KF School of Computing and Information Sciences
Florida International University

CNT 4403
Computing and Network Security

Key Management – Symmetric & Asymmetric Keys

Dr. Kemal Akkaya

E-mail: *kakkaya@fiu.edu*

Key Management

- **Key management is the set of techniques and procedures supporting the establishment and maintenance of keying_relationships between authorized parties**
  - Symmetric Key Management
  - Public Key Management
- **Key management encompasses techniques and procedures supporting:**
  - generation, distribution, and installation of keying material
  - update, revocation, and destruction of keying material
  - storage, backup/recovery, and archival of keying material

Symmetric Session Keys

- Real-world communication between two end system is done via temporary symmetric keys called session keys.
- Used for a predefined time and then discarded
  - Needs to be renewed for each new session
- Advantages
  - Lifetime is too short for doing cryptanalysis
  - If it is compromised, only the conversation during that single communication session is decryptable by the eavesdroppers
- How to generate and distribute these keys?
  - Key Distribution Problem

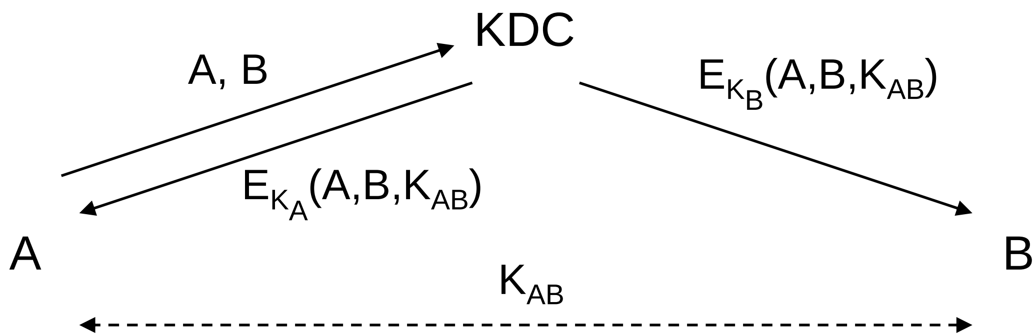
Session Key Distribution

- **How to have two parties agree on a session key (or secret key) securely?**
  - A can select key and physically deliver to B
  - Third party can select & deliver key to A & B
  - if A & B have communicated previously, they can use previous key to encrypt a new key
  - if A & B have secure communications with a third party C, C can relay key between A & B
    - Key Distribution Center (KDC)
- **Other Options:**
  - Via Public-key systems
    - Keys are distributed via public-key encryption
    - A.k.a Key encapsulation Mechanism (KEM)
  - Diffie-Hellman Key Exchange

Solution 1: Key Distribution Center (KDC)

❑ A simple protocol:

- Each user shares a long-term secret key (master key) with the KDC.
  - ✓ Master Key – renewed infrequently using non-cryptographic approach
- K_A, K_B: Long-term secret keys of Alice, Bob.



❑ Problems with this protocol:

- All resources are vulnerable if the KDC is compromised
- KDC is a single point of failure / performance bottleneck
- Possible delayed delivery of $E_{K_B}(A,B,K_{AB})$
- No freshness guarantee for B (i.e., Trudy can replay $E_{K_B}(A,B,K_{AB})$ for a previously compromised K_{AB}).

Solution 2: Via PKC

- **A simple protocol (Key Encapsulation Mechanism)**
  - Public keys are obtained in advance
  - session key transport with public key encryption between two parties:

$$A \xrightarrow{E_B(A,\ B,\ K_{AB})} B$$

$$A \xleftarrow{E_{K_{AB}}(M)} B$$

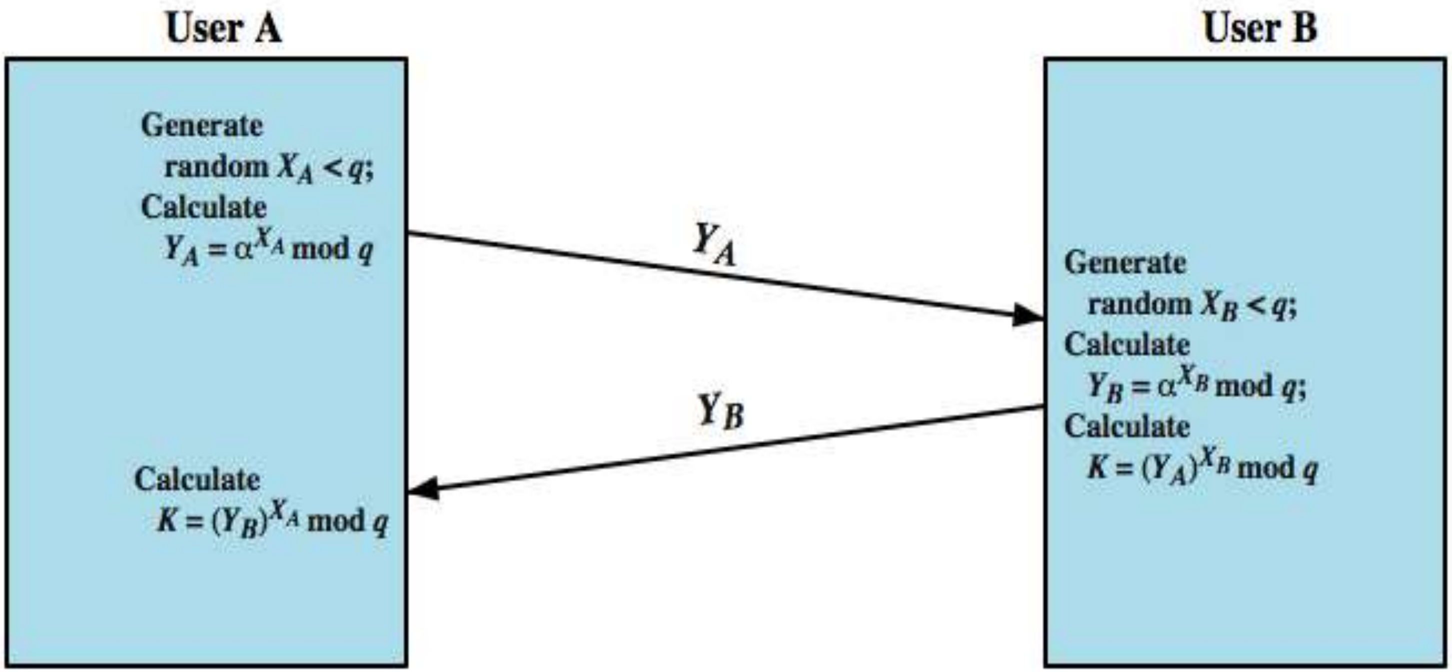
- **Problems with this protocol:**
  - Authentication of both parties
  - No freshness guarantee for B
- **This also works for Symmetric-Keys**
  - E.g., if both parties share a long term Master key and would like to generate a session key

Solution 3: Diffie-Hellman Key Exchange

- **First scheme to exchange a secret key**
  - Used in symmetric ciphers
- **Proposed by Diffie & Hellman in 1976**
  - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- **Practical method to exchange a secret key**
- **Used in a number of commercial products**
- **Security relies on difficulty of computing discrete logarithms**
  - Given a prime *p* and a generator *g* of *p*, and a number $X = g^a \bmod p$
  - What is *a*?

Diffie-Hellman Algorithm

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q



Diffie-Hellman Example

- **Have**
  - prime number $q = 353$
  - primitive root $\alpha = 3$
- **A and B each compute their public keys**
  - A, using $X_A = 97$ computes $Y_A = 3^{97} \bmod 353 = 40$
  - B, using $X_B = 233$ computes $Y_B = 3^{233} \bmod 353 = 248$
- **Then exchange and compute secret key:**
  - for A: $K = (Y_B)^{XA} \bmod 353 = 248^{97} \bmod 353 = 160$
  - *for B:* $K = (Y_A)^{XB} \bmod 353 = 40^{233} \bmod 353 = 160$
- **Attacker must solve:**
  - $3^a \bmod 353 = 40$ which is hard
  - Desired answer is 97, then compute key as B does

Active Attack on DH

- **Attacker can intercept, modify, insert, delete messages on the network.**
- **E.g., Man-in-the-Middle attack against DH:**

