshake messages, the two principals can perform an abbreviated handshake, with messages exchanged depicted in Fig. 3. Client A first sends a ClientHello using the session ID of the session to be resumed. Upon receiving that message, server B sends a ServerHello message with the same session ID value after checking that there exists a matching session ID. Then, server B jumps to sending a ChangeCipherSpec message followed by a Finished message. Note that, the handshake keys are computed from the old master secret saved from the last full handshake and the new random values exchanged in the two new Hello messages. Similarly, client A sends their ChangeCipherSpec message followed by a Finished message.

Appendix B
Cipher suites for the extended protocol

Table III defines the cipher suites used in the extended protocol. Each cipher suite name consists of the hybrid key exchange method name, followed by the hybrid signature algorithm name, followed by the cipher and hash algorithms. For instance, with the fifth cipher suite in the Table, ECDH and ML-KEM are used as the hybrid key exchange method, ECDSA and ML-DSA are used as the hybrid signature algorithm, and AES-128 and SHA-256 are used for encryption and hash algorithms.

Appendix C
Counterexample of server authentication property

The original protocol does not enjoy the server authentication property due to the classical digital signature algorithm suffers from quantum attacks. A counterexample of this property was found, which is available on the webpage[1]. Readers can inspect and/or execute it to confirm its correctness. The diagram in Fig. 4 outlines how the counterexample happens. There exists an honest client Alice and an honest server Bob. The intruder in the middle of the connection between them, acts as server Bob in communication with Alice, and acts as client Alice in communication with Bob. Alice sends a ClientHello message to Bob. In the response, Bob sends back to Alice a ServerHello message followed by a Certificate message. The intruder grasps Bob's certificate and from now on she impersonates Bob to exchange messages with Alice. Under our assumption, the intruder breaks the security of the classical signature algorithm to get Bob's long-term private key (private signature signing key). She generates two ephemeral secret keys `k` and `k'`, computes the corresponding public keys `pk` and `pk'`, and signs them by Bob's long-term private key. She impersonates Bob to send the two public keys and the signature to Alice. Alice verifies the received signature, and then sends back to Bob (but actually, the intruder) her ECDH public key `pk2` and PQ KEM encapsulation `en` followed by her Finished message. In this state, it is true that Alice has sent the Finished message to Bob, indicating that she accepted the two public keys `pk`, `pk'`, and the signature apparently received from Bob, but Bob has never sent them to anyone. Thus, the server authentication property captured by the predicate `auth` in Section V does not hold.

Formal specifications and proofs are available on Github

We make our formal specifications and proofs publicly available at https://github.com/duongtd23/HB-PQ-TLS12-analysis. From this webpage, readers can also find the elaborations on our formal specifications, the instructions to execute our proofs with CafeOBJ and CafeInMaude to re-check our verification, and the detailed guidelines to replicate the proofs.

[1] https://github.com/duongtd23/HB-PQ-TLS12-analysis

TABLE III
CIPHER SUITES FOR THE EXTENDED PROTOCOL

Cipher suite	**Value**
TLS_ECDHE_BIKE_ECDSA_MLDSA_WITH_AES_128_GCM_SHA256	{ 0xFF, 0x01 }
TLS_ECDHE_BIKE_ECDSA_MLDSA_WITH_AES_256_GCM_SHA384	{ 0xFF, 0x02 }
TLS_ECDHE_BIKE_RSA_MLDSA_WITH_AES_128_GCM_SHA256	{ 0xFF, 0x03 }
TLS_ECDHE_BIKE_RSA_MLDSA_WITH_AES_256_GCM_SHA384	{ 0xFF, 0x04 }
TLS_ECDHE_MLKEM_ECDSA_MLDSA_WITH_AES_128_GCM_SHA256	{ 0xFF, 0x09 }
TLS_ECDHE_MLKEM_ECDSA_MLDSA_WITH_AES_256_GCM_SHA384	{ 0xFF, 0x0A }
TLS_ECDHE_MLKEM_RSA_MLDSA_WITH_AES_128_GCM_SHA256	{ 0xFF, 0x0B }
TLS_ECDHE_MLKEM_RSA_MLDSA_WITH_AES_256_GCM_SHA384	{ 0xFF, 0x0C }

