CS 798

Solutions to Assignment 3

1. Recall that the Computational Diffie-Hellman problem is defined as follows:

   Problem: Computational Diffie-Hellman (CDH)
   Instance: A multiplicative group $(G, \cdot)$, an element $\alpha \in G$ having order n, and two elements $\beta, \gamma \in \langle \alpha \rangle$.
   Question: Find $\delta \in \langle \alpha \rangle$ such that

   $$\log_\alpha \delta \equiv \log_\alpha \beta \times \log_\alpha \gamma \pmod{n}.$$

   (Equivalently, given α^b and α^c, find α^{bc}.)

   The problem that a passive adversary faces in trying to compute a key in the *MTI* protocol is the following:

   Problem: Computational MTI (CMTI)
   Instance: A multiplicative group $(G, \cdot)$, an element $\alpha \in G$ having order n, and four elements $\beta_1, \gamma_1, \beta_2, \gamma_2 \in \langle \alpha \rangle$.
   Question: Find $\delta \in \langle \alpha \rangle$ such that

   $$\log_\alpha \delta \equiv \log_\alpha \beta_1 \times \log_\alpha \gamma_2 + \log_\alpha \beta_2 \times \log_\alpha \gamma_1 \pmod{n}.$$

   (Equivalently, given α^{b_1}, α^{b_2}, and α^{c_1} and α^{c_2}, find $\alpha^{b_1 c_2 + b_2 c_1}$.)

   (a) Prove that CMTI can be solved easily, given an oracle to solve CDH.
   Answer: Let OracleCDH be an oracle to solve CDH, and let GroupMultiply be an algorithm to perform group multiplication. Then CMTI can be solved as follows:

   Algorithm: SolveCMTI$(G, \alpha, \beta_1, \gamma_1, \beta_2, \gamma_2)$
   external OracleCDH, GroupMultiply
   $\delta_1 \leftarrow$ OracleCDH$(G, \alpha, \beta_1, \gamma_2)$
   $\delta_2 \leftarrow$ OracleCDH$(G, \alpha, \beta_2, \gamma_1)$
   $\delta \leftarrow$ GroupMultiply(G, δ_1, δ_2)
   return δ

   (b) Prove that CDH can be solved easily, given an oracle to solve CMTI.
   Answer: Let id denote the identity element of the group G, and let OracleCMTI be an oracle to solve CMTI. Then CDH can be solved as follows:

   Algorithm: SolveCDH$(G, \alpha, \beta, \gamma)$
   external OracleCMTI
   $\delta \leftarrow$ OracleMTI$(G, \alpha, \beta, \mathsf{id}, \mathsf{id}, \gamma)$
   return δ

2. In this question, suppose that $p = 12132217$.

(a) Factor $p-1$, and prove that the smallest primitive element modulo p is $\alpha = 5$.

Answer: $p - 1 = 2^3 3^2 167^1 1009^1$. An integer $\alpha \in \mathbb{Z}_p{}^*$ is a primitive root if and only if $\alpha^{(p-1)/q} \not\equiv 1 \pmod p$ for $q = 2, 3, 167, 1009$. $2^{(p-1)/2} \equiv 1 \pmod p$, so 2 is not a primitive root. $3^{(p-1)/2} \equiv 1 \pmod p$, so 3 is not a primitive root. $4^{(p-1)/2} \equiv 1 \pmod p$, so 4 is not a primitive root. $5^{(p-1)/2} \equiv 12132216 \pmod p$; $5^{(p-1)/3} \equiv 11746374 \pmod p$; $5^{(p-1)/167} \equiv 6224301 \pmod p$; and $5^{(p-1)/1009} \equiv 7475787 \pmod p$; so 5 is a primitive root.

