Cryptography
Project 2: Chat Client

June 13, 2024

Nguyen Duy An 20214943, Duong Hong Nam 20214967, Vu Hoang Ngoc 20210646

---

1. In our implementation, Alice and Bob increment their Diffie-Hellman ratchets every time they exchange messages. Could the protocol be modified to have them increment the DH ratchets once every ten messages without compromising confidentiality against an eavesdropper (i.e., semantic security)? ()

   Ans: In theory, it would be possible to modify the protocol such that Alice and Bob increase their Diffie-Hellman ratchets once per 10 messages without sacrificing secrecy against an eavesdropper, but in practice, this is not possible. Consider the following scenario: Alice and Bob both restarted their DH ratchets after every 10 messages, resulting in the usage of DH public keys from the same interval numerous times during the session. In this case, an eavesdropper might acquire the secret key and thereby jeopardize the secrecy of subsequent messages.

2. What if they never update their DH keys at all? Please explain the security consequences of this change with regards to Forward Secrecy and Break-in Recovery. ()

   Ans: Ephemeral Diffie-Hellman uses different key pairs each time the protocol is run. This gives the connection perfect foward secrecy, because even if a key is compromised in the future, it cannot be used to decrypt all of the past messages. For our case where there is no update of DH keys at all, there will be experienced security consequences since the forward secrecy if limited and some keys can be used to decrypt past messages meaning there is great possibility for data loss due to lack of updates.
   Diffie-Hellman exchange will not give a provision of authentication of the communicating parties and thus is vulnerable to man-in-the-middle attack. For this case there can be break-in recovery according to the security measures set up by the institution. Several policies for recovery mechanism can be used like having some heavy backups. To some extents the recovery may not be possible due to the extent of the attack and preparedness of the institution.

3. Consider the following conversation between Alice and Bob, protected via the Double Ratchet Algorithm according to the spec:

   ALICE: Hey Bob, can you send me the l o c k e r combo?

   ALICE: I need t o g e t my l a p t o p.

   BOB: Sure, i t ' s 1 2 3 4!

   ALICE: Great, thanks! I used i t and d e l e t e d the p r e vi o u s message.

   BOB: Did it work?

   What is the length of the longest sending chain used by Alice? By Bob? Please explain. ()

   Ans:

   - Longest sending chain by Alice: 2 (for messages 1 and 3). After sending message 1, the chain increments to 1. After deleting message 1 and sending message 3, the chain increments again to 2.
   - Longest sending chain by Bob: 1 (for message 2). Bob only sends one message in this conversation.

4. Unfortunately, in the situation above, Mallory has been monitoring their communications and finally managed to compromise Alice's phone and steal all her keys just before she sent her third message. Mallory will be unable to determine the locker combination. State and describe the relevant security property and justify why double ratchet provides this property. ()

   Ans: Double Ratchet provides Forward Secrecy (FS), ensuring that compromising Alice's keys after message 3 does not allow Mallory to decrypt the locker combination (message 1). The compromised keys only decrypt messages secured with the corresponding key ratchet, which gets updated after sending message 3.

5. The method of government surveillance is deeply flawed. Why might it not be as effective as intended? What are the major risks involved with this method? ()

Ans: The government surveillance method is flawed because the government can only decrypt messages with their public key, which is known to everyone. Anyone can encrypt messages for Alice that the government can decrypt, making the surveillance ineffective. Additionally, it leaks information about who is communicating with whom.

6. The SubtleCrypto library is able to generate signatures in various ways, including both ECDSA and RSA keys. For both the ECDSA and RSA-based signature techniques, we will compare the following aspects:

   (a) Which keys take longer to generate (timing SubtleCrypto.generateKey)?

   (b) Which signature takes longer to generate (timing SubtleCrypto.sign)?

   (c) Which signature is longer in length (length of output of SubtleCrypto.sign)?

   (d) Which signature takes longer to verify (timing SubtleCrypto.verify)?

   Note: For this question, you can choose to conjecture what the right answer might be or run the script provided in the starter code (in folder /question6code). A well justified conjecture that may not be fully correct can still receive full credit. To run the code in /question6code, navigate to the folder in your terminal and then execute the command node q6code.js. ()

Ans:

(a) Key Generation Time: ECDSA is generally faster than RSA due to shorter key lengths.

(b) Signature Generation Time: ECDSA is faster than RSA due to its simpler mathematical operations (elliptic curve operations vs modular exponentiation in RSA).

(c) Signature Length: ECDSA produces shorter signatures compared to RSA for the same level of security.

(d) Verification Time: ECDSA is generally faster than RSA due to the shorter signature length and simpler verification process.