CSCE 44303/54203 Homework 4

Release date: September 25, 2024

Due date: October 2, 2024

Full Grade: 80 pts

Blake Williams

Note: If you use handwriting, please write clearly. Unrecognizable writing will cause loss of points.

1. **(8 pts)** Generating a signature with RSA alone on a long message would be too slow (presumably using ciper block chaining). Suppose we could do division quickly. Would it be secure to compute an RSA signature on a long message by first finding what the message equals (taking the message as a big integer), mod ***n***, and signing that? That is, for message *m*, the sender computes *d = m mod n*, signs on d to get signature *s*, and then sends <m, s> to the receiver.
    a. **No, one of the examples on why this is a bad idea is that there is a potential that given the same n value on two messages, then the d values would be the same. If we had 2 messages m1 = 123 and m2 = 223 then n = 100. Solving for d on both of these messages, would result in 23. Therefore different messages can produce the same signature making the signature unreliable because they are no longer uniquely bound to the specific message**

2. **(8 pts)** It is common, for performance reasons, to digitally sign (e.g., with RSA) the hash of a message rather than the message itself. Why is it so important that it be difficult to find two messages with the same message digest?
    a. **Lets say Alice signs a legitimate message m1 by generating a hash of it H(m1). She then signs this hash using her private key such that the signature = Sign(H(M1)) which is publicly verifiable by using her public key. If an attacker finds a M2 such that H(M2) = H(M1) then the attacker can replace the signature = Sign(H(M2)) making it seem as if Alice signed off on M2.**

3. **(8 pts)** Prove that if H is collision resistant then so is $HMAC_k(m)=H(k|H(k|m))$. (hint: proof by contradiction)
    a. **Lets assume that H is collision resistant, but $HMAC_k(m)$ is not collision resistant. This means that there exists two distinct messages $HMAC_k(m1)= HMAC_k(m2)$. This can be rewritten as $H(k|H(k|m1)) = H(k|H(k|m2))$. Since H is deterministic the statement can be rewritten as $k|H(k|m1) = k|H(k|m2)$, and since the key k is the same for m1 and m2 the statement can be rewritten as $H(k|m1) = H(k|m2)$. This is a collision for the hash function h since k|m1 and k|m2 are two distinct messages that when hashed equal each other, contradicts the assumption that H is collision resistant. Therefore the initial assumption that $HMAC_k(m)$ is not collision resistant must be false leading to the proof that if $HMAC_k(m)$ is collision resistant if H is collision-resistant.**
4. **(10 pts)** Suppose digital signature is used to protect the integrity of the communications between Alice and Bob. Alice's public key is e_A, private key is d_A; Bob's public key is e_B, private key is d_B. Now Alice wants to send a message m to Bob. Describe the signature generation process at Alice,

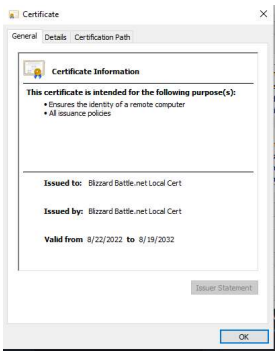
the components of the message that Alice sends to Bob, and the signature verification process at Bob.

a. **Alice**
   i. **Hash the message m with h = H(m)**
   ii. **Encrypt the hash using Alice's private key. S = Encrypt(h,d_a)**
   iii. ***Assuming Bob does not have e_A* Send {m,S, e_A} to Bob**
b. **Bob**
   i. **Hash the received message h` = H(m)**
   ii. **Use Alice Public key e_A to decrypt the Signature S**
      1. **h`` = Decrypt(S, e_A)**
   iii. **Compare h` and h`` to see if they match**
   iv. **If they match the message is valid, else the message has been altered.**

