CS 166 ASSIGNMENT 5 SOLUTION

OVERVIEW

Marks on question:

- No mark: 1-part short answer question (the key point(s) for explanation is underlined)
- !: 1 part auto-graded question (answer in **bold**)
- +: n-parts question: the key point(s) for explanation is underlined.

Basic rules of grading (see grader's comment for more details):

- Those 0.5pt sub-questions are all-or-nothing.
- For 1 pt yes/no question, wrong answer or answer without explanation will result in a 0.
- For whether message proves identity or not...
  - If the answer is yes, 0.5 pt for explaining why only XXX can get the correct response and 0.5 pt for explaining how the other party can verify the response
  - If the answer is no, partial credits (0.5) if is on the right track but didn't include all key points.
- You will get 0.5 (on the whole question, not each sub-question) as long as you wrote something.

WEEK 10 - CONCEPUTAL PROTOCOLS (Q1 - Q10)

!Q1: Match the term with the descriptions. (L18 P8 - 9 & L19 P15)

I. "Eavesdropping" (observe) conversations between Alice & Bob: **Passive attack**;
II. Modify the conversation between Alice & Bob (without them knowing): **Active attack**;
III. A scheme that uses nonces to authenticate that prevents replay attacks: **Challenge-response**;
IV. One of the ways that can achieve prefect forward secrecy: **Ephemeral Diffie-Hellman**

!Q2: What is the purpose of Perfect Forward Secrecy? (L19 P14)

To ensure that it's impossible to decrypt a secret after the conversion is done.

Q3: Why sending the name(s)? (L18 & L19)

Since the other party needs to know which shared key or whose public key to use for decryption/encryption.

Q4: Using timestamp pros & cons (L19 P11)

I. Pros: T is public so no need to send a separate message for nonce, which can reduce the number of messages needed.

II. Cons: time skew is needed for syncing/delays, which increases the chance of replay attack.

+Q5: Authentication by symmetric key with nonces -1 (L18 P12 - 13)

```
Protocol:
1) Alice -> Bob: "I'm Alice", R
2) Bob -> Alice: E(R, K_AB)
3) Alice -> Bob: E(R + 1, K_AB)
```

5.1) If not consider any attacks...

I. No, message 1 doesn't show Alice is Alice. R is just a random number, so anyone can send it.
II. Yes, message 3 shows Alice is Alice. Since K_{AB} is known only to Alice & Bob, being able to encrypt R + 1 using K_{AB} shows Alice is Alice and Bob knows R (after getting 1st message) to verify if E(R + 1, K_{AB}) is correct.
III. Yes, message 2 shows Bob is Bob. Since K_{AB} is known only to Alice & Bob, being able to encrypt R using K_{AB} shows Bob is Bob and Alice knows R (R is generated by Alice) to verify if E(R, K_{AB}) is correct.

5.2) Yes, Trudy can do a reflection attack since the protocol asks Alice & Bob to prove identity in the "same" way - encrypting a message using K_{AB}. That is, Trudy can get the 3rd message by opening a 2nd session and sending R + 1 to Bob as "Alice", then Bob will reply her E(R + 1, K_{AB}), which is what Trudy needs.

+Q6: Authentication by symmetric key with nonces - 2 (L18 P12 - 13)

Protocol:
1) Alice -> Bob: "I'm Alice", E(R, K_{AB})
2) Bob -> Alice: R + 1
3) Alice -> Bob: E(R + 1, K_{AB})

6.1) If not consider any attacks...

I. No, message 1 doesn't show Alice is Alice. Bob didn't know R before so he can't verify if E(R, K_{AB}) is correct or not.
II. Yes, message 3 shows Alice is Alice. Since K_{AB} is known only to Alice & Bob, being able to encrypt R + 1 using K_{AB} shows Alice is Alice and Bob now knows R (after getting 1st message) to verify if E(R + 1, K_{AB}) is correct.
III. Yes, message 2 shows Bob is Bob. Since K_{AB} is known only to Alice & Bob, being able to decrypt E(R, K_{AB}) using K_{AB} to get R shows Bob is Bob and Alice knows R (R is generated by Alice) to verify if R + 1 is correct.

6.2) No, a reflection attack is not possible. Alice & Bob are doing "different" things to prove their identities: Bob proves his identity by decrypting a message while Alice proves her identity by encrypting a message.

