Overview: Question 1 [7 Marks]
These are short-answer questions. Please answer in 2-4 lines and not in paragraphs. ***Explain your reasoning behind the answer.*** *Yes/no answers will not fetch you marks.*

a) If P=NP, one-time pads will not be secure any more. True or False. Justify.

They are secure independent of P=NP since a given ciphertext can match many valid plain text. As was mentioned in class these are unconditionally secure.

b) Ceaser cipher (monoalphabetic) is vulnerable to chosen plain-text attack. True or false? Justify.

Yes, for a given plain text and the corresponding cipher, it is easy to figure out the transformation.

c) In RSA, for n=35 and e=7, what is d?

*N=35 = 7 * 5; phi(N) = 6 * 4 = 24; e = 7*
ed mod phi(N) = 1
*7 * d mod 24 = 1*
d = 7

d) In which cipher block mode is decryption faster than encryption. Why?

CBC or CFB since decryption can happen in parallel (for other modes, both take the same time and/or can happen in parallel)

e) In a system with N entities, what is the minimum number of keys required if communication among these entities must be kept secret from intruders *apart* from these N entities? Which type/model of cryptography achieves this?

1 key would suffice which is shared by all N entities. This is based on share key crypto.

f) In SSL, what is the advantage of session resume over creating a new session?

SSL session resume reuses the pre-master key and avoids the cost of decryption.

g) In class, we saw how to authenticate the server to the client in SSL. How can mutual authentication happen within the same framework?

Mutual authentication can happen where the client can send its password to the server using the already established secure channel.

Question 2: Block Cipher [3 Marks]

Let $m_1, m_2, \ldots, m_t$ be a sequence of t plaintext blocks. Hacker Hakim is bored with the modes he saw in class and wants to invent one of his own. He defines a sequence of $t + 2$ ciphertext blocks $c_0, c_1, c_2, \ldots, c_t, c_{t+1}$ which satisfies the equation $c_i = E_k(c_{i-1} \oplus m_i \oplus c_{i+1})$, for $i = 1, \ldots, t$.

a) Describe how to reconstruct $m_1, \ldots, m_t$ given $c_0, \ldots, c_{t+1}$.

For i = 1, . . ., t, we have the condition
ci = Ek(ci−1 ⊕ mi ⊕ ci+1)
Applying decryption function to both sides
Dk(ci) = Dk(Ek(ci−1 ⊕ mi ⊕ ci+1))
= ci−1 ⊕ mi ⊕ ci+1
Swapping the two sides gives
mi = Dk(ci) ⊕ ci−1 ⊕ ci+1

b) Hakim was having trouble figuring out how to compute the c_i's because of the circular dependencies. Please help him by showing how to compute c_{i+1} from c_{i-1}, c_i, and m_i, where $1 \leq i \leq t$. How should he choose $c0$ and $c1$?

From above, Dk(ci) = Dk(Ek(ci−1 ⊕ mi ⊕ ci+1))
= ci−1 ⊕ mi ⊕ ci+1

Swapping the two sides gives
$ci+1 = Dk(ci) \oplus ci-1 \oplus mi$, where $1 \leq i \leq t$.

In order to get started, we set $c0$ and $c1$ to some fixed initialization vectors.

Question 3: Cryptography Building Blocks [6 Marks]
Suppose A wants to send a message M to B. A wants to ensure integrity and authenticity (not confidentiality) of the message. Which of the following options meet this objective? Justify. You can ignore freshness concerns (i.e. replay attacks). The terminology is same as followed in class.

a) A sends $E_{B,pu}$ (M) to B

No. Any one can send such a message using B's public key.

b) A sends { M, $E_{A,pr}$(M) } to B

Yes. The fact that A signed the message implies that it indeed came from A and no one modified it. If it were modified, the sign will not match.

c) A sends { M, $E_{B,pu}(k)$, $MAC_k(M)$ } to B; k is a newly chosen symmetric key

No. Anyone could send such a message. Further, the message M could be modified and new MAC calculated with a new key.

d) A sends { M, $E_{B,pu}(k)$, $MAC_k(M)$, $E_{A,pr}(k)$ } to B; k is a newly chosen symmetric key

No. Once B receives these messages, he can send forged messages to C since he has A's signature on k. Also anyone can recover k from the signature, modify M and reconstruct the MAC.

