Cryptography II

Question 1 ***Public-key encryption and digital signature*** **(10 min)**

Alice and Bob want to communicate over an insecure network using public-key cryptography. They know each other's public key.

(a) Alice receives a message: Hey Alice, it's Bob. You owe me $100. Plz send ASAP.
The message is encrypted with Alice's public key.
◇ *Question:* Can Alice be sure that this message is from Bob?

> **Solution:** No. Alice's public key is public. Anyone can encrypt a message under Alice's public key, not necessarily Bob.

(b) Bob receives a message: Hey Bob, it's Alice. I don't think I owe you $100. You owe me.
The message is digitally signed using Alice's private key.
◇ *Question:* Can Bob be sure that this message is from Alice?

◇ *Question:* How does Bob verify this message?

> **Solution:** Yes. Only Alice can create a signature under her key.
>
> Bob can verify it using Alice's public key.

(c) Alice receives a message: Hey Alice, it's Bob. Find that $100 in my online wallet, my password is xxxxxx.
The message is encrypted with Alice's public key.
Alice decrypted this and tested the password, and it was in fact Bob's.

◇ *Question:* Can an eavesdropper also figure out the password?

> **Solution:** No. The eavesdropper does not have Alice's private key, which is needed to decrypt the message.

Question 2 ***Encryption provides no integrity, signature provides no confidentiality*** **(25 min)**

Alice and Bob want to communicate with confidentiality and integrity. They have:

- Symmetric encryption.
  - Encryption: $\mathsf{Enc}(\mathsf{k}, \mathsf{m})$.
  - Decryption: $\mathsf{Dec}(\mathsf{k}, \mathsf{c})$.
- Cryptographic hash function: $\mathsf{Hash}(\mathsf{m})$.
- MAC: $\mathsf{MAC}(\mathsf{k}, \mathsf{m})$.
- Signature: $\mathsf{Sign}_{\mathsf{sk}}(\mathsf{m})$.

They share a symmetric key K and know each other's public key.

We assume these cryptographic tools do not interfere with each other when used in combination; *i.e.*, we can safely use the same key for encryption and MAC.

Alice sends to Bob
1. $\mathsf{c} = \mathsf{Hash}(\mathsf{Enc}(\mathsf{k}, \mathsf{m}))$
2. $\mathsf{c} = \mathsf{c}_1, \mathsf{c}_2$: where $\mathsf{c}_1 = \mathsf{Enc}(\mathsf{k}, \mathsf{m})$ and $\mathsf{c}_2 = \mathsf{Hash}(\mathsf{Enc}(\mathsf{k}, \mathsf{m}))$
3. $\mathsf{c} = \mathsf{c}_1, \mathsf{c}_2$: where $\mathsf{c}_1 = \mathsf{Enc}(\mathsf{k}, \mathsf{m})$ and $\mathsf{c}_2 = \mathsf{MAC}(\mathsf{k}, \mathsf{m})$
4. $\mathsf{c} = \mathsf{c}_1, \mathsf{c}_2$: where $\mathsf{c}_1 = \mathsf{Enc}(\mathsf{k}, \mathsf{m})$ and $\mathsf{c}_2 = \mathsf{MAC}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, \mathsf{m}))$
5. $\mathsf{c} = \mathsf{Sign}_{\mathsf{sk}}(\mathsf{Enc}(\mathsf{k}, \mathsf{m}))$
6. $\mathsf{c} = \mathsf{c}_1, \mathsf{c}_2$: where $\mathsf{c}_1 = \mathsf{Enc}(\mathsf{k}, \mathsf{m})$ and $\mathsf{c}_2 = \mathsf{Enc}(\mathsf{k}, \mathsf{Sign}_{\mathsf{sk}}(\mathsf{m}))$

(a) Which ones of them can Bob decrypt?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

> **Solution:** Bob cannot decrypt Scheme 1 because he cannot invert Hash. Similarly, he cannot extract the original message from the signature sent in Scheme 5.
>
> The signature does not include the message.
>
> ***In sum,*** 2-4, 6.

(b) Consider an eavesdropper Eve, who can see the communication between Alice and Bob.

Which schemes, of those decryptable in (a), also provide *confidentiality* against Eve?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

Solution: Scheme 3 does not provide confidentiality because the MAC is sent in plaintext. For the same message, the MAC is the same, thus leaky.

In sum, 2, 4, 6.

(c) Consider a man-in-the-middle Mallory, who can eavesdrop and modify the communication between Alice and Bob.

Which schemes, of those decryptable in (a), provide *integrity* against Mallory?
i.e., Bob can detect any tampering with the message?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

Solution: Scheme 2 does not provide integrity as Mallory can forge a message by sending Bob $(c', \text{Hash}(c'))$.

In sum, 3, 4, 6.

(d) Many of the schemes above are insecure against a *replay attack*.

If Alice and Bob use these schemes to send many messages, and Mallory remembers an encrypted message that Alice sent to Bob, some time later, Mallory can send the exact same encrypted message to Bob, and Bob will believe that Alice sent the message *again*.

How to modify those schemes with confidentiality & integrity to prevent replay attack?

◇ The first scheme providing confidentiality & integrity is Scheme ☐.

The modification is:

◇ The second scheme providing confidentiality & integrity is Scheme ☐.

The modification is:

Solution: Add a non-repeating nonce or a timestamp in the MAC (Scheme 4) or in the signature (Scheme 6).

In sum, In both 4 and 6, we replace message m with timestamp || m.

Question 3 ***Hashing password with salts*** **(10 min)**

When storing a password pw, a website generates a random string salt, and saves:

$$(\mathsf{salt}, \mathsf{Hash}(\mathsf{pw} \parallel \mathsf{salt}))$$

in the database, where Hash is a cryptographic hash function.

(a) If a user tries to log in with password pw′ (which may or may not be the same as pw). How does the site check if the user has the correct password?

(b) Why use a hash function Hash rather than just store pw directly?

(c) What is the purpose of the salt?

Solution:

(a) The site computes $\mathsf{Hash}(\mathsf{pw}'||\mathsf{salt})$ using the salt in the database. If this hash output equals the stored hash value, the password is correct.

(b) If the hash function is secure and the password has good entropy, even if an attacker hacks into the site, the attacker cannot figure out the passwords.

(c) Many hackers use a precomputed inverse hash table for some common passwords to reverse the hashes of common passwords.

Salts disable such tables and force the attacker to do at least dictionary attack.