

April 7 & April 8, 2015

Question 1 *RSA*

(10 min)

- (a) Describe how to find a pair of public key and private key for RSA encryption system.

Solution: Find two distinct large prime numbers p and q , calculate $n = pq$, calculate $\phi(n) = (p-1)(q-1)$, and determine e, d such that $ed \equiv 1 \pmod{\phi(n)}$. public key is (e, n) , and (d, n) is private key.

- (b) What is the encryption function?

Solution: Suppose m is the message. $c \equiv m^e \pmod{n}$. c is the ciphertext.

- (c) What is the decryption function?

Solution: $m \equiv c^d \pmod{n}$.

- (d) How to use RSA as a digital signature scheme?

Solution: Signer has the private key and calculates $S(m) \equiv H(m)^d \pmod{n}$. Signer sends $(m, S(m))$ to the verifier who knows the public key (e, n) . The verifier determines whether $H(m)$ equals to $S(m)^e \pmod{n}$. If yes, the verifier knows the message was signed by the signer. H is a cryptographic hash function that both the signer and the verifier know. Please see the homework 3 for why we need a hash function.

Question 2 *Public Key Basics*

(10 min)

Assume Alice and Bob are trying to communicate over an insecure network with public key encryption. Both Alice and Bob have published their public keys on their website. (You can assume they already know each other's correct public key)

- (a) Alice receives a message: *Hey Alice, it's Bob. You owe me 100 bucks. Plz send ASAP.* The message is encrypted with Alice's public key. Can she trust it?

Solution: No she cannot. As the name implies, a public key is generally publicly known. Anyone can go to Alice's website and use her public key to encrypt whatever they want.

- (b) Bob receives a message: *Hey Bob, that last message you sent me was sketchy. I don't think I owe you 100 bucks. You owe me.* The message is digitally signed using Alice's private key. Can he trust it was from Alice? How does he verify this message?

Solution: Yes. As long as Alice has kept her private secure, she is the only one who could construct such a message. Bob would verify it using Alice's public key.

- (c) Alice receives a message: *Hey Alice, I know things have been rough these last two messages. But I trust you now. Here is my password: hel1xfoss1l.* The message is encrypted with Alice's public key. Alice decrypted this and tested the password, and it was in fact Bob's! Can an eavesdropper figure out the password?

Solution: No. An eavesdropper would not have Alice's private key, which is needed to decrypt the message.

Question 3 *Confidentiality and Integrity* (10 min)

Alice and Bob want to communicate with both confidentiality and integrity. They have access to a symmetric key encryption function E and corresponding decryption function D and have already securely shared the key K . They also have a cryptographic hash function H . Recall that H is not keyed, so anyone can compute $H(x)$ given x . However, due to the constraints of their computers they only have enough computing power to compute H once and E or D once per message (whether sent or received). You may assume that H and E do not interfere with each other when used in combination – for example, if you compute $H(E(M))$, the message M will be confidential because E guarantees it, and the computation of H makes no difference. To send message M to Bob, Alice has the option to use any of the five schemes listed on the next page.¹

- a) Consider the threat model in which Eve is only able to eavesdrop and Alice and Bob are using the key K for the first time and will use it once and only once. In addition, Eve has no partial information about the message that will be sent. For each scheme, determine whether or not the scheme allows Bob to decrypt messages from Alice. (Don't worry about integrity yet.)

¹ Alice could also send the plaintext, but that is not a good idea.

Solution: 2. $E(M)$, $H(E(M))$ and 5. $E(M)$, $H(M)$.

The other options cannot be decrypted, because Bob cannot invert the hash function H .

In this question, we wanted you to consider passive attacks on confidentiality - that is, whether or not Eve can recover M if she eavesdrops on the ciphertext sent by Alice. We did not consider active attacks such as chosen-plaintext or chosen-ciphertext attacks.

For option 2, we only work with the encrypted version of the message $E(M)$, which guarantees confidentiality. However, if we allow active attacks on confidentiality, then this may not be secure, as it is possible that an active attack would allow an attacker to recover M from $E(M)$. (For example, CBC mode is vulnerable to an active attack called a padding oracle attack.)

Option 5 is a little trickier. Certainly as before Eve cannot recover M from $E(M)$. However, Eve also knows $H(M)$. Since H is a cryptographic hash function, Eve cannot invert H to get M . However, if M is low-entropy (there are only a few likely possibilities of M) then Eve could just compute $H(M')$ for all of these M' and see which ones match the value of $H(M)$ sent by Alice. The question specified that Eve has no partial information about the message - this means that the message M has high entropy (as long as it is sufficiently long) and so the attack will not work. So in this threat model option 5 does have confidentiality.

b) Out of the schemes you chose in part a), determine which of the schemes also provide integrity; that is, Bob will be able to detect any tampering with the message that Alice sends. Describe what Bob has to do in order to decrypt the message and make sure that it came from Alice. (Remember, Bob can only use D once and H once.) If you think it will not work, explain why. For any that do not work, what is the vulnerability?

Alice Sends to Bob	Confidentiality	Integrity	Decrypt Steps
1. $H(E(M))$			
2. $E(M)$, $H(E(M))$			
3. $E(H(M))$			
4. $E(H(M))$, $H(M)$			
5. $E(M)$, $H(M)$			

Solution: Only option 5 provides integrity, for high entropy messages. The idea is that $E(M)$ provides confidentiality, and $H(M)$ cannot be inverted, so the attacker cannot find M . Integrity is guaranteed because since K is used for the first time, Eve

cannot know a plaintext-ciphertext pair, and so if she replaces $E(M)$ with some C , she cannot know $D(C)$, and so she cannot compute $H(D(C))$. She might be able to make small modifications to M to create $E(M')$ by messing with $E(M)$, but since she does not know M she still cannot compute $H(M')$.