Trudy

Alice $\xrightarrow{g^{a} \bmod p}$ | $\xrightarrow{g^{a'} \bmod p}$ Bob

Alice $\xleftarrow{g^{b'} \bmod p}$ | $\xleftarrow{g^{b} \bmod p}$ Bob

$K' = g^{ab'} \bmod p$ | $K'' = g^{a'b} \bmod p$

- Trudy can translate messages between Alice & Bob without being noticed and can get two sessions keys for talking to both parties.
- Solution: Digitally sign the messages

"Station-to-Station (STS)" Protocol

- **Authenticated DH protocol; basis for many real-life apps.**
- **Certified PKs are used for signing the public DH parameters. A slightly simplified version:**

Alice $\xrightarrow{x}$ Bob

Alice $\xleftarrow{\text{cert(B)}, y, sig_B(x,y)}$ Bob

Alice $\xrightarrow{\text{cert(A)}, sig_A(x,y)}$ Bob

where $x = g^a \bmod p$, $y = g^b \bmod p$, $k = g^{ab} \bmod p$.

B: Bob's private-key and A: Alice's private-key

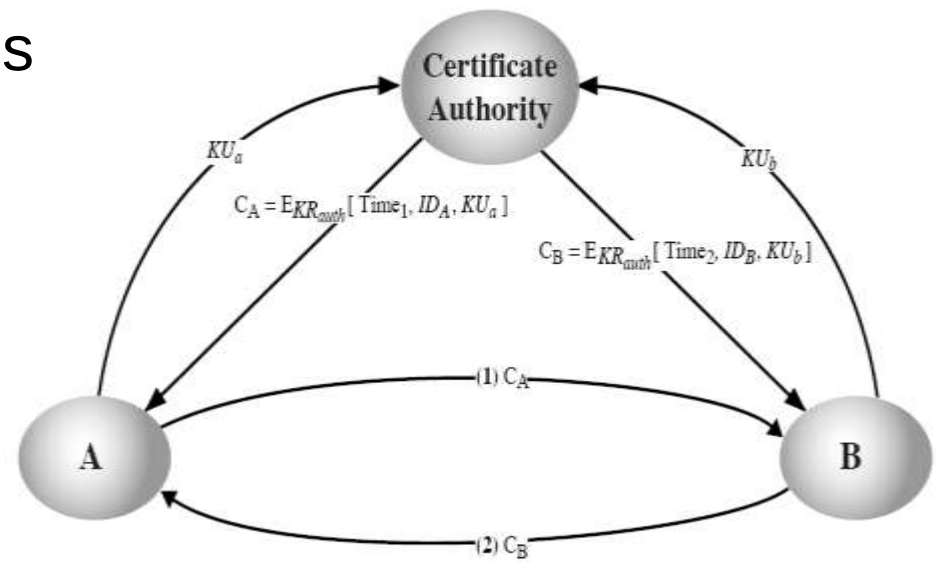
- **STS vs. PKC transport: STS (DH) provides "perfect forward secrecy".**
  - Even if B or A is compromised, the attacker cannot find session key *k*.
  - In PKC transport, if the long-term private-key is compromised, the session keys are also compromised.

Public Key Management

- **How to distribute the public keys?**
  - Binding ID and public-key
- **Solutions:**
  - Public-key Announcement
    - No Authentication
  - Publicly Available Directory
    - The entire directory is published periodically
    - Anyone can access the directory via secure authenticated communication
- **Problem with these:**
  - If Alice does not authenticate Bob's public key when she obtains it, an imposter can send his/her own public key to Alice under Bob's name.
- **Solution:**
  - Public-key Certificates

Certificate

- **Certificate: A document signed by a Certified Authority CA (with the private key), including the ID and the public key of the subject.**
- **CA: A trusted notary that certifies the identity of individuals and their public keys**
  - Everyone registers with the CA, obtains a "certificate" for his/her public key.
  - People obtain each other's certificates thru a repository, a webpage and use the certified public keys in the protocols
  - Will see how this can be implemented in a public-key infrastructure



KDC vs. CA

- **KDC**
  - faster (being based on symmetric keys)
  - has to be online
  - Preferred for LANs
- **CA**
  - Doesn’t have to be online
  - if crashes, doesn’t disable the network
  - much simpler
  - scales better
  - certificates are not disclosure-sensitive
  - a compromised CA can’t decrypt conversations
  - Preferred for WANs (e.g., the Internet).