Nicholas & Peyrin Summer 2021	CS 161 Computer Security	Discussion 5

Cryptography III

Question 1 ***Public-key encryption and digital signatures*** ()

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 money. 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. How many dollars do I owe you? 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 response: 10000

The message is encrypted with Alice's public key using ElGamal encryption. Alice decrypted this successfully, but suddently remembered that she only owed Bob $100.

◇ *Question:* Assume Bob would not lie. How did an attacker tamper with the message?

◇ *Question:* What could Bob have additionally sent that would've stopped this attack?

> **Solution:** The attacker multiplied c_2 by 100, or multiplied $c_1 \cdot c_1', c_2 \cdot c_2'$ where c' is a valid encryption of 100, or they encrypted an entirely new message.
>
> Bob could attach a signature to his original message.

Question 2 *Why do RSA signatures need a hash?* (min)

To generate RSA signatures, Alice first creates a standard RSA key pair: (n, e) is the RSA public key and d is the RSA private key, where n is the RSA modulus. For standard RSA signatures, we typically set e to a small prime value such as 3; for this problem, let $e = 3$.

To generate a **standard** RSA signature S on a message M, Alice computes $S = H(M)^d \bmod n$. If Bob wants to verify whether S is a valid signature on message M, he simply checks whether $S^3 = H(M) \bmod n$ holds. d is a private key known only to Alice and $(n, 3)$ is a publicly known verification key that anyone can use to check if a message was signed using Alice's private signing key.

Suppose we instead used a **simplified** scheme for RSA signatures which skips using a hash function and instead uses M directly, so the signature S on a message M is $S = M^d \bmod n$. In other words, if Alice wants to send a signed message to Bob, she will send (M, S) to Bob where $S = M^d \bmod n$ is computed using her private signing key d.

(a) With this **simplified** RSA scheme, how can Bob verify whether S is a valid signature on message M? In other words, what equation should he check, to confirm whether M was validly signed by Alice?

> **Solution:** $S^3 = M \bmod n$.

(b) Mallory learns that Alice and Bob are using the **simplified** signature scheme described above and decides to trick Bob into beliving that one of Mallory's messages is from Alice. Explain how Mallory can find an (M, S) pair such that S will be a valid signature on M.

You should assume that Mallory knows Alice's public key n, but not Alice's private key d. The message M does not have to be chosen in advance and can be gibberish.

> **Solution:** Mallory should choose some random value to be S and then compute $S^3 \bmod n$ to find the corresponding M value. This (M, S) pair will satisfy the equation in part (a).
>
> **Alternative solution:** Choose $M = 1$ and $S = 1$. This will satisfy the equation.

(c) Is the attack in part (b) possible against the **standard** RSA signature scheme (the one that includes the cryptographic hash function)? Why or why not?

> **Solution:** This attack is not possible. A hash function is one way, so the attack in part (b) won't work: we can pick a random S and cube it, but then we'd need to find some message M such that $H(M)$ is equal to this value, and that's not possible since H is one-way.

Comment: This is why the real RSA signature scheme includes a hash function: exactly to prevent the attack you've seen in this question.

Question 3 *Hashing passwords with salts* (15 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) Suppose the site doesn't use a salt and just stores $\mathsf{Hash}(\mathsf{pw})$. What attack becomes easier?

(d) Suppose the site has two candidate hash functions Hash_1 and Hash_2. Their properties are shown in the table below.

Function	**One-Way**	**Collision Resistant**
Hash_1	Yes	No
Hash_2	Yes	Yes

Which of them suffice for password hashing?

Solution:

(a) The site computes $\mathsf{Hash}(\mathsf{pw}' \parallel \mathsf{salt})$ 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) It makes inverting the hash much easier. 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 perform at least a dictionary attack for each user.

(d) Both suffice since we only need one-wayness.