Tutorial 1

ID

1.1 Identification 1

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
	$\xrightarrow{(Alice, pw')}$	
		1. Alice is verified if $pw' = pw$

Fig. 1. The simplest Identification Protocol.

Question: explain why this protocol is bad.

1

- **Impersonation attack**: An **impersonation attack** is an **attack** in which an adversary successfully assumes the identity of one of the legitimate parties in a system or in a communications protocol. (Adversary logins as the identity Alice)

- Sensitive information is transmitted in plaintext without protection.

- The adversary can obtain the username and pw, and then launch the impersonation attack.

1.2 Identification 2

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
1. Compute $H = H(pw)$		
	$\xrightarrow{(Alice,H)}$	
		2. Alice is verified if $H = H(pw)$

Fig. 2. The Identification Protocol Where $H(pw)$ is transmitted instead of pw.

Questions:
Is this protocol secure in terms of identification against the adversary?
Is this protocol more secure in terms of identification compared to the protocol 1?
What is the advantage of this protocol compared to the protocol 1?

2

- No, because the adversary can still launch the impersonation attack after obtaining (Alice, H(pw)).

- Since it cannot withstand the impersonation attack, this protocol is as insecure as the first one.

2

- The adversary can obtain H(pw) from the eavesdropping but it takes times to crack the pw from H(pw). Or, the adversary cannot immediately get Alice's pw from the eavesdropping.

- dictionary attack

2

- A **dictionary attack** is a form of brute force attack technique by trying to determine its passwords by trying hundreds or sometimes millions of likely possibilities.

- The adversary can pre-compute a database (dictionary) about

  pw, H(pw)

  for all potential pw. Then, with such a database, it can quickly crack the pw. (offline dictionary attack)

1.3 Identification 3

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
1. Randomly choose $r \in \{0,1\}^n$ 2. Compute $H = H(r, pw)$		
	$\xrightarrow{(Alice, r, H)}$	
		3. Alice is verified if $H = H(r, pw)$

Fig. 3. The Identification Protocol Using a Random Salt.

Questions:
Is this protocol secure in terms of identification against the adversary?
Is this protocol more secure in terms of identification compared to the protocol 2?
What is the advantage of this protocol compared to the protocol 2?

3

- No, because the adversary can still launch the impersonation attack after obtaining (Alice, r, H(r,pw)). (The adversary cannot change r here)

- Since it cannot withstand the impersonation attack, this protocol is as insecure as the protocol 2.

3

- The database about

  pw, H(pw)

  for all potential pw cannot be used to crack the pw because H(r,pw) has a random salt.

- Suppose the adversary knows (r, H(r, pw)) from the eavesdropping. The adversary needs to have the database

  pw, H(r, pw)

  for all potential pw under r specified by Alice. Then, with such a database, it could crack the pw. (offline dictionary attack)

This database cannot be used to crack pw with a different r.

1.4 Identification 4

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
		1. Randomly choose $r \in \{0,1\}^n$
	$\xleftarrow{\quad r \quad}$	
2. Compute $H = H(r, pw)$		
	$\xrightarrow{(Alice, H)}$	
		3. Alice is verified if $H = H(r, pw)$

Fig. 4. The Identification Protocol Using a Random Salt Chosen by Bob.

Questions:
Is this protocol secure if the adversary cannot compute pw from the communication?
Is this protocol still secure if r chosen by Bob is always the same?