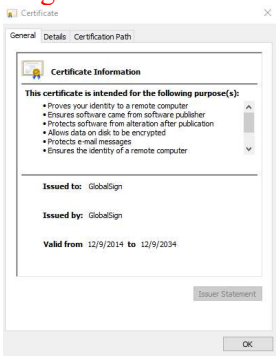
5. **(10 pts)** Suppose HMAC is used to protect the integrity of the communications between Alice and Bob. Suppose Alice and Bob already have a shared secret key k. Now Alice wants to send a message m to Bob. Describe the HMAC generation process at Alice, the components of the message that Alice sends to Bob, and the HMAC verification process at Bob.
   a. **Using HMAC(k, m) = H(k|H(k|m))**
      i. **Alice Generation**
         1. **Alice Concatenates k and m this will be Data1**
         2. **Alice computes the hash of Data 1 H(Data1) or H(K|M) this will be Hash1 (H(Data1) = Hash1)**
         3. **Alice then concatenates the secrete key k with Hash 1 (k | H(k | m)) making Data2**
         4. **Alice finally Hashes Data 2 making the final HMAC**
            a. **HMAC(k,m) = H(Data2) = H(k | H(k | m))**
      ii. **Alice Components**
         1. **The components of the message sent to Bob are {m, HMAC(k,m)}**
      iii. **Bob Verification**
         1. **Bob follows the same steps as Alice did when Generating her HMAC(k,m) using the secret key k and the received message m.**
         2. **The HMAC bob calculated will be called HMAC`(k,m)**
         3. **After calculating HMAC` Bob compares the calculated HMAC` to Alices sent HMAC.**
            a. **If HMAC == HMAC` then the message is authentic and has not been modified**
            b. **If HMAC != HMAC` then the message has been altered**

6. **(8 pts)** If public key certificates do not have the CA's signature, what will be the problem?
   a. **If the public key certificates do not have a CA signature then there is no trusted third party vouching for the authenticity of the public key. Without CA's signature anyone could generate and distribute face certificates enabling impersonation and/or man in the middle attacks.**
   b. **A public key certificate binds to an entity such as a website. The CAs signature acts as the guarantee that this binding has been verified, without CAs signature there is no way to verify the public key is bound to the appropriate entity.**

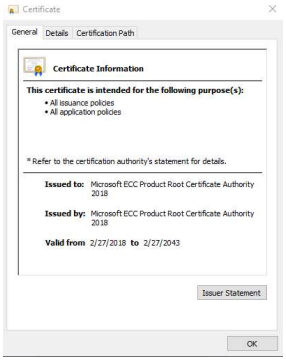
7. **(6 pts)** Find and list three trusted CAs in your web browser.
   1. Blizzard Battle.net



2. GlobalSign



3. Microsoft ECC Product Root Certificate Authority 2018



8. **(6 pts)** SHA-256 has 256-bit hash values. If you try random messages and compute their hashes, how many trials are needed until you have 0.5 or higher probability to find two messages with the same hash value?
    a. **2^{128}**

9. **(8 pts)** Mr. Random has been told to design a scheme to prevent messages from being modified by an attacker when it is being transmitted from the sender to the receiver. Suppose he designs a scheme that appends to each message m a hash of the message concatenated by the receiver's name B, i.e., H(m|B) so that the receiver can verify the hash after receiving the message. Can this solve the problem? Why or why not?
    a. **No, the hash alone does not prove the authenticity of a message. Let's say Bob has a message to send to Alice and uses Mr. Randoms scheme. Bobs message would be m | H(m | B) where B is Bobs name. Bob sends the message to Alice, however John intercepts the message in transit. Because John knows Bobs name, he is able to generate a new message m` and a new Hash H(m`|B). When The message arrives, and Alice hashes the modified message m` with Bobs name it will be the same as the**

modified Hash that was sent. Making it seem as if the message was never tampered with. Essentially a man in the middle attack.

10. **(8 pts)** Suppose Bob is offline. How can Alice leave an encrypted message to Bob using the Diffie-Hellman protocol? (Hint: Bob can publish his TB=g^b mod p in advance.)
    a. **Alice**
        i. **If Alice wants to send a message to Bob while he is offline she can first generate a random private key a. Then computes her public key with TA = g^a mod p**
        ii. **Using Bobs pubic key TB Alice computes a shared secret K = $(TB)^a$ mod p = g^{ab} mod p where a is Alices private key.**
        iii. **Alice encrypts her message using K as the key**
        iv. **Alice then sends the Encrypted message and her public key TA = g^a to Bob.**
    b. **Bob**
        i. **When Bob comes back online he uses Alices public key TA to compute the shared secret by doing K = $(TA)^b$ mod p = g^{ab} mod p**
        ii. **The result is the same shared secret K = g^{ab} that Alice used for encryption**
        iii. **Bob uses this shared secret to decrypt the ciphertext**