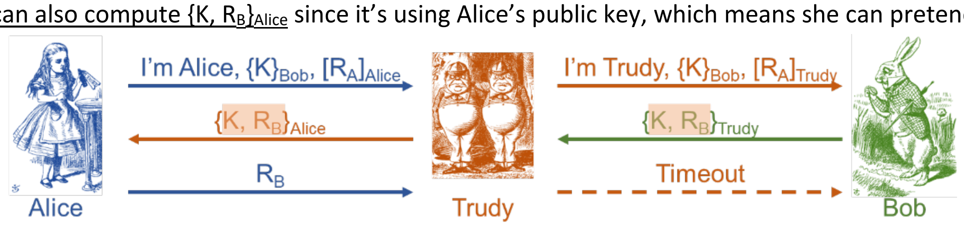
+Q7: Authentication by public key with nonces (L19 P6 - 9)

Protocol:
1) Alice -> Bob: "I'm Alice", $\{K\}_{Bob}$, $[R_A]_{Alice}$
2) Bob -> Alice: $\{K, R_B\}_{Alice}$
3) Alice -> Bob: R_B

7.1) If NOT consider any attacks...

I. No, message 1 doesn't show Alice is Alice. Everyone can compute $\{K\}_{Bob}$ using Bob's public key. Though only Alice can compute $[R_A]_{Alice}$, Bob didn't know R_A before so he can't verify if R_A is correct or not after decrypting $[R_A]_{Alice}$.
II. Yes, message 3 shows Alice is Alice. R_B is encrypted using Alice's public key so only Alice can decrypt it by her private key to get R_B. And Bob knows R_B to verify if it is correct.
III. Yes, message 2 shows Bob is Bob. K is encrypted using Bob's public key so only Bob can decrypt it by his private key to get K. And Alice knows K to verify if it is correct after decrypting $\{K, R_B\}_{Alice}$.

7.2) No, K is not exchanged securely since MiM attack is possible. Trudy can be in the middle forwarding the message from Alice to Bob but as Trudy, then Bob will send her $\{K, R_B\}_{Trudy}$ which Trudy can decrypt to get K, R_B (i.e., K is exposed!) And Trudy can also compute $\{K, R_B\}_{Alice}$ since it's using Alice's public key, which means she can pretend to be Bob.



7.3) Yes, answer to II in 7.1 will change if message 2 is changed to $\{K\}_{Alice}$, $[R_B]_{Bob}$. Everyone can decrypt $[R_B]_{Bob}$ to get R_B using Bob's public key, so knowing R_B cannot prove Alice is Alice in this case.
Note that answer to III will not change since knowing K shows Bob is Bob ($[R_B]_{Bob}$ doesn't show Bob is Bob though since Alice cannot verify it!)

+Q8: Authentication by public key with timestamp (L19 P12 - 13)

```
Protocol:
1) Alice -> Bob: {"I'm Alice", [T, K]Alice}Bob
2) Bob -> Alice: [T + 1]Bob
```

8.1)

I. Yes, message 1 shows Alice is Alice. Only Alice can compute $[T, K]_{Alice}$ and Bob knows T (T is public!) so he can verify if T is correct or not after decrypting $[T, K]_{Alice}$ using Alice's public key.

II. Yes, message 2 shows Bob is Bob. Only Bob can compute $[T + 1]_{Bob}$ and Alice knows T +1 (T is public!) so she can verify if T + 1 is correct or not after decrypting $[T + 1]_{Bob}$ using Bob's public key.

III. Yes, K is exchanged securely. K only appeared in the first message which is encrypted using Bob's public key, so only Bob can decrypt it to get K using his private key. And since the second message doesn't include K, Trudy can't get it by opening another session.

8.2) Yes, answer to I in 8.1 will change if message 1 is changed to {"I'm Alice", T, $[K]_{Alice}\}_{Bob}$. Though only Alice can compute $[K]_{Alice}$, Bob didn't know K before so he can't verify if K is correct or not after decrypting $[K]_{Alice}$ using Alice's public key.

+Q9: Authentication by public key with timestamp and Diffie-Hellman (L19 P12 - 15)

```
Protocol:
1) Alice -> Bob: "I'm Alice", [T]Alice, g^a mod p
2) Bob -> Alice: {T}Alice, g^b mod p
```

9.1) If NOT consider any attacks...

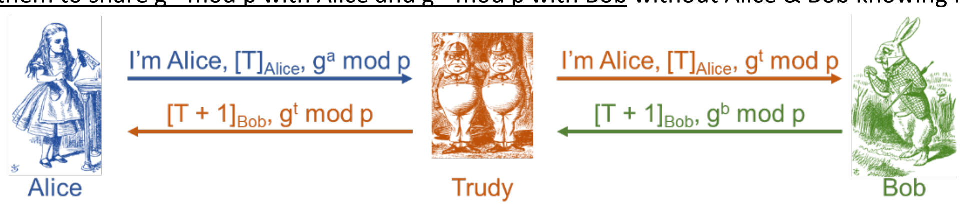
I. Yes, message 1 shows Alice is Alice. Only Alice can compute $[T]_{Alice}$ and Bob knows T (T is public!) so he can verify if T is correct or not after decrypting $[T]_{Alice}$ using Alice's public key.

