# Key Distribution and Management

Dr. Natarajan Meghanathan
Associate Professor of Computer Science
Jackson State University
E-mail: natarajan.meghanathan@jsums.edu

# Key Distribution

