1. The first X.509 standard contained a three-way authentication procedure which had a security flaw. The essence of the protocol is as follows:

1. $A \rightarrow B$: $t_A$, $r_A$, $ID_B$, $E_B(K_{AB})$, $Sig_A(t_A, r_A, B, E_B(K_{AB}))$

2. $B \rightarrow A$: $t_B$, $r_B$, $A$, $r_A$, $E_A(K_{AB})$, $Sig_B(t_B, r_B, A, r_A, E_A(K_{AB}))$

3. $A \rightarrow B$: $r_B, Sig_A(r_B)$

Above, $r_A$ and $r_B$ are random numbers, nonces, $t_A$ and $t_B$ are timestamps, $Sig_A$ means a signature with A’s private key, $E_B$ means an encryption with B’s public key, and $K_{AB}$ is a generated session key (which is not important in this phase, only later). The text of X.509 states that checking timestamps is optional for three-way authentication. But consider the following example: Suppose A and B have used the preceding protocol on some previous occasion, and that opponent C has intercepted the three preceding messages. In addition, suppose that timestamps are not used and are all set to 0. Finally, suppose C wishes to impersonate A to B. C initially sends the first captured message to B:

$C \rightarrow B$: $0$, $r_A$, $ID_B$, $E_B(K_{AB})$, $Sig_A(0, r_A, B, E_B(K_{AB}))$.

B responds, thinking it is talking to A but is actually talking to C:

$B \rightarrow C$: $0$, $r'_B$, $A$, $r_A$, $E_A(K_{AB})$, $Sig_B(0, r'_B, A, r_A, E_A(K_{AB}))$.

C meanwhile causes A to initiate authentication with C by some other means. As a result, A sends C the following:

$A \rightarrow C$: $0$, $r'_A$, $ID_C$, $E_C(K'_{AB})$, $Sig_A(0, r'_A, C, E_C(K'_{AB}))$.

C responds to A using the same nonce provided to C by B:

$C \rightarrow A$: $0$, $r'_B$, $A$, $r'_A$, $E_A(K'_{AB})$, $Sig_B(0, r'_B, A, r'_A, E_A(K'_{AB}))$.

A responds with

$A \rightarrow C$: $r'_B, Sig_A(r'_B)$.

This is exactly what C needs to convince B that it is talking to A, so C now repeats the incoming message back out to B:

$C \rightarrow B$: $r'_B, Sig_A(r'_B)$.

So B will believe it is talking to A whereas it is actually talking to C. Suggest a simple solution to this problem that does not involve the use of timestamps.
2. Discrete logarithm problem: If \( p \) is a large prime, \( \alpha \) its primitive root, and if \( \alpha^k \) is known modulo \( p \), it is computationally impossible to calculate \( k \) from \( p \), \( \alpha \), and \( \alpha^k \). However, if \( p, k, \alpha^k \) are known and \( \gcd(k, p - 1) = 1 \), it is possible effectively to calculate \( \alpha \). Show how. (Hint: Because \( \alpha \) is a primitive root, \( p - 1 \) is the smallest positive exponent such that \( \alpha^{p-1} = 1 \).)

3. Consider the following key transport protocol which uses public key cryptography:

1. \( A \rightarrow B: \ E_B(A, K_{AB}, N_A) \)
2. \( B \rightarrow A: \ E_A(K_{BA}, N_A, N_B) \)
3. \( A \rightarrow B: \ N_B \)

Design an attack, where the adversary \( C \) induces \( A \) to commence the protocol with \( C \), and then starts a protocol run with \( B \) while masquerading as \( A \). Use the same idea as in Lowe’s attack against public key Needham Schroeder.

4. Consider the following key exchange protocol:

1. \( A \rightarrow S: \ A, \{T_A, B, K_{AB}\}_{K_{AS}} \)
2. \( S \rightarrow B: \ \{T_S, A, K_{AB}\}_{K_{BS}} \)

\( A \) generates the session key \( K_{AB} \) and sends it to \( B \) via the server \( S \). The protocol uses timestamps. However, there is an attack against the protocol. It assumes that the intruder \( I \) has recorded one earlier protocol run:

1’. \( I_B \rightarrow S: \ B, \{T_S, A, K_{AB}\}_{K_{BS}} \)
2’. \( S \rightarrow I_A: \ \{T_S, B, K_{AB}\}_{K_{AS}} \)
1”. \( I_A \rightarrow S: \ A, \{T'_S, B, K_{AB}\}_{K_{AS}} \)
2”. \( S \rightarrow B: \ \{T''_S, A, K_{AB}\}_{K_{BS}} \)

Explain the attack. What has the intruder achieved? What is needed (extra ingredients in the protocol or extra assumptions) in order to prevent the attack?

5. Consider the following key agreement protocols I and II. The protocols are the modifications of the basic Diffie-Hellman key exchange. The group \( \mathbb{Z}_q^* \) and its generator \( g \) are public knowledge. The participants \( A \) and \( B \) have long term private keys \( x_A \) and \( x_B \), respectively. They also have public keys \( y_A = g^{x_A} \) and \( y_B = g^{x_B} \), respectively. \( A \) chooses a secret integer \( r_A \), and \( B \) \( r_B \). The protocol I is as follows:

i) \( A \rightarrow B: \ g^{r_A} \)
ii) \( B \rightarrow A: \ g^{r_B} \)

After the second step, the participants can calculate the common secret

\[
K_{AB} = t_B^{x_A} y_B^{r_A} = t_B^{x_B} y_A^{r_B}.
\]

The second protocol II is as follows:
i) $A \rightarrow B$: $y_B^A$

ii) $B \rightarrow A$: $y_A^B$

The common key is

$$K_{AB} = g^{x_A r_B} = (y_A^B)^{x_A^{-1} r_A} = (y_B^A)^{x_B^{-1} r_B}.$$ 

Do the protocols satisfy the forward or partial forward security property?