Four-in-a-Row Online – Project Report

Mirko Laruina, Riccardo Mancini

June 15, 2020

Contents

1 Project Specification

2 Communication Protocol

2.1 Description

The communication protocol is the same for both the client-server and client-client communications. It has been adapted from TLS 1.3 (RFC 8446[1]). The choice of this adaptation is due to the fact that the requested specifications closely match the features of TLS (e.g. perfect forward secrecy, authentication, message integrity, etc.).

The communication protocol is split in two parts:

- **handshake protocol**: establishes a shared key among the two parties.
- **record protocol**: sends an encrypted message to the other party.

2.1.1 Handshake Protocol

The differences of the proposed handshake protocol from TLS 1.3 handshake protocol using ECDH are the following:

- **No cipher suite negotiation** since we are going to always use the same algorithms.
- **No certificate exchange**: certificates are exchanged before the connection (either by explicitly asking for it, e.g. client connecting to the server, or from a previous knowledge, e.g. clients receiving the peer certificate from the server at the start of the game).
- **certificate verification and finished messages are collapsed in the same message**: it would have been redundant to send both HMAC and DS. Furthermore, the DS is computed on just the key parameters (ephemeral public keys, nonces and identities) instead of the whole communication transcript.

Here are the steps that the parties A and B need to perform, shown also in figure 1:

1. **A generates** her random nonce N_A and her ECDH ephemeral public key K_A^E.
2. **A sends to B** her identifier A, B's identifier B, N_A and K_A^E.
3. **B generates** his random nonce N_B and his ECDH ephemeral public key K_B^E.
4. **B computes** the digital signature using his public key K_B over the plaintext:

$$(A, B, N_A, K_A^E, N_B, K_B^E)$$

5. **B sends to A** his identifier B, A's identifier A, N_B, K_B^E and his digital signature S_B.
6. **A verifies** B's signature.
7. **A computes** the digital signature using her public key K_A over the same plaintext as B did.
8. **A sends to B** the digital signature.
9. **B verifies** A's signature.