II. No, message 2 does not show Bob is Bob. Everyone can compute $\{T\}_{Alice}$ using Alice's public key, and Alice didn't know g^b mod p beforehand to verify if it's correct

III. Yes, if Alice and Bob use Ephemeral Diffie Hellman, i.e., they "throw" away a & b after session is done.

9.2) Consider an MiM attack...

I. Yes, MiM will succeed. Bob didn't know the "correct" g^a mod p and Alice didn't know the "correct" g^b mod p and these two messages are sent publicly, so if Trudy can act within time skew, she can be in the middle sending g^t mod p to each of them to share g^{at} mod p with Alice and g^{bt} mod p with Bob without Alice & Bob knowing it.



II. There are different ways to prevent such an MiM attack. For example, signing the g^a mod p and g^b mod p with T so Trudy can't change them (recall that digital signature provides integrity). Or, encrypting the g^a mod p and g^b mod p - though Trudy can change them, she can't compute g^{at} mod p or g^{bt} mod p without knowing the g^a mod p or g^b mod p.

+Q10: Authentication by public key and symmetric key (L18 P11 & L19 P6)

```
Protocol:
1) Alice -> Bob: "I'm Alice", {S}Bob, E(CONST1, K) // K = h(S)
2) Bob -> Alice: E(CONST2, K)
```

10.1) If NOT consider any attacks...

I. No, message 1 doesn't show Alice is Alice. Everyone can compute $\{S\}_{Bob}$ using Bob's public key. K = h(S) and S is generated by the sender of the message, so of course the sender knows K and can compute E(CONST1, K). (CONST1 is a public constant known to everyone).

II. Yes, message 2 shows Bob is Bob. Only Bob can decrypt $\{S\}_{Bob}$ using his private key to get S and then compute K = h(S). So being able to encrypt CONST2 using K shows Bob is Bob. And Alice knows K to verify if E(CONST2, K) is correct.

III. Yes, since K is used to encrypt messages (not included in the messages) and knowing the ciphertext won't reveal the key used. And S only appears in the first message (encrypted) so Trudy can't open another session to get S.

10.2) Consider a passive attack (Trudy observes the messages)...

I. Yes, Trudy can figure out K if S only contains 4 digits since she can do a brute-force attack to try all the possible hashes. There are only 10^4 = 10,000 possibilities, won't take too long to compute.

II. No, Trudy can't figure out K if S is a 256-bit key. If doing a brute-force attack, there are 2^{256} possibilities, which will take too long time to compute.

WEEK 11 ~ 12 - REAL-LIFE & NETWORK PROTOCOLS (Q10 - Q20)

Q11: Data transfer over network (L22 P10 - 11)

Down-Up-Down-Up on the protocol stack: From sender's application layer, down to the physical layer, then up to the network layer where routing info lives, after that, down to the physical layer again and finally up to the destination's application layer. When data goes down, each layer adds header information; when data goes up, the headers are stripped off layer by layer.

Q12: Network Protocols (L22)

You can pick any network protocol we covered, for example:

I. HTTP (Hyper Text Transfer Protocol)

II. Application layer

III. Used when browsing a website

+Q13: SSH (L20 P9 – P11)

See the protocol in the lecture notes/question.

13.1) If not consider any attacks

I. Yes, message 4 proves Bob is Bob. Only Bob can compute S_B using his private key, and Alice knows H to verify.

II. Yes, message 5 proves Alice is Alice. Only Alice can compute S_A using her private key, and Bob knows H to verify.

13.2) Change certificate to password

I. No, encryption is not necessary if using certificate since the certificate & signature can be public.

II. Yes, encryption is necessary if using password since the password should be kept secret.

III. No, Trudy can't. The MiM attack will result in different H's for Alice & Bob since H is based on the Diffie-Hellman values and the result shared key. That is, Alice will get H_A, but Bob will get and sign H_B. Therefore, Alice will refuse to authenticate Bob after she checked the 4th message, and she will not send the 5th message. That is, even Trudy knows the K, she can't get the password since Trudy can't even get the 5th message.

+Q14: SSL (L20 P13 - 15)

See the protocol in the lecture notes/question.

14.1) If not consider any attacks

I. No, message 3 doesn't proves Alice is Alice. Everyone can compute $\{S\}_{Bob}$ using Bob's public key. And $K = h(S, R_A, R_B)$ where S is generated by the sender of the message, and R_A & R_B are public, so of course the sender knows K and can compute E(h(msgs, CLNT, K), K) where msgs, SRVR are all public information.

II. Yes, message 4 proves Bob is Bob. Only Bob can decrypt $\{S\}_{Bob}$ using his private key to get S and then compute $K = h(S, R_A, R_B)$ So being able to compute h(msgs, SRVR, K) shows Bob is Bob. And Alice knows K (and everything else) to verify if the response is correct.