Suppose now A wants to ensure confidentiality in addition to integrity and authenticity of the message. Which of the following options meet this objective? You can ignore freshness concerns (i.e. replay attacks).

a) $E_{B,pu}(M)$, $E_{A,pr}(B,pu)$

No. While it provides confidentiality, this does not provide message authentication/integrity. One can change message and pass it to B using its public key.

b) $E_{B,pu}(M)$, $E_{A,pr}(M)$

No. Does not provide confidentiality. M can be recovered from the signature part using A's public key. Provides integrity/authenticity given the signature.

c) $E_{B,pu}(M, E_{A,pr}(M))$

Yes. This works. The fact that is it signed and then encrypted by B's public key means it achieves all three.

d) $E_{B,pu}(k)$, $E_k(M)$, $E_{B,pu}(E_{A,pr}(k))$, k is a newly chosen symmetric key

No. It provides confidentiality. Does not provide integrity since message has no integrity check. Some one could flip the bits of the message and no one can detect it.

Question 4: Passwords [4 Marks]

LinkedIn experienced a major data breach in June 2012 where 6.5 million email-addresses and passwords were compromised. Unsalted password hashes were supposedly the culprit.
In unsalted version, the database maintained records containing two fields [user-email, hash(password)]. In salted version, it maintains three fields [user-email, salt, hash(password|salt)].

a) Suppose an attacker got hold of the database records and his main goal is to break <u>*your*</u> password via a dictionary attack. Does the lack of salting in LinkedIn's scheme make this goal substantially easier? Justify. A dictionary attack is based on trying all the strings in

a pre-arranged listing, typically popular passwords, often derived from a list of words such as in a dictionary.

No. For a given user, since the salt is known, the time taken to create the hashes of the dictionary is pretty much the same in both schemes.

b) Suppose the attacker's goal is to break at least half of the passwords in the database via a dictionary attack. Does the lack of salting in this scheme make this goal substantially easier? Justify.

Yes. Without salting one dictionary of hashes can be used across all the users. With salting, one has to create a different dictionary with each salt.

c) Suppose you are contacted by the attacker and given a set of password hashes (that's it, no user-email, no salt). Assuming the hash function is known, is there a measurement you could make in order to infer if the hashes are likely salted or not? Explain how.

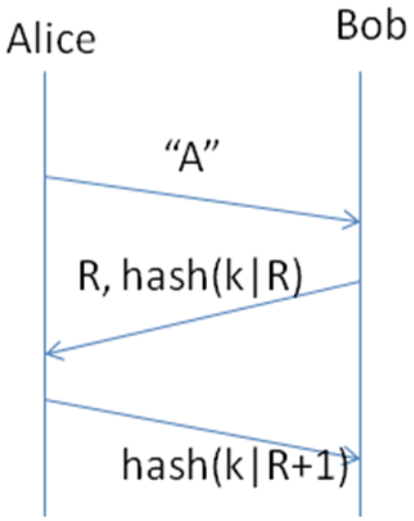
Some passwords are more popular than others and in a list, they can occur with probability say 0.1%. If the list is unsalted, you should see a similar percentage of similarity. If salted, the percentage will be lot lower and one can't find any pattern (will look like a truly random list).

d) It turns out that that roughly 20% of LinkedIn users with Yahoo Mail e-mail addresses used the same password at LinkedIn as Yahoo. You learn that, unlike LinkedIn, Yahoo salts its passwords. Should Yahoo be concerned about the LinkedIn breach or not?

Yes. Once the linkedin password is obtained, the attacker can use it to access Yahoo account.

Question 5: Authentication [4 Marks]

Alice and Bob wish to achieve mutual authentication based on symmetric cryptography using the the below protocol. What are the possible flaws in this protocol? Also propose how to fix the possible flaws with minimal modifications to the protocol. Justify how the modification fixes the flaw.