However, if we expand our threat model to also allow Eve to have partial information about the message, then we lose the guarantee of integrity. For example, if Eve knows what the message M is (so her “partial information” is knowledge of the message itself), then she can modify $E(M)$ to get $E(M')$ where she knows M' , and then she can compute $H(M')$. Why can Eve modify $E(M)$ to get $E(M')$? Because the encryption algorithm does not provide integrity. In fact, on HW3 you will show how to modify $E(M)$ to get $E(M')$ for some encryption schemes.

Option 2 is bad because Eve could replace the message with $(x, H(x))$ for a random x , and Alice wouldn't realize that the message was not from Bob. Bob will see a gibberish message $D(x)$, but he will still believe it is from Alice, which is a failure of integrity.

It doesn't make sense to talk about the integrity of the other options - since Bob can't read the message, it doesn't matter how Eve tampers with the ciphertext.

Alice Sends to Bob	Confidentiality	Integrity	Decrypt Steps
1. $H(E(M))$	yes	??	no H^{-1}
2. $E(M), H(E(M))$	yes	no	$D(C)$, can't check integrity
3. $E(H(M))$	yes	??	no H^{-1}
4. $E(H(M)), H(M)$	yes	??	no H^{-1}
5. $E(M), H(M)$	yes	yes	$D(C), H(D(C)) = 'H(M)'$

c) If Alice and Bob use these schemes to send many messages, the schemes become vulnerable to a replay attack. In a replay attack, Eve remembers a message that Alice sent to Bob, and some time later sends the exact same message to Bob, and Bob will believe that Alice sent the message. How might Alice and Bob redesign the scheme to prevent or detect replay attacks?

Solution:

Send $E(M), H(M)$, but make sure that M has a unique identifier that both Alice and Bob know (similar to TCP sequence numbers).

This prevents the replay attack because Bob will realize that he is getting the same message twice and will throw away the second one.

It is important that the identifier is inside the encryption function E and the hash function H , because the identifier needs to have integrity (otherwise Eve could simply

change the identifier to what it needs to be and then send the same message).

Question 4 *Diffie-Hellman Basics* (10 min)

Recall that in a Diffie-Hellman key exchange, there are values a , b , g and p . Alice computes $g^a \bmod p$ and Bob computes $g^b \bmod p$.

- (a) Which of these values are publicly known and which must be kept private?

Solution:

g and p are publicly known. Implementations of Diffie-Hellman often have carefully picked values of g and p which are known to everyone. Alice and Bob must keep a and b secret respectively.

- (b) Assume Eve has tapped into the network between Alice and Bob. Eve can only view the traffic, she cannot change it. Alice and Bob perform the Diffie-Hellman key exchange and have agreed on a shared symmetric key K . However, Bob accidentally sent his b over the network in plain text. If Eve viewed all traffic since the beginning of the exchange, can she figure out what K is?

Solution:

Yes, this will be very easy for Eve. Assuming Eve realizes that Bob sent b in plain text, she can use the value $A = g^a \bmod p$ which Alice sent to calculate $K = A^b \bmod p$.

- (c) Assume Mallory has tapped into the network but has managed to not only view the traffic but also intercept and modify it. Alice and Bob perform Diffie-Hellman to agree on a shared symmetric key K . After the exchange, Bob gets the feeling something went wrong and calls Alice. He compares his value of K to Alice's and realizes that they are different. Explain how Mallory could know Bob's K_b and Alice's K_a and how she could use this secretly rewrite traffic between them.

Solution:

Mallory is performing a man-in-the-middle attack on Alice and Bob. One way to think about this is that Mallory pretends to be Bob when she talks to Alice, and Mallory also pretends to be Alice when she talks to Bob. In this way, both Alice and Bob are unknowingly talking to Mallory. Mallory can then decrypt/encrypt the traffic in both directions and modify it however she wishes to.

More technically, when Alice sends $A = g^a \bmod p$ to Bob, Mallory intercepts this (preventing it from going to Bob), and sends back to Alice: $M = g^c \bmod p$. Now when Alice sends a message to Bob, she uses $K_{bad} = M^a \bmod p$ which

Mallory knows as $K_{bad} = A^c \text{ mod } p$. Mallory can then decrypt all messages sent from Alice. She can also send messages to Alice which Alice thinks are from Bob. Mallory then does the same trick to Bob, and she has finished her man-in-the-middle attack.

Question 5 *TLS*

(10 min)

- (a) In TLS, what security properties are achieved, and what components of the TLS protocol enable these properties?

Solution: Confidentiality - communication is encrypted using session keys. Integrity - TLS uses MACs to prevent message tampering. One-way authentication - Signed certificates provide a means of authenticating the server.

- (b) Recall that in practice, TLS as used on the web typically only provides one-way authentication – that is, when communicating securely over the web, only the server is required to authenticate themselves, and not the client. Why is TLS usually used this way?

Solution: One reason is that it is rather inconvenient for the average user to have to set up their own client certificate. Websites would not want to block users who haven't set up a client certificate. Another reason is that in many cases, the server doesn't care who is connecting to it, since it's providing a service over the web that anyone can use. A user certainly cares that they are connecting to the correct server; a server likely expects connections from many clients, so it may not need or want to authenticate them all.

- (c) How else might a web server authenticate a user? (if the user is not authenticated by TLS)

Solution: Servers often authenticate the user by requiring them to establish a username and password with the site. That way, once a secure TLS connection has been established, the server knows which of its users it is communicating with.

A final note: do not hesitate to ask for help! Our office hours exist to help you. Please visit us if you have any questions or doubts about the material.