Lecture A5:

Public-Key Cryptography for Exchanging Secret Session Keys

4th Year Course- CCSIT, UoA

Lecture goals

To understand:

- Direct key exchange protocols
- Man-in-the-middle attack
- Authenticating users and public keys with the help of Certificate Authorities
- Diffie-Hellman (DH) algorithm for secret key exchange

Using Public Keys to Exchange Secret Session Keys (1)

- From the presentation on RSA cryptography, you saw that public key cryptography is **not suitable** for the encryption of the actual message content.
- However, public key cryptography fulfills an extremely important role in the overall design and operation of secure computer networks because it leads to **superior protocols for managing and distributing secret session keys** that can subsequently be used for the encryption of actual message content.
- If a party *A* **simply** wants to receive all communications **confidentially** (meaning that *A* does not want anyone to snoop on the incoming message traffic) and that *A* is not worried about the authenticity of the messages received, all that *A* has to do is to publish his/her public key in **some publicly accessible place** (such as on a web page).

Using Public Keys to Exchange Secret Session Keys (2)

- If two parties *A* and *B* are sure about each other's identity, can be certain that a third party will not masquerade as either *A* or *B* vis-a-vis the other, one can use simple and direct key exchange protocols that do not require support from any coordinating or certificating agencies.
- The key exchange protocols are more complex for security that provides a higher level of mutual authentication between two communicating parties. These protocols may involve certificating agencies.

A Simple Key Exchange Protocol (1)

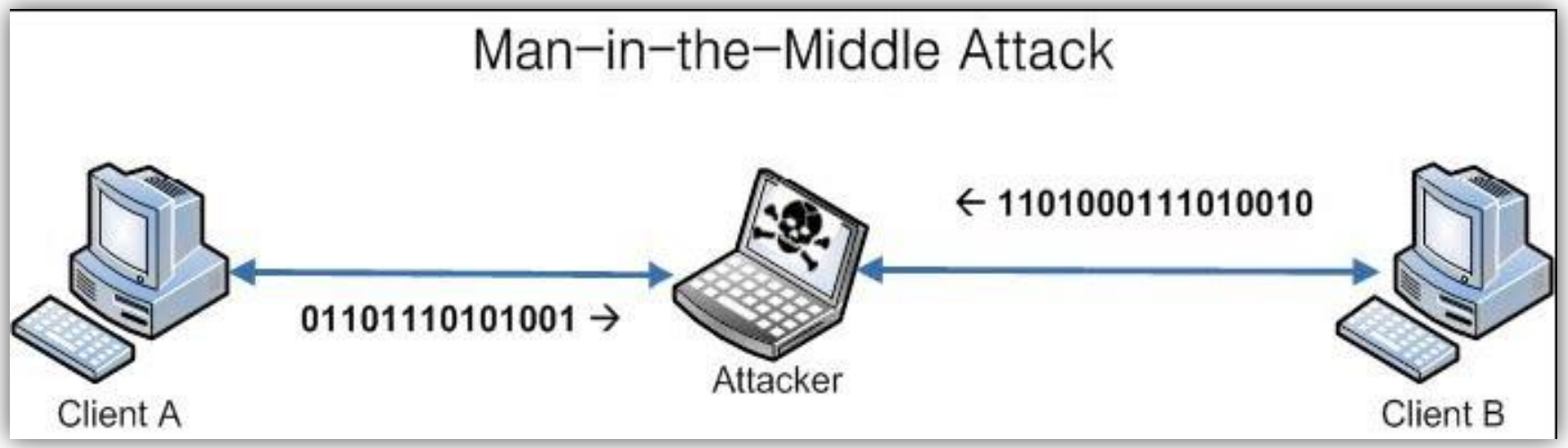
- If each of the two parties *A* and *B* has full confidence that a message received from the other party is indeed authentic, the exchange of the secret session key for a symmetric-key based secure communication link can be carried out with a simple protocol such as the one described below:

  1. Wishing to communicate with *B* , *A* generates a public/private key pair {$P U_A$, $P R_A$} and transmits an unencrypted message to *B* consisting of $P U_A$ and *A*'s identifier, ID_A (which can be *A*'s IP address). Note that $P U_A$ is party *A*'s public key and $P R_A$ the private key.
  2. Upon receiving the message from *A* , *B* generates and stores a secret session key K_S . Next, *B* responds to *A* with the secret session key K_S . This response to *A* is encrypted with *A*'s public key $P U_A$. We can express this message from *B* to *A* as $E (P U_A , K_S)$. Obviously, since only *A* has access to the private key $P R_A$, only *A* can decrypt the message containing the session key.