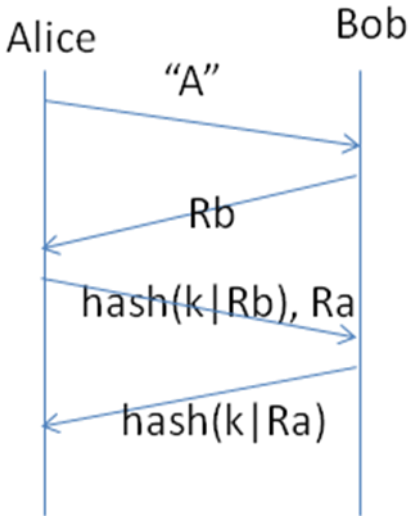


(More than one attack possible).

Mallory can record the above messages. Later, when A tries to contact B, it can pretend to be B and replay the 2nd message. Alice would think it is Bob since if she were to take the shared key k , append R and calculate the hash, it will match the received hash value.

(or) Mallory can act as a MIM and just relay the messages. Alice will think it is talking with Bob, while Bob will think it is talking with Alice, when in fact both are talking with Mallory.

Solution: More than one possible



Question 6: More Authentication [6 Marks]

Alice and Bob wish to exchange messages with each other (bi-directional traffic) and have agreed upon a sequence $K_1, K_2, \ldots$ of shared secret keys. Each key K_i consists of a pair (a_i,b_i) so that the MAC of message M computed with key K_i is $MAC(K_i,M) = a_iM + b_i \pmod p$, where p is a publicly known large prime. Given the nature of the function, they know that each K_i should not be used to MAC more than one message.

Alice and Bob agree on the following protocol for managing their use of keys and for exchanging messages. Both parties keep track of the index t that was most recently used by either themselves or their counterparty. Thus, when Alice sends a message M, she increments her t by one and then MAC's M with key K_t. The message she sends has the format: ("Alice", "Bob", M, t, MAC (K_t, M)).

When Bob receives this message, he checks that the MAC is correctly computed for the given value of t. If it is not correct for that value of t, he ignores the message altogether. If the MAC is correct for that value of t, Bob accepts the message and sets his local value of t to be equal to the t of the received message. The same procedures are followed symmetrically (with roles switched) when Bob sends to Alice. Note, each party only maintains a single local variable t.

Marks will be based on clarity and thoroughness of the solution. So, be clear in your head before writing down the answer. Be precise and to the point.

a) Why should K_i not be used to MAC more than one message?

If Ki is used twice, knowing M and the MAC (which is sent on the channel), one can solve the equations to obtain ai and bi and hence break the key

b) Identify as many vulnerabilities as you can in the above protocol design. You may assume that the messages sent on the channel can be controlled by an adversary i.e it can insert, delete, modify, or replay messages on the channel.

Adversary delaying/dropping messages to Bob can cause Bob to use a lower value of t whose corresponding key Alice already used.

Adversary could replay a very old message causing B to set t to a low value ang again causing it to reuse keys.

Adversary could garble the message with same consequences as above.

There are synchronization issues also, when both A and B send messages at the same time using the same value t.

Once the adversary has computed Ki. The adversary can send any message to both Alice and Bob and cause them to set their t value to i. The adversary can then modify any message sent and still produce a valid MAC. Let's say Alice sends a message ("Alice", "Bob", M, i + 1,MAC(Ki+1, M)). The adversary can modify the message to("Alice", "Bob", M', i,MAC(Ki, M')). The adversary has changed the message M to M' and the t value from i + 1 to i but he/she is still able to produce a valid MAC.

c) Propose a revised protocol that doesn't suffer from these problems.

A revised protocol could be the following: Alice and Bob agree on the sequence of shared secret keys, K1, K2, ... The keys are generated as in the previous protocol but now Alice uses the odd number keys to compute the MAC of her messages while Bob uses the even number keys. This guarantees that Alice and Bob will never use the same key to compute a MAC so as long as one of them does not reuse the same key the adversary will not be able to compute any secret key.

When Alice receives a message, she checks that the value of t is greater than the one she has locally and if so she checks if the MAC has the correct value and then sets her local value of t to match the message. If t is by more than 2 greater than her local t then either t was modified by the adversary or some of Bob's previous messages have been deleted, either by the adversary or by an error in the transmission. If the adversary has simply modified t, then the MAC code should not be valid for the given Kt and thus the message can be disregarded. Alice can accept the message if the MAC is valid and she can set her value of t to match the one in the message. She will know, however, that some messages were either deleted or lost in the transmission and she can notify Bob about it.