CS255, Winter 2020
Introduction to Cryptography

Project 2 Short Answer Questions
Silvia Gong, Julian Cooper

Question 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?

Semantic security here means that an adversary eavesdropping cannot distinguish between random and ciphertext produced by our algorithm. Whether this property is provided by our implementation depends on the security properties Galois Counter Mode which is our underlying encryption algorithm.

Since GCM is semantically secure, we can argue that our algorithm is also semantically secure. Moreover, if we reduce the frequency of DH Ratchet steps, our algorithm will remain semantically secure since we are still performing the symmetric ratchet which provides GCM with new message keys each time.

However, by reducing the frequency of our DH ratchets, we do compromise Forward Secrecy since an attacker that steals our root key will be able to decrypt upto the last 10 messages.

Question 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.

Forward Secrecy: refers to an attacker not being able to read past messages given knowledge of the current root and chain keys. If we do not perform the DH ratchet, the attacker would be able to read all past communications if they possess the root key output from the last DH ratchet step!

However, if the attacker only is able to steal a chain key, then our algorithm will still provide Forward Secrecy since we update these for every message sent via our one-way HMAC function (Symmetric Chain KDF).

Break-in Recovery: refers to our users being able to recover from an attack where the adversary steals one party's key. Since inputs for the sending and receiving chains are constant, the symmetric ratchet itself does not provide break-in recovery. If we do not update the DH keys at all, then we cannot recover from an attack and therefore would not provide Break-in Recovery.

Question 3: Consider the following conversation between Alice and Bob, protected via 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.

Longest sending chain used by Alice is length 2. This is occurs when Alice sends 2 messages without response from Bob. Since Bob did not respond in between, Alice completes a second symmetric ratchet to update the chain and message keys without needing to complete a DH ratchet to update the root key.

Longest sending chain used by Bob is length 1. This is because Bob never sends more than one message in sequence without Alice responding. Therefore, Bob needs to perform a DH ratchet and creates and new sending chain every time he sends a message.

Question 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.

Forward Secrecy. This property states that compromising long term keys or current session key must not compromise past communications.

The double ratchet provides this property: output keys from the past appear random to an adversary who learns the KDF key at some point in time. This happens because each ratchet step uses a KDF which acts like a one-way PRF (deterministic and output looks random) so long as we include sufficient entropy.

Question 5: The method of government surveillance is deeply flawed. Why might it not be as effective as intended?

We interpreted this question to be about citizens' ability to evade government eavesdropping. Idea: what is the user (citizen) simply did not follow the protocol and used the wrong government public key. Without updates to the protocol, the government would likely not even know this had happened unless it tried to decrypt every message!

What are the major risks involved with this method?

The major risk we saw was around the government becoming a single source of failure. For example, if the government was hacked, and their secret key was stolen, the attacker could access and read all communciations that use the messanging service.

Question 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 technqiues, please compare:

1. Which keys take longer to generate (timing SubtleCrypto.generateKey)? RSA-based techniques (eg. RSA-PSS) are much slower for generating keys: 300ms vs 2ms
2. Which signature takes longer to generate (timing SubtleCrypto.sign)? RSA-based signatures take longer to generate: 5ms vs 1ms
3. Which signature is longer in length (length of output of SubtleCrypto.sign)? RSA signature byte length is longer: 512 vs 96 bytes
4. Which signature takes longer to verify (timing SubtleCrypto.verify)? ECDSA takes longer to verify: 0.26ms vs 0.74ms