(b) Suppose $p - 1 \equiv 0 \pmod q$ and $\beta \in \mathbb{Z}_p{}^*$. Let $a_0 = \log_\alpha \beta \bmod q$. Then it is possible to prove that

$$\beta^{(p-1)/q} \equiv \alpha^{a_0(p-1)/q} \pmod p.$$

(You are not required to prove this formula.) The formula permits a_0 to be computed in at most q "iterations".

Using this formula with $q = 167$, apply the method described in class to prove that $(5957975, 512317, 11127721)$ is not a Diffie-Hellman triple in $\mathbb{Z}_p{}^*$.

Answer: Using the formula, we see that $\log_5 5957975 \equiv 130 \pmod{167}$, $\log_5 512317 \equiv 66 \pmod{167}$, and $\log_5 11127721 \equiv 39 \pmod{167}$. Now, $130 \times 66 \equiv 63 \not\equiv 39 \pmod{167}$, so $(5957975, 512317, 11127721)$ is not a Diffie-Hellman triple in $\mathbb{Z}_p{}^*$.

3. We describe a secret-key based three-party session key distribution scheme. In this scheme, K_{Alice} is a secret key shared by Alice and the *TA*, and K_{Bob} is a secret key shared by Bob and the *TA*.

Protocol: SESSION KEY DISTRIBUTION SCHEME

1. Alice chooses a random number, r_A. Alice sends $ID(Alice)$, $ID(Bob)$ and

$$y_A = e_{K_{Alice}}(ID(Alice) \parallel ID(Bob) \parallel r_A)$$

to Bob.

2. Bob chooses a random number, r_B. Bob sends $ID(Alice)$, $ID(Bob)$, y_A and

$$y_B = e_{K_{Bob}}(ID(Alice) \parallel ID(Bob) \parallel r_B)$$

to the *TA*.

3. The *TA* decrypts y_A using the key K_{Alice} and it decrypts y_B using the key K_{Bob}, thus obtaining r_A and r_B. It chooses a random session key, K, and computes

$$z_A = e_{K_{Alice}}(r_A \parallel K)$$

and

$$z_B = e_{K_{Bob}}(r_B \parallel K).$$

z_A is sent to Alice and z_B is sent to Bob.

4. Alice decrypts z_A using the key K_{Alice}, obtaining K; and Bob decrypts z_B using the key K_{Bob}, obtaining K.

(a) State all consistency checks that should be performed by Alice, Bob and the *TA* during a session of the protocol.

Answer: The *TA* receives two identifying strings, say $ID(Alice)$ and $ID(Bob)$, in the clear. Then the *TA* should verify that $d_{K_{Alice}}(y_A)$ has the form

$$(ID(Alice) \parallel ID(Bob) \parallel r_A);$$

and $d_{K_{Bob}}(y_B)$ has the form

$$(ID(Alice) \parallel ID(Bob) \parallel r_B);$$

where r_A and r_B are arbitrary.

Alice should verify that $d_{K_{Alice}}(z_A)$ has the form $(r_A \parallel K)$, where r_A was the challenge that she chose in step 1.

Bob should verify that $d_{K_{Bob}}(z_A)$ has the form $(r_B \parallel K)$, where r_B was the challenge that he chose in step 2.

(b) The protocol is vulnerable to an attack if the *TA* does not perform the necessary consistency checks you described in part (a). Suppose that Oscar replaces $ID(Bob)$ by $ID(Oscar)$, and he also replaces y_B by

$$y_O = e_{K_{Oscar}}(ID(Alice) \parallel ID(Bob) \parallel r'_B)$$

in step 2, where r'_B is random. Describe the possible consequences of this attack if the *TA* does not carry out its consistency checks properly.