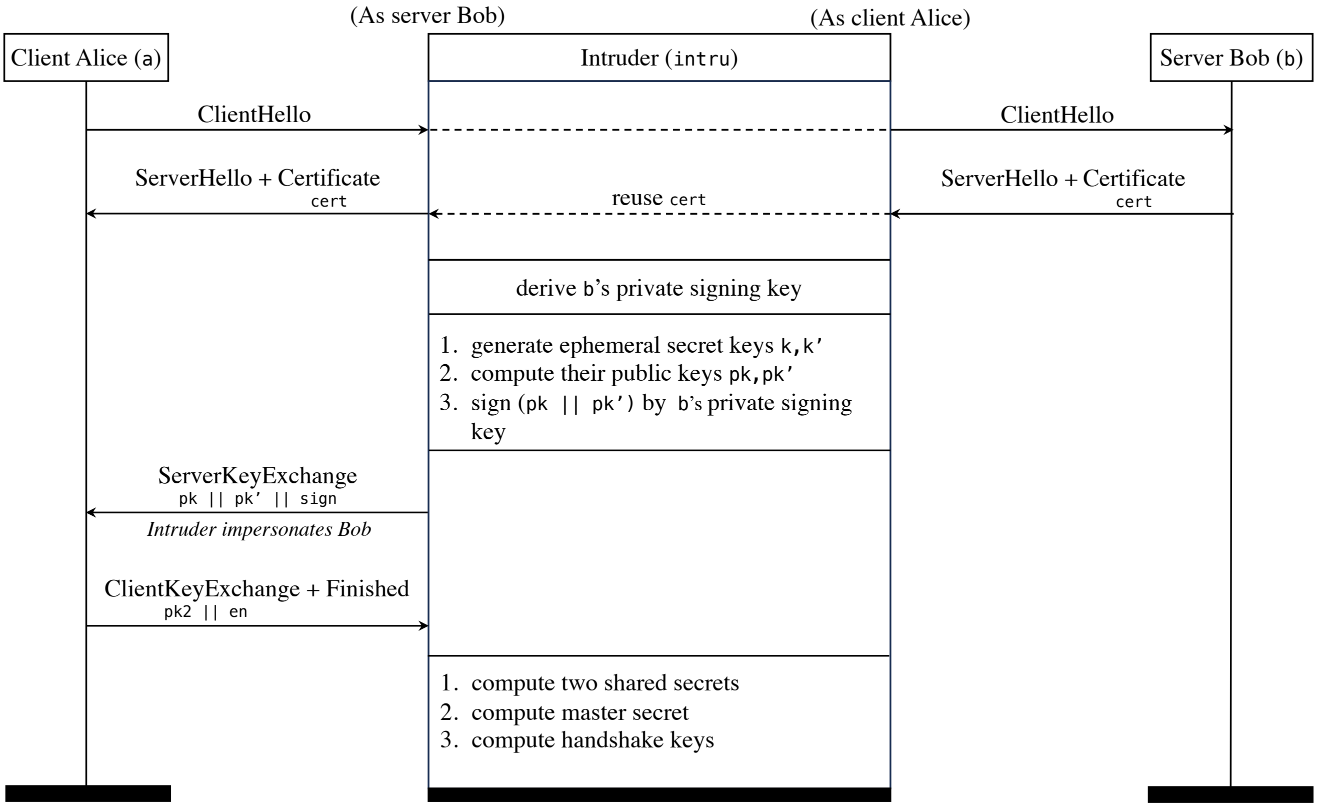
(As server Bob)
(As client Alice)
Client Alice (a)
Intruder (intru)
Server Bob (b)
ClientHello
ClientHello
ServerHello + Certificate
cert
reuse cert
ServerHello + Certificate
cert
derive b's private signing key
1. generate ephemeral secret keys k,k'
2. compute their public keys pk,pk'
3. sign (pk || pk') by b's private signing key
ServerKeyExchange
pk || pk' || sign
Intruder impersonates Bob
ClientKeyExchange + Finished
pk2 || en
1. compute two shared secrets
2. compute master secret
3. compute handshake keys

Fig. 4. The server authentication property is not met in the original protocol.