[1] `https://tools.ietf.org/html/rfc8446`

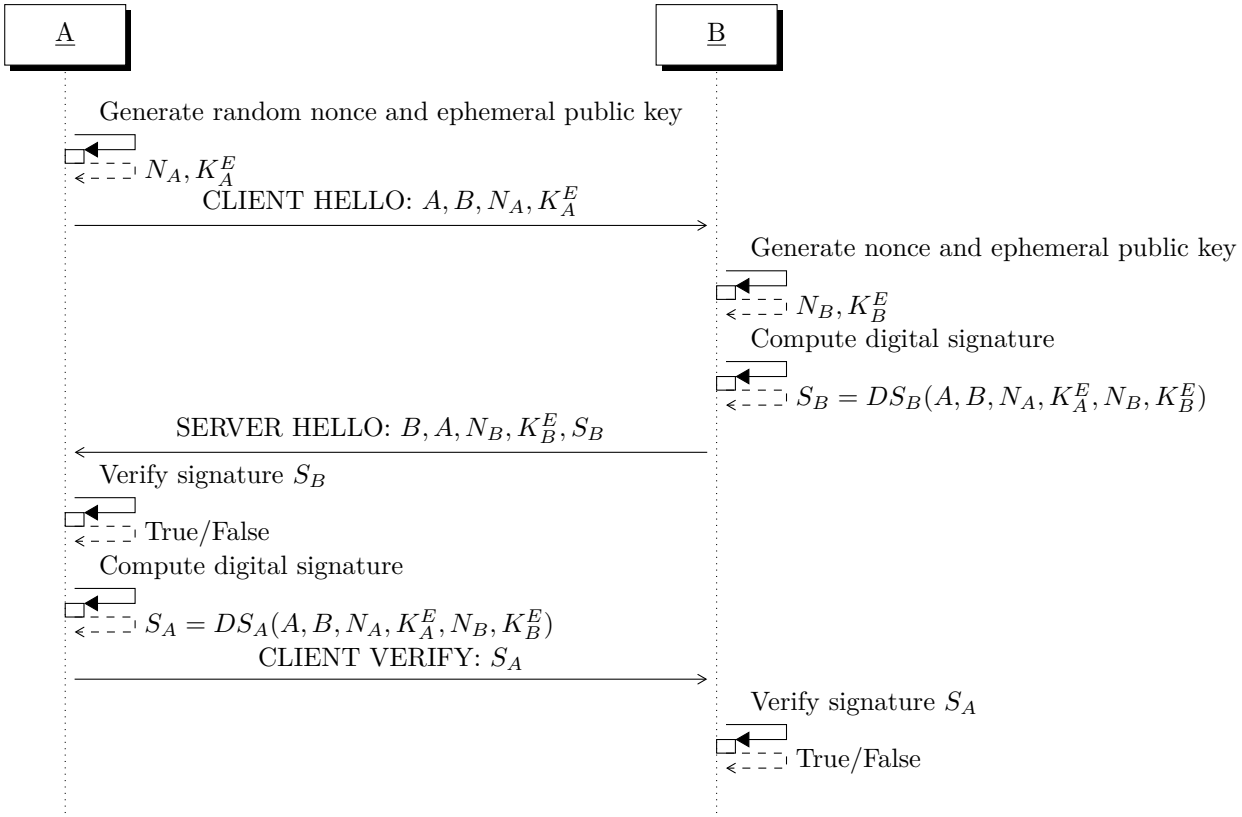

Figure 1: Sequence diagram of the handshake protocol. A and B are the identities of the two parties, N_A and N_B are random nonces, K_A^E and K_B^E are ECDH ephemeral public keys, K_A and K_B are the parties' public keys, S_A and S_B are the digital signatures.

2.1.2 Record Protocol

2.2 BAN-logic proof

2.2.1 Real protocol

$$
\begin{aligned}
&M1\ A \rightarrow B : A, B, N_A, K_A^E \\
&M2\ B \rightarrow A : B, A, N_B, K_B^E, \{h(A, B, N_A, N_B, K_A^E, K_B^E)\}_{k_B^{-1}} \\
&M3\ A \rightarrow B : \{h(A, B, N_A, N_B, K_A^E, K_B^E)\}_{k_A^{-1}}
\end{aligned}
$$

2.2.2 Assumptions

$$
\begin{aligned}
&A \mid\equiv \overset{K_B}{\longmapsto} B && B \mid\equiv \overset{K_A}{\longmapsto} A \\
&A \mid\equiv \#(\overset{K_A^E}{\longmapsto} A) && B \mid\equiv \#(\overset{K_B^E}{\longmapsto} B) \\
&A \mid\equiv \#(N_A) && B \mid\equiv \#(N_B) \\
&A \mid\equiv B \Rightarrow \overset{K_B^E}{\longmapsto} B && B \mid\equiv A \Rightarrow \overset{K_A^E}{\longmapsto} A \\
&A \mid\equiv B \Rightarrow N_B && B \mid\equiv A \Rightarrow N_A
\end{aligned}
$$

2.2.3 Expected Conclusions

$$A \mid\equiv (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B) \qquad B \mid\equiv (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B)$$

$$A \mid\equiv B \mid\equiv (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B) \qquad B \mid\equiv B \mid\equiv (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B)$$

2.2.4 Idealized protocol

$$M1\ A \to B : N_A, \overset{K_A^E}{\longmapsto} A$$

$$M2\ B \to A : \{N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B\}_{k_B^{-1}}$$

$$M3\ A \to B : \{N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B\}_{k_A^{-1}}$$

2.2.5 Proof

After message 1 From the first message, B just knows SOMEONE sent him a nonce and an ephemeral public key. He will use these values in later exchanges but at the moment he knows nothing about the sender.

$$M1\ A \to B : N_A, \overset{K_A^E}{\longmapsto} A \qquad \text{B does not know anything about the sender}$$

$$B \triangleleft (N_A, \overset{K_A^E}{\longmapsto} A)$$

After message 2 From this message A is able to verify B's identity and has confirmation that B actually knows the nonce and her ephemeral public key. A also receives B's nonce and ephemeral public key.

$$M2\ B \to A : \{N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B\}_{k_B^{-1}} \qquad \text{apply message meaning postulate}$$

$$A \mid\equiv B \mid\sim (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B) \qquad \text{apply nonce verification postulate}$$

$$A \mid\equiv B \mid\equiv (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B) \qquad \text{furthermore, due to juristiction postulate}$$

$$A \mid\equiv (N_B, \overset{K_B^E}{\longmapsto} B)$$

After message 3 From this message B is able to verify A's identity and has confirmation that A actually knows the nonce and his ephemeral public key.

$$M3\ A \to B : \{N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B\}_{k_A^{-1}} \qquad \text{apply message meaning postulate}$$

$$B \mid\equiv A \mid\sim (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B) \qquad \text{apply nonce verification postulate}$$

$$B \mid\equiv A \mid\equiv (N_A, N_B, \overset{K_A^E}{\longmapsto} A, \overset{K_B^E}{\longmapsto} B) \qquad \text{furthermore, due to juristiction postulate}$$

$$B \mid\equiv (N_A, \overset{K_A^E}{\longmapsto} A)$$

2.2.6 Final remarks

It is worthwhile to note that the nonces N_A, N_B play no role in the proof of the protocol. In fact, the nonce verification postulate could be applied also on the ephemeral public keys (since they are fresh too). Furthermore, the shared DH secret is guaranteed to always be different since both parties choose an ephemeral public key at random at every handshake. Therefore, we could remove the random nonces from the protocol without loss of security.

3 Messages

3.1 REGISTER

Registers user to the server.