Answer: The *TA* receives identity strings $ID(Alice)$, $ID(Oscar)$, so it computes

$$d_{K_{Alice}}(y_A) = (ID(Alice) \parallel ID(Bob) \parallel r_A)$$

and

$$d_{K_{Oscar}}(y_O) = (ID(Alice) \parallel ID(Bob) \parallel r'_B).$$

The *TA* should reject because $ID(Bob)$ occurs in the decryption of the strings y_A and y_O (rather than $ID(Oscar)$). If the *TA* fails to perform this check, it may go ahead and compute

$$z_A = e_{K_{Alice}}(r_A \parallel K)$$

and

$$z_O = e_{K_{Oscar}}(r'_B \parallel K).$$

z_A is sent to Alice and z_O is sent to Oscar. Alice and Oscar both obtain the key K, but Alice thinks she shares a session key with Bob. If Alice encrypts information using the key K, Oscar will be able to decrypt it and Bob will not be able to do so.

(c) In this protocol, encryption is being done to ensure both confidentiality and data integrity. Indicate which pieces of data require encryption for the purposes of confidentiality, and which ones only need to be authenticated. Rewrite the protocol, using MACs for authentication in the appropriate places.

Answer: In general, the first transmission of challenges and IDs does not need to be encrypted or authenticated. (Authentication does not achieve anything useful, because the adversary can reuse values from a previous session, even if they are authenticated.) Responses to challenges (including IDs) need to be authenticated (not encrypted). Session keys should be encrypted, and the encrypted session keys should be authenticated.

The protocol could be modified as follows:

Protocol: SESSION KEY DISTRIBUTION SCHEME

1′. Alice chooses a random number, r_A. Alice sends $ID(Alice)$, $ID(Bob)$ and r_A to Bob.

2′. Bob chooses a random number, r_B. Bob sends $ID(Alice)$, $ID(Bob)$, r_A and r_B to the TA.

3′. The TA chooses a random session key, K, and computes

$$
\begin{aligned}
z_A &= e_{K_{Alice}}(K), \\
w_A &= MAC_{Alice}(ID(Alice) \parallel ID(Bob) \parallel z_A \parallel r_A), \\
z_B &= e_{K_{Bob}}(K), \quad \text{and} \\
w_B &= MAC_{Bob}(ID(Alice) \parallel ID(Bob) \parallel z_B \parallel r_B)
\end{aligned}
$$

(z_A, w_A) is sent to Alice and (z_B, w_B) is sent to Bob.

4′. Alice decrypts z_A using the key K_{Alice}, obtaining K; and Bob decrypts z_B using the key K_{Bob}, obtaining K. Alice and Bob also verify the MACs, w_A and w_B, respectively.

4. We describe a public-key protocol in which Alice chooses a random session key and transmits it to Bob in encrypted form. (A protocol of this type is called a *key transport scheme.*) In this scheme, K_{Bob} is Bob's public encryption key. Alice and Bob also have private signing keys and public verification keys for a signature scheme.

Protocol: PUBLIC-KEY KEY TRANSPORT SCHEME

1. Bob chooses a random challenge, r_1. He sends r_1 and $\mathbf{Cert}(Bob)$ to Alice.
2. Alice verifies Bob's public encryption key, K_{Bob}, on the certificate $\mathbf{Cert}(Bob)$. Then Alice chooses a random session key, K, and computes
$$z = e_{K_{Bob}}(ID(Alice) \parallel K).$$
She also computes
$$y_1 = sig_{Alice}(r_1 \parallel z)$$
and sends $\mathbf{Cert}(Alice)$, z and y_1 to Bob.
3. Bob verifies Alice's public verification key, ver_{Alice}, on the certificate $\mathbf{Cert}(Alice)$. Then he checks that
$$ver_{Alice}(r_1 \parallel z, y_1) = true.$$
Bob also decrypts z, and verifies that $d_{K_{Bob}}(z)$ has the form $ID(Alice) \parallel K$ for some K. If so, then Bob outputs "accept" and takes K to be the new session key; otherwise, Bob outputs "reject" and quits. Bob also computes
$$y_2 = sig_{Bob}(z)$$
and sends y_2 to Alice.
4. Alice verifies Bob's public verification key, ver_{Bob}, on the certificate $\mathbf{Cert}(Bob)$. Then she checks that
$$ver_{Bob}(z, y_2) = true.$$
If so, then Alice outputs "accept"; otherwise, Alice outputs "reject".