A Simple Key Exchange Protocol (2)

3. A decrypts the message received from B with the help of the private key PR_A and **retrieves** the session key K_S.
4. A **discards** both the public and private keys, PU_A and PR_A, and B **discards** PU_A.

- Now A and B can communicate **confidentially** with the help of the session key K_S.
- However, this protocol is vulnerable to the **man-in-the-middle attack** by an **adversary** E (See the figure below) who is able to intercept messages between A and B. This is how this attack takes place:



A Simple Key Exchange Protocol (3)

1. When A sends the very first unencrypted message consisting of $P\,U_A$ and ID_A, E intercepts the message. (Therefore, B never sees this initial message.)
2. The adversary E generates its own public/private key pair $\{P\,U_E, P\,R_E\}$ and transmits $\{P\,U_E, ID_A\}$ to B.
3. Assuming that the message received came from A, B generates the secret key K_S, encodes it with $P\,U_E$, and sends it back to A.
4. This transmission from B is again intercepted by E, who for obvious reasons is able to decode the message.
5. E now encodes the secret key K_S with A's public key $P\,U_A$ and sends the encoded message back to A.
6. A retrieves the secret key and, not suspecting any foul play, starts communicating with B using the secret key.
7. E can now successfully eavesdrop on all communications between A and B.

Certificate Authorities for Authentication (1)

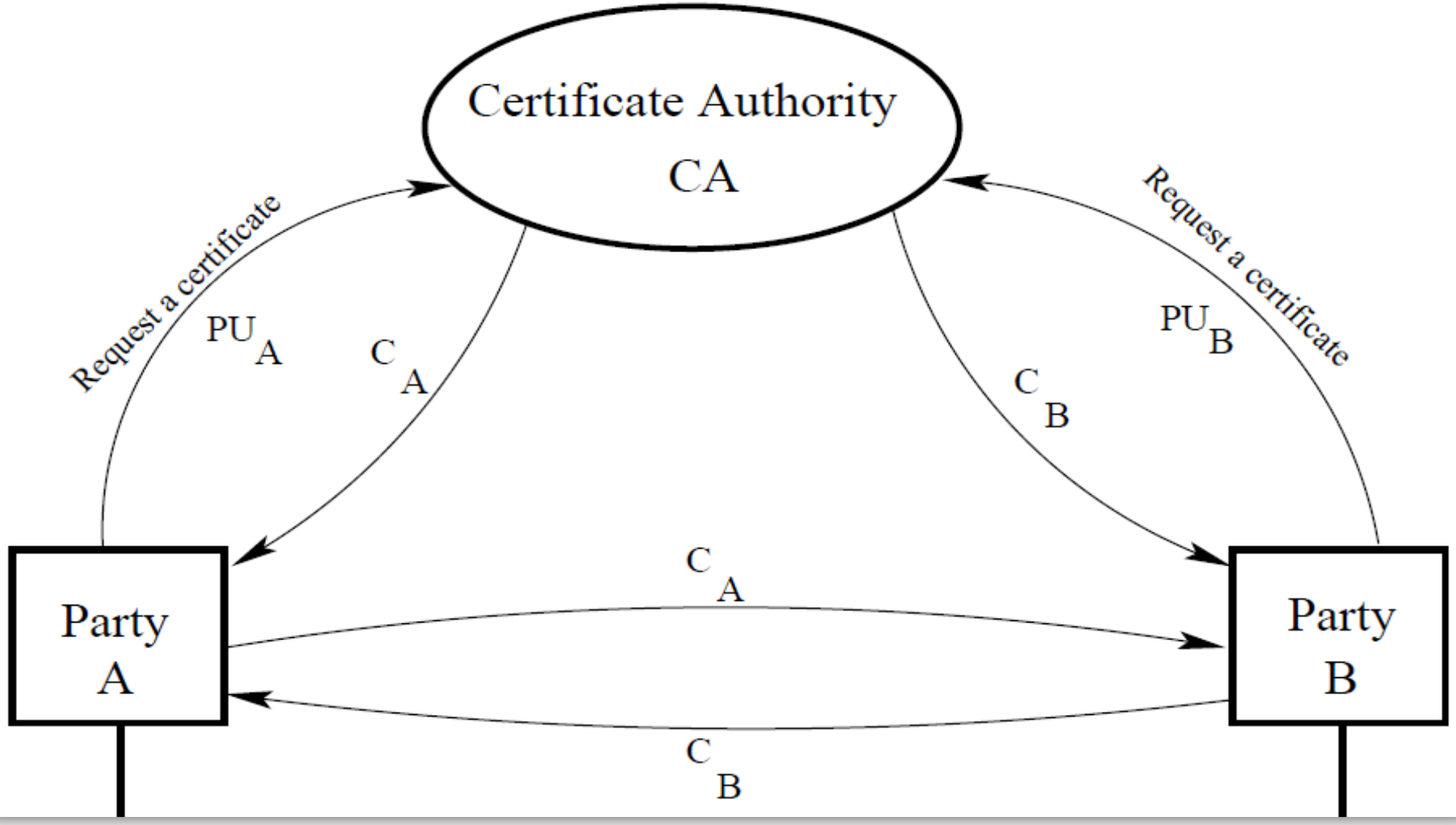
- A certificate issued by a certificate authority (CA) authenticates that its possessor is who he/she claims to be.
- To obtain a certificate, a user presents his public key to the CA.
- CA based authentication of a user is based on the assumption that when a new user applies to the CA for a certificate, the CA can authenticate the identity of the applicant through other means.
- Stated simply, a certificate assigned to a user consists of the user's public key, the identifier of the key owner, a time stamp (in the form of a period of validity), etc., the whole block encrypted with the CA's private key. Encrypting of the block with the CA's private key is referred to as the CA having signed the certificate. We may therefore express a certificate issued to A by CA to be:

$$E\,(PR_{auth}, [T, ID_A, PU_A])$$

Certificate Authorities for Authentication (2)

- Subsequently, when *A* presents his/her certificate to *B* , the latter can **verify the legitimacy** of the certificate by decrypting it with the CA's public key. (Successful decryption authenticates the certificate issuing authority.) This also provides *B* with **authentication** for *A*'s **identity** since only the real *A* could have provided a legitimate certificate with *A*'s identifier in it.
- Having established the certificate's legitimacy, having authenticated *A* , and having acquired *A*'s public key, *B* responds back to *A* with its **own certificate**. *A* processes *B* 's certificate in the **same manner** as *B* processed *A*'s certificate.
- This exchange results in *A* and *B* acquiring **authenticated public keys for each other**.
- The figure in the next slide shows this approach to user and public key authentication

(Party A initiates a request for the link)



The notion of Z_p^* and cyclic subgroups (1)

- The **order** of a group is the cardinality of the group, meaning the number of elements in the group.
- Note that Z_p^* is not the same as the more familiar finite field Z_p. The elements of the group Z_p^* are all the element of the set of remainders Z_p that are **coprime** to p and the group p operator is **multiplication** modulo p. With p as a prime number, the elements of the group Z_p^* are obviously $\{1, 2, 3, \ldots, p-1\}$.
- When a group contains a subgroup, the **order** of the subgroup divides the order the group (**Lagrange's theorem**).
- A **cyclic group** has the property that it contains an element α so that all of the group elements can be expressed as α^k for different values of the integer exponent k. Continuing to assume that the **group operator** is multiplication modulo p, the group identity element in a cyclic group is always 1 and this element corresponds to the power α^0.