4

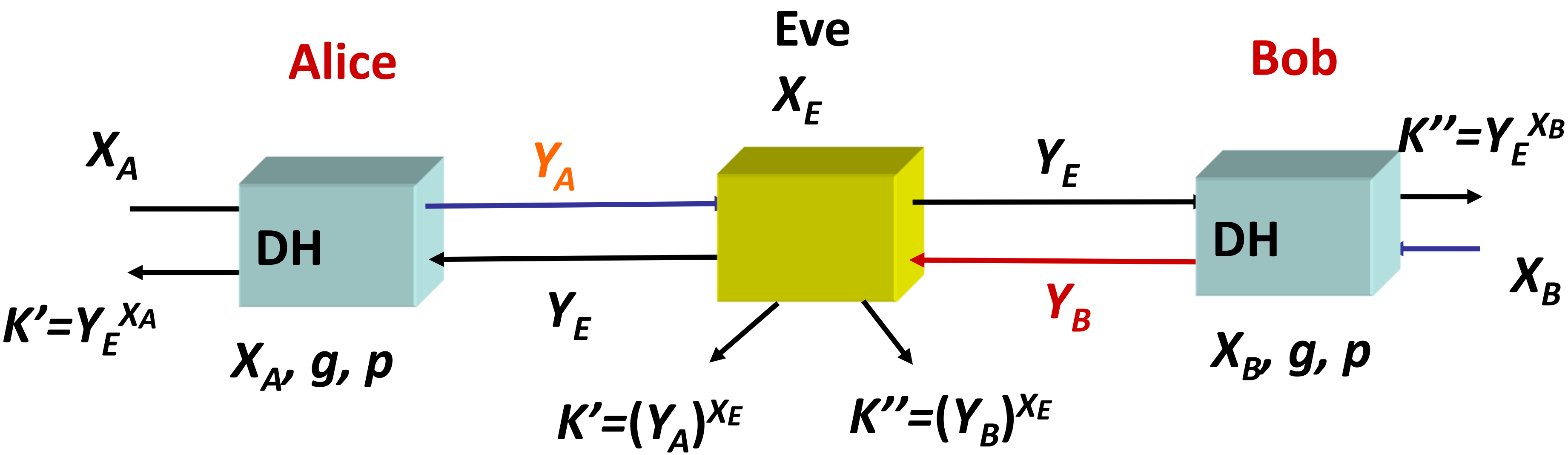
- Even the adversary eavesdrops r and H(r,pw) from the communication, they cannot be used again because r will be different when the adversary impersonates Alice during identification. For a different r, it requires the adversary to compute H(r, pw). Without knowing pw, the adversary cannot respond correctly and hence the impersonation attack cannot work.

- Insecure if r chosen by Bob is always the same, because the impersonation attack can work once the adversary eavesdrops r and H(r,pw) from the communication.

Key Establishment

Diffie-Hellman Key Agreement

- **Man-in-the-Middle Attack**



1: $A \rightarrow E$: Y_A

2: $E \rightarrow B$: Y_E

3: $B \rightarrow E$: Y_B

4: $E \rightarrow A$: Y_E

Remark: all the computation should involve the "mod p" operation, we omit this operation to simplify the presentation

1.MITM attack

What is the cause for the MITM attack?

- Lack of authentication
- Alice cannot be assured that she performed the key exchange with Bob
  - In the attack, Alice actually performed the Diffie-Hellman key exchange with Eve

2.MITM attack

How to prevent the MITM attack?

- Introduce authentication into the key agreement protocol
- Alice should ensure that the key agreement protocol must be done with Bob
  - This can be done by involving Bob’s (certified) long-term public key in the key agreement protocol
  - Intuition: only Bob has the corresponding private key to complete the protocol and compute the session key
- Vice versa

3.MTI Families of KE Protocols

- Matsumoto, Takashima, Imai (MTI, 1986)
- Incorporate ***authentication*** into Diffie-Hellman exchange by combining the

<u>long-term keys</u> and <u>ephemeral keys</u>

into a single equation.

- Three families of protocols
  - A(k), B(k), C(k) (k: any integer)
  - Only look at A(0) here

MTI Families of Protocols

- p: a large prime number (e.g. 1024-bit)
- q: a large prime number (e.g. 160-bit) such that q divides (p-1)
  - Z*p has a cyclic subgroup G which has order q
- g is a generator $G=\{g^1, g^2, g^3, \ldots, g^q\}$
- All the computation will be done under G

(G, g, p, q)

Cyclic group $\mathbf{Z_{11}}^*$

	1	2	3	4	5	6	7	8	9	10
1	1	1								
2	**2**	**4**	**8**	**5**	**10**	**9**	**7**	**3**	**6**	**1**
3	3	9	5	4	1	3				
4	4	5	9	3	1	4				
5	5	3	4	9	1	5				
6	**6**	**3**	**7**	**9**	**10**	**5**	**8**	**4**	**2**	**1**
7	**7**	**5**	**2**	**3**	**10**	**4**	**6**	**9**	**8**	**1**
8	**8**	**9**	**6**	**4**	**10**	**3**	**2**	**5**	**7**	**1**
9	9	4	3	5	1	9				
10	10	1	10							

MTI A(0)

Certified Long-term keys (in the form of digital certificates)
Alice: $(PK_A, SK_A) = (g^{x_A}, x_A)$
Bob: $(PK_B, SK_B) = (g^{x_B}, x_B)$

Key agreement
1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B}

Note: r_A and r_B are ephemeral secret keys randomly chosen in each session

Shared Key $Z_{AB} = g^{r_A x_B + r_B x_A}$

- How does each party derive the key?
- Does the MITM attack still work?
- Does this protocol have key freshness?
- Does this protocol have key authentication?