(a) Prove that the above protocol is a secure mutual identification scheme. Analyze an active adversary's probability of successfully deceiving Alice or Bob, given suitable assumptions on the security of the signature scheme.

Answer: We assume the signature scheme is (q, ϵ)-*secure*: the adversary cannot construct a valid signature for any new message with probability greater than ϵ, given that the adversary has previously seen valid signatures for at most q messages.

Suppose the adversary has (passively) observed $q/2$ sessions between Bob and Alice, in which a total of q valid signatures have been transmitted between them. Then the adversary attempts to have Alice or Bob "accept" in some new session, say $\mathcal{S}_{new}$, in which the adversary is active.

Suppose that the random challenges are (positive) k-bit integers. Under these conditions, we can easily give an upper bound of $q/2^k + 2\epsilon$ on the adversary's probability of deceiving at least one of Alice or Bob. The adversary can reuse a signature from an old session if a challenge is repeated by coincidence, or he can try to guess (i.e., forge) a signature for a new challenge.

The following analysis is not required in order to obtain full marks to this question:

However, we also need to address the following question: can an active adversary obtain Alice or Bob's signature after a flow in which the adversary is active, other than by the two methods mentioned above?

In flow 1, Bob transmits a challenge r_1 and his certificate; and in flow 2 he receives $\mathbf{Cert}(Alice)$, z and y_1, where

$$d_{K_{Bob}}(z) = ID(Alice) \parallel K$$

for some K, and

$$ver_{Alice}(r_1 \parallel z, y_1) = true.$$

Suppose first that Oscar changed the value of r_1, to r_1', say, before sending it to Alice. Then Alice would not have signed r_1, and Alice's signature on r_1 (which is verified by Bob) must have arisen by one of the two methods enumerated above. Therefore, with high probability, Oscar did not change r_1.

Could Oscar have replaced Bob's certificate with someone else's, say Charlie's? If this happened, then Alice would have computed

$$z' = e_{K_{Charlie}}(ID(Alice) \parallel K)$$

and

$$y_1 = sig_{Alice}(r_1 \parallel z').$$

If Oscar replaced z' by some other string, say z'', then he would be forced to find a valid signature on z'' (i.e., by creating a new forged signature, or by reusing a previously generated signature). On the other hand if Oscar sends z' to Bob, then Bob will not accept because $d_{K_{Bob}}(z')$ will not have the correct form. (The only way Bob would accept is if

$$e_{K_{Charlie}}(ID(Alice) \parallel K) = e_{K_{Bob}}(ID(Alice) \parallel K')$$

for some K', which, presumably, will happen only with some small (unspecified) probability.)

We conclude that Oscar was not active during flow 1 or flow 2. Now we can consider flow 3. We have already argued that Oscar was not active in the first two flows. In the third flow, Bob's signature on the new, random quantity z is created. Clearly this can only have been done in reponse to Alice sending z to Bob in the previous flow. If Oscar is active, then this signature is newly created by Oscar or re-used from a previous session.

End of optional part of answer

Therefore the analysis of the protocol "reduces" to previous analyses, in which we analyze the probability that a challenge is repeated from a previous session, or that Oscar can create a new valid forged signature. Oscar's probability of deceiving Alice or Bob in this way is at most $2 \times ((q/2)/2^k + \epsilon) = q/2^k + 2\epsilon$.

(b) What type of key authentication or confirmation is provided by this protocol? Justify your answer briefly.

Note: You can make an appropriate security assumption regarding the encryption scheme, such as the following: given

$$z = e_{K_{Bob}}(ID(Alice) \parallel K),$$

where K is unknown, it is infeasible for Oscar to compute

$$z' = e_{K_{Bob}}(ID(Oscar) \parallel K)$$

for the same K. (This property is called *non-malleability*.)