The notion of Z_p^* and cyclic subgroups (2)

- When we increment the exponent beyond 0 , we obtain different value of the cyclic group. At some point, we will get $\alpha^M \equiv 1 \pmod{p}$. This M will turn out to be equal to the number of elements in the cyclic group, that is, equal to the order of the cyclic group.
- The group Z_p^* that we will use for the DH protocol will contain various cyclic subgroups that can be obtained by choosing any arbitrary element from the set $\{2, 3, \ldots, p-2\}$ as a generator and then collecting all its powers modulo p . (We intentionally exclude 1 and $p-1$ for α because, as generators, they both produce trivial cyclic subgroups.) We are specifically interested in that cyclic subgroup whose order M is large.
- Note that by Lagrange's theorem, the order M will always be a divisor of the order $p-1$ of the group Z_p^*.
- Let α be the generator element for such a subgroup. Usually α is chosen so that the order M is a large prime factor of $p-1$.

Diffie-Hellman Algorithm for Exchanging Secret Keys (1)

- When the authenticity of two parties can be established by other means, it would be possible then to use an approach for creating a shared secret key that is based on the Diffie-Hellman (DH) Key Exchange algorithm (see the next figure in slide 17).
- Two parties *A* and *B* using this algorithm for creating a shared secret key first agree on a large prime number *p* and an element α of Z_p^* that generates cyclic subgroup of large order.
- The pair of numbers (p, α) is public. This pair of numbers may be used for several runs of the protocol. These two numbers may even stay the same for a large number of users for a long period of time.

Diffie-Hellman Algorithm for Exchanging Secret Keys (2)

- *A* and *B* also make available to each other their respective public keys to be calculated as shown below.
- We will denote *A*'s and *B*'s private keys by X_A and X_B. And their public keys by Y_A and Y_B. In other words, *X* stands for **private** and *Y* for **public**.
- *A* selects a random number X_A from the set $\{1, 2, \ldots, p-2\}$ to serve as his/her private key. *A* then calculates a public-key integer Y_A that is guaranteed to exist:

$$Y_A = \alpha^{X_A} \bmod p$$

A makes the public key Y_A available to *B*.

Diffie-Hellman Algorithm for Exchanging Secret Keys (3)

- Similarly, B selects a random number X_B from the set $\{1, 2, \ldots, p-2\}$ to serve as his/her private key. B then calculates an integer Y_B that serves his/her public key:

$$Y_B = \alpha^{X_B} \bmod p$$

B makes the public-key Y_B available to A.

- A now calculates the secret key K from his/her private key X_A and B's public key Y_B:

$$K = (Y_B)^{X_A} \bmod p \quad (1)$$

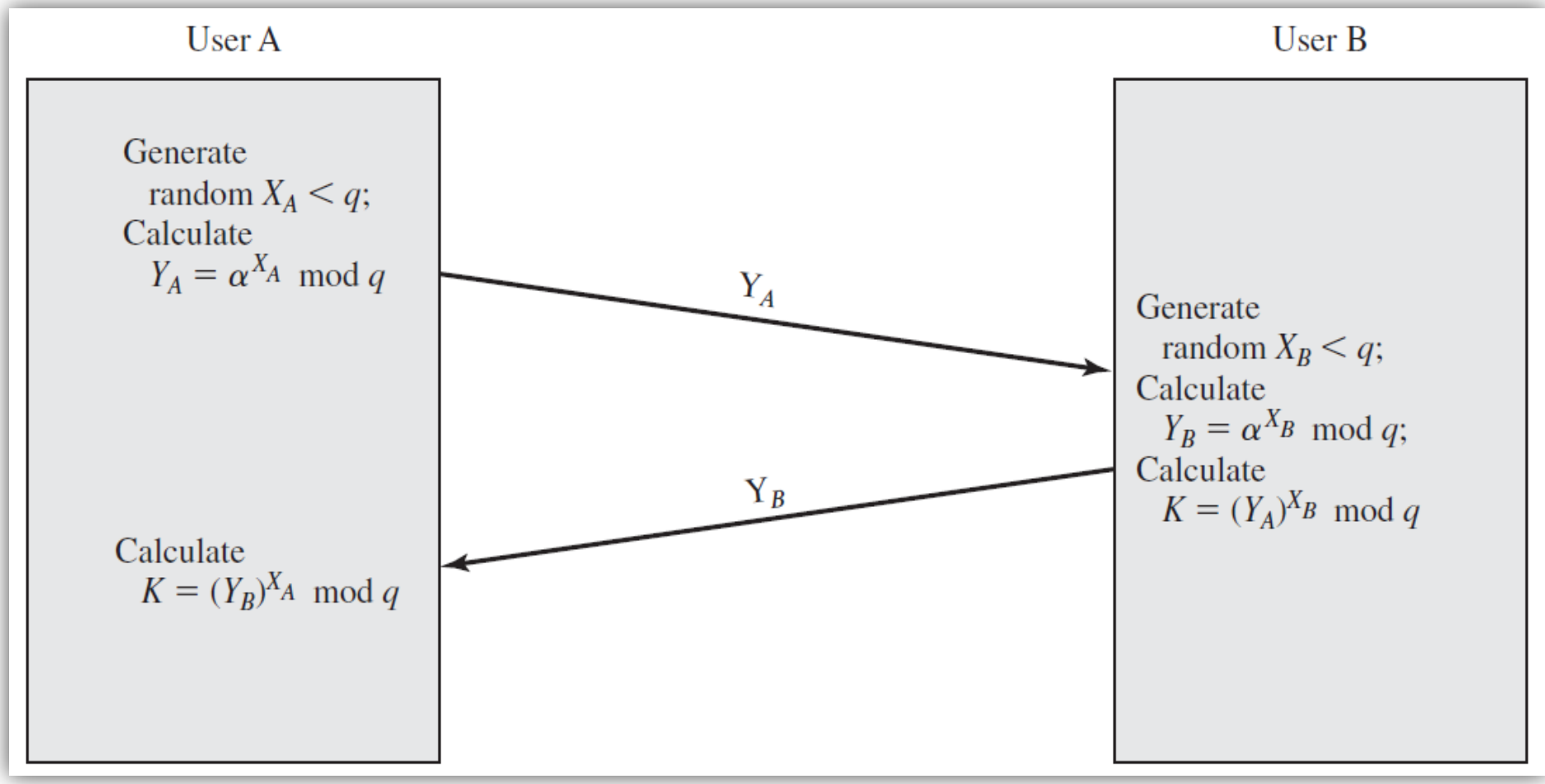
- B carries out a similar calculation for locally generating the shared secret key K from his/her private key X_B and A's public key Y_A:

$$K = (Y_A)^{X_B} \bmod p \quad (2)$$

Diffie-Hellman Algorithm for Exchanging Secret Keys (4)

- The following equalities demonstrate that the secret key *K* in both the equation Eq. (1) and Eq. (2) will be the same:

$$
\begin{aligned}
K \text{ as calculated by } A &= (Y_B)^{X_A} \bmod p \\
&= (\alpha^{X_B} \bmod p)^{X_A} \bmod p \\
&= (\alpha^{X_B})^{X_A} \bmod p \\
&= \alpha^{X_B X_A} \bmod p \\
&= (\alpha^{X_A})^{X_B} \bmod p \\
&= (\alpha^{X_A} \bmod p)^{X_B} \bmod p \\
&= (Y_A)^{X_B} \bmod p \\
&= K \text{ as calculated by } B
\end{aligned}
$$



[Note that *q* is being used here in this figure instead of *p*]

Security of Diffie-Hellman Algorithm

- The seemingly magical thing about the DH protocol is that an eavesdropper having access to the public keys for both A and B would still **not be able** to figure out the secret key K .
- The security of the Diffie-Hellman algorithm is based on the fact that whereas it is relatively **easy to compute the powers** of an integer in a finite field, it is extremely **hard to compute the discrete logarithms**. For a adversary to figure out the private keys X_A or X_B from a knowledge of all of the **publicly available information** $\{ p , \alpha , Y_A , Y_B \}$, the adversary would have to carry out the following sort of a **discrete logarithm** calculation

$$X_A = d\log_{\alpha,p} Y_A$$

  for which there **do not exist any efficient** algorithms.
- However, it can be shown that [**How?**] the DH protocol is vulnerable to the **man-in-the-middle attack**, the DH protocol must be used in conjunction with sender **authentication**.

Finally . . .

- Acknowledgment: These lecture notes are based on the textbook by William Stallings and notes prepared by Avinash Kak, Purdue University. My sincere thanks are devoted to them and to all other people who offered the material on the web.

- Students are advised to study and solve the problems and answer the questions in **Assignment-A5**.