MTI A(0)

Certified Long-term keys (in the form of digital certificates)
Alice: $(PK_A, SK_A) = (g^{x_A}, x_A)$
Bob: $(PK_B, SK_B) = (g^{x_B}, x_B)$

Key agreement
1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B}

Note: r_A and r_B are ephemeral secret keys randomly chosen in each session

Shared Key $Z_{AB} = g^{r_A x_B + r_B x_A}$

- How does each party derive the key?
  - For A: $Z_{AB} = (g^{x_B})^{r_A} (g^{r_B})^{x_A}$
  - For B: $Z_{AB} = (g^{x_A})^{r_B} (g^{r_A})^{x_B}$

MTI A(0)

Certified Long-term keys (in the form of digital certificates)
Alice: $(PK_A, SK_A) = (g^{x_A}, x_A)$
Bob: $(PK_B, SK_B) = (g^{x_B}, x_B)$

Key agreement
1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B}

Note: r_A and r_B are ephemeral secret keys randomly chosen in each session

Shared Key $Z_{AB} = g^{r_A x_B + r_B x_A}$

- Does the MITM attack still work?
  - Suppose that Eve replaces g^{r_B} in the key agreement protocol by g^{r_E}, as in the MITM attack shown at the beginning, what is the session key computed by Alice?
  - Can Eve still compute Alice’s session key?

MTI A(0)

Certified Long-term keys (in the form of digital certificates)
Alice: $(PK_A, SK_A) = (g^{x_A}, x_A)$
Bob: $(PK_B, SK_B) = (g^{x_B}, x_B)$

Key agreement

1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B}

Note: r_A and r_B are ephemeral secret keys randomly chosen in each session

Shared Key $Z_{AB} = g^{r_A x_B + r_B x_A}$

- Does the MITM attack still work?
  - Suppose that Eve replaces g^{r_B} in the key agreement protocol by g^{r_E}, as in the MITM attack shown at the beginning, what is the session key computed by Alice?
  - Alice's session key: $Z = (g^{x_B})^{r_A} (g^{r_E})^{x_A}$
  - Eve can't compute this key since Eve can't produce $(g^{x_B})^{r_A}$

MTI A(0)

Certified Long-term keys (in the form of digital certificates)

Alice: $(PK_A, SK_A) = (g^{x_A}, x_A)$

Bob: $(PK_B, SK_B) = (g^{x_B}, x_B)$

Key agreement

1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B}

Note: r_A and r_B are ephemeral secret keys randomly chosen in each session

Shared Key $Z_{AB} = g^{r_A x_B + r_B x_A}$

- Key freshness?
  - Obvious, due to fresh r_A and r_B in each session

MTI A(0)

Certified Long-term keys (in the form of digital certificates)

Alice: $(PK_A, SK_A) = (g^{x_A}, x_A)$

Bob: $(PK_B, SK_B) = (g^{x_B}, x_B)$

Key agreement

1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B}

Note: r_A and r_B are ephemeral secret keys randomly chosen in each session

Shared Key $Z_{AB} = g^{r_A x_B + r_B x_A}$

- Key authentication?
  - No, due to the triangle attack (next slide)

Triangle Attack on MTI A(0)

1: $A \rightarrow B$: g^{r_A}

2: $B \rightarrow A$: g^{r_B} Shared key $Z_{AB} = g^{r_A x_B + r_B x_A}$

1’: $E \rightarrow B$: g^{r_A}

2’: $B \rightarrow E$: $g^{r'_B}$ Shared key $Z_{EB} = g^{r_A x_B + r'_B x_E}$

1’’: $E \rightarrow A$: g^{r_B}

2’’: $A \rightarrow E$: $g^{r'_A}$ Shared key $Z_{EA} = g^{r'_A x_E + r_B x_A}$

$$\mathbf{Z_{AB} = (Z_{EB}/(g^{r'_B})^{x_E}) \bullet (Z_{EA}/(g^{r'_A})^{x_E})}$$