Answer: Bob receives an encrypted version of K in the second flow of the protocol and he believes that he has been communicating with Alice (because the protocol is a mutual identification scheme). Bob therefore believes that Alice has sent the value z to Bob and it has not been changed by an adversary.

Does this imply that Alice knows the value of K? Certainly it does if Alice follows the protocol, i.e., if she chose K and encrypted $ID(Alice) \parallel K$ to form z. But what if Alice re-used or modified a value z from a previous session which included Bob?

If the previous session was one between Alice and Bob, and the value of z *was not* changed, then we can argue that Alice must have known the value of K at that time, so she still knows K (unless she "forgot it").

If the previous session was one between Alice and Bob, and the value of z *was* changed; or if the previous session was one between Charlie and Bob, say, then Alice would have to create an encryption of a "new" plaintext $ID(Alice) \parallel K$ without knowing K. This is impossible if the encryption scheme is non-malleable.

Summarizing, Bob is convinced that Alice *has* computed K, and no one else can compute K. Hence, the protocol provides Bob with explicit key confirmation.

Alice receives Bob's signature on z, where z was chosen by Alice and encrypted with Bob's public key. Alice believes that she has been communicating with Bob. Therefore the protocol provides Alice with implicit key confirmation (she believes that Bob *can* compute K, namely, by decrypting z). Alice does not know if Bob has actually decrypted z.

(c) Prove that the protocol is insecure if Alice does not sign z in step 2, by describing two attacks that an active adversary (Oscar) can carry out. Briefly discuss the consequences of these attacks.

Hint: In one attack, Oscar deceives Bob (the initiator of the session); and in the other attack, Oscar deceives Alice.

Notes:

- You can assume that the length of a challenge is the same as the length of an encrypted string (e.g., $|r_1| = |z|$).
- A successful attack is any attack that causes at least one of the honest particpants to "accept" and have false beliefs at the end of a session.

Answer: Oscar deceives Bob in the following attack (note that other attacks are possible, too):

$$
\begin{array}{ccccc}
\text{Bob} & & \text{Oscar} & & \text{Alice} \\
 & \xrightarrow{\quad r_1, \mathbf{C}(B) \quad} & & \xrightarrow{\quad r_1, \mathbf{C}(O) \quad} & \\
 & \xleftarrow{\quad z', y_1, \mathbf{C}(A) \quad} & z' = e_{K_B}(ID(A) \parallel K) & \xleftarrow{\quad z, y_1, \mathbf{C}(A) \quad} & \begin{array}{l} z = e_{K_O}(ID(A) \parallel K) \\ y_1 = sig_A(r_1) \end{array} \\
y_2 = sig_B(z') & \xrightarrow{\quad y_2 \quad} & & &
\end{array}
$$

Here Bob accepts, thinking he is sharing a key with Alice. However, Oscar also knows the value of the key.

The second attack is as follows:

$$
\begin{array}{ccccc}
\text{Bob} & & \text{Oscar} & & \text{Alice} \\
 & & & \xrightarrow{\quad r_1, \mathbf{C}(B) \quad} & \\
 & \xleftarrow{\quad z, \mathbf{C}(A) \quad} & & \xleftarrow{\quad z, y_1, \mathbf{C}(A) \quad} & \begin{array}{l} z = e_{K_B}(ID(A) \parallel K) \\ y_1 = sig_A(r_1) \end{array} \\
\begin{array}{l} z' = e_{K_A}(ID(B) \parallel K') \\ y_2 = sig_B(z) \end{array} & \xrightarrow{\quad z', y_2, \mathbf{C}(B) \quad} & & \xrightarrow{\quad y_2 \quad} &
\end{array}
$$

Here Alice accepts, thinking she has shared a key with Bob. Neither Oscar nor Bob knows the value of the key chosen by Alice. However, the scheme is not a secure mutual identification scheme, because Alice has accepted in a session in which the adversary is active.