CS255 Project 2

Neha Kunjal, nkunjal@stanford.edu
Yan Mia Min, yanmin@stanford.edu

February 2023

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)?

   In this question, we shall consider whether semantic security depends on DH ratchet or not. DH ratchet generates the key used for encryption and for the next key generation which in this problem only happens 10 times. The encryption algorithm for the messages from the key generated by DH ratchet is AES which is known to be a semantically secure cipher. Thus even if we only generated one key for the encryption of messages, our encryption would be semantically secure since we are using a semantically secure algorithm to generate the encryptions. Thus, we could have the protocol increment DH ratchet every 10 messages without compromising confidentiality.

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.

   If the DH keys never updates, that means that we always increment the chain key for both message received and message sent and we never move to another chain. This system still has forward secrecy, as every message is encrypted using an AES key derived from a hash function, which is irreversible. So if a current message key is compromised, the attacker is not able to learn the past messages because the attacker cannot reverse engineer the previous messages keys. However this system will not guarantee break-in recovery. This is because with a compromised key, the attacker can do the symmetric-key ratchet, and obtain all the following message keys that Alice/Bob will use. Thus all future messages with be known to the attacker.

3. Consider the following conversation between Alice and Bob, protected via the Double Ratchet Algorithm according to the spec:
   A: Hey Bob, can you send me the locker combo?
   A: I need to get my laptop
   B: Sure, it's 1234!
   A: Great, thanks! I used it and deleted the previous message.
   B: Did it work?
   What is the length of the longest sending chain used by Alice? By Bob? Please explain.

   The length of a sending chain is the number of consecutive message sent by one person before a message is received. Whenever an exchange of messages meaning the other party has sent a message that the original party received, a new chain will be formed as we do the DH ratchet step in the `receiveMessage` function. Thus, the length of the longest sending chain for Alice is 2 because of the first two messages sent by Alice. The length of the longest sending for Bob is 1 since Bob received messages from Alice each time a message was sent.

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.

The relevant security property that prevents Mallory from determining the locker combination is forward secrecy. The reason for this is because when Alice receives the message from Bob getting the locker combo, she decrypts it using her current key and then recomputes a new DH key for future messages and throws away the message keys needed to decrypt the message, thus starting a new chain for when she will send a message. Therefore, when Mallory steals Alice's keys she will get the new DH key that has not been used to encrypt any messages and since with a key you can't get the past keys due to the fact `HMACtoHMAC` is not irreversible (which is why this system has forward secrecy), Mallory is unable to decrypt the message getting the combination.

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?

One major risk to the government surveillance method is that if the government's secret key is leaked all messages can be read by anyone who knows the government's secret key. The reason this is true is because `vGov` is public for every message so with the government's secret key the attacker can derive all past and forward keys used to encrypt the message and thus can successfully decrypt all messages. This means that the government encryption does not have the properties of Break-in Recovery and Forward Secrecy like the communication between the users do which is a major flaw.

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, please compare:
(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)

(a) Generating an RSA key takes longer than generating an ECDSA key, as finding a large prime for RSA takes time. From the code run, RSA took 3.620s and ECDSA took 2.551ms.
(b) RSA signature takes longer to generate than ECDSA. From the code run, RSA takes 8.232ms and ECDSA takes 2.058ms.
(c) RSA has longer signatures than ECDSA. From the code run, RSA is length 512 and ECDSA is 96.
(d) ECDSA takes longer to verify than RSA, as RSA use a public key for its verification. From the code run, RSA takes 0.496ms and ECDSA takes 1.109ms.