III. 1) Encryption in the 3rd message is not necessary, since S is already encrypted, and hash can already hide K.
2) R_A and R_B are not necessary since S is the challenge. Being able to decrypt $\{S\}_{Bob}$ to get S (and compute K) already shows Bob is Bob. Indeed, S is a nonce since Alice will choose different S randomly for every session.

14.2) Why MiM on SSL would fail?

I. It would fail since no way Trudy can fake a certificate.

II. Alice's browser will warn her that the certificate is not valid, but Alice can choose to ignore the warning and continue...then the attack can succeed.

+Q15: Kerberos (L20 P17 - 21)

See the protocol in the lecture notes/question.

15.1) Consider the "getting TGT (Kerberized login)" step.

I. TGT is sent to the user so KDC can remain stateless. KDC doesn't need to maintain a database for users and their session key, since the user will present his/her TGT when getting regular tickets. KDC can decrypt the TGT and get all the information needed.

II. Pros: Security is transparent to Alice; Cons: KDC must be secure –it's trusted!

15.2) Consider the "requesting tickets" step.

I. KDC distributes the shared session keys to reduce the number of shared keys. Everyone is sharing a key with KDC, so N shared K needed for N users. If sharing Ks directly between users, need order of N^2 shared keys for N users.

II. Yes, since her name is encrypted in TGT, which can only be decrypted by KDC.

III. Yes, since her name is encrypted in ticket and only Bob can decrypt it.

IV. Ticket to Bob is sent to Alice, then Alice forwards it to Bob, so Bob can remain stateless. Otherwise, Bob needs to store the ticket and wait for Alice to contact him. Actually, Alice also remains stateless, since she can require the ticket whenever she wants to communicate to Bob, there's no need to store any ticket for future use.

Q16: Pick all true statements about IPSec (L21 P5 & P8 - 10)

All correct.

Q17: IKE Phase 1 (L21 P12)

No, separating public key encryption and signature as two options is not over-engineering - it is necessary.

Everyone always knows his/her private key but may not initially know other side's public key. So public key signature option is more efficient, since people can start the protocol with their private key first, then search for other's public key simultaneously.

Q18: Plausible Deniability (L21 P14 - 15)

```
Protocol:
1) Alice -> Bob: "I'm Alice", {RA}Bob
2) Bob -> Alice: {RA, RB}Alice
3) Alice -> Bob: E(RB, K) // K = h(RA, RB)
```

I. Yes, Alice is authenticated. Message 3 shows Alice is Alice since only Alice can decrypt $\{R_A, R_B\}_{Alice}$ to get R_B using her private key, and also compute K to encrypt RB using K. And Bob knows R_B & K to verify if $E(R_B, K)$ is correct.

II. Yes, message 2 shows Bob is Bob, since only Bob can decrypt $\{R_A\}_{Bob}$ to get R_A using his private key and Alice knew R_A to verify.

III. It can provide plausible deniability. Everything in this protocol is a "public" operation: everyone can pick a random R_A and R_B then compute $K = h(R_A, R_B)$; $\{R_A\}_{Bob}$ and $\{R_A, R_B\}_{Alice}$ also can be computed by anyone. That is, Trudy can fake a conversation between Alice & Bob that appears valid to any observer; as a result, Alice & Bob can deny any conversation.

Q19: WEP (L21 P16 - 18)

19.1) Integrity issues:

I. cipher = IP address ⊕ keystream, so Trudy can get keystream = cipher ⊕ IP address. Then she can change the cipher so that cipher = Trudy's IP address ⊕ keystream.

II. CRC can't be used for integrity since it's linear, which means it's easy to solve. CRC only detect unintentional errors, NOT "intelligent" changes. I.e., Trudy can change ciphertext & CRC value so that checksum on plaintext remains valid. A correct algorithm to use for integrity: MAC/HMAC

19.2) Confidentiality issues:

RC4, which is a stream cipher (recall: it's based on onetime pads) is used for confidentiality so using the same key will arise security issues. The long-term key K is pre-pended with IV when used for encryption so the keys can be different. But if IV got repeated, then we are using the same key.

Also, with enough IV & ciphertext (both are public), Trudy may be able to find K! (WEP tries to prevent this by discarding first 256 keystream bytes).

Q20: GSM insecurity (L21 P23 - 24)

Pick any 2 mentioned on the lecture notes (for how to fix, you can be "creative"), for example:

- No encryption from base station to base station controller. To fix this, add encryption there.
- GSM uses weak cryptos to protect data. To fix this, we can use stronger cryptos.
- Base station is not authenticated so there are fake ones. To fix this, also authenticate the base station.
- Replay attacks possible. To fix this, modify the protocol such as signing the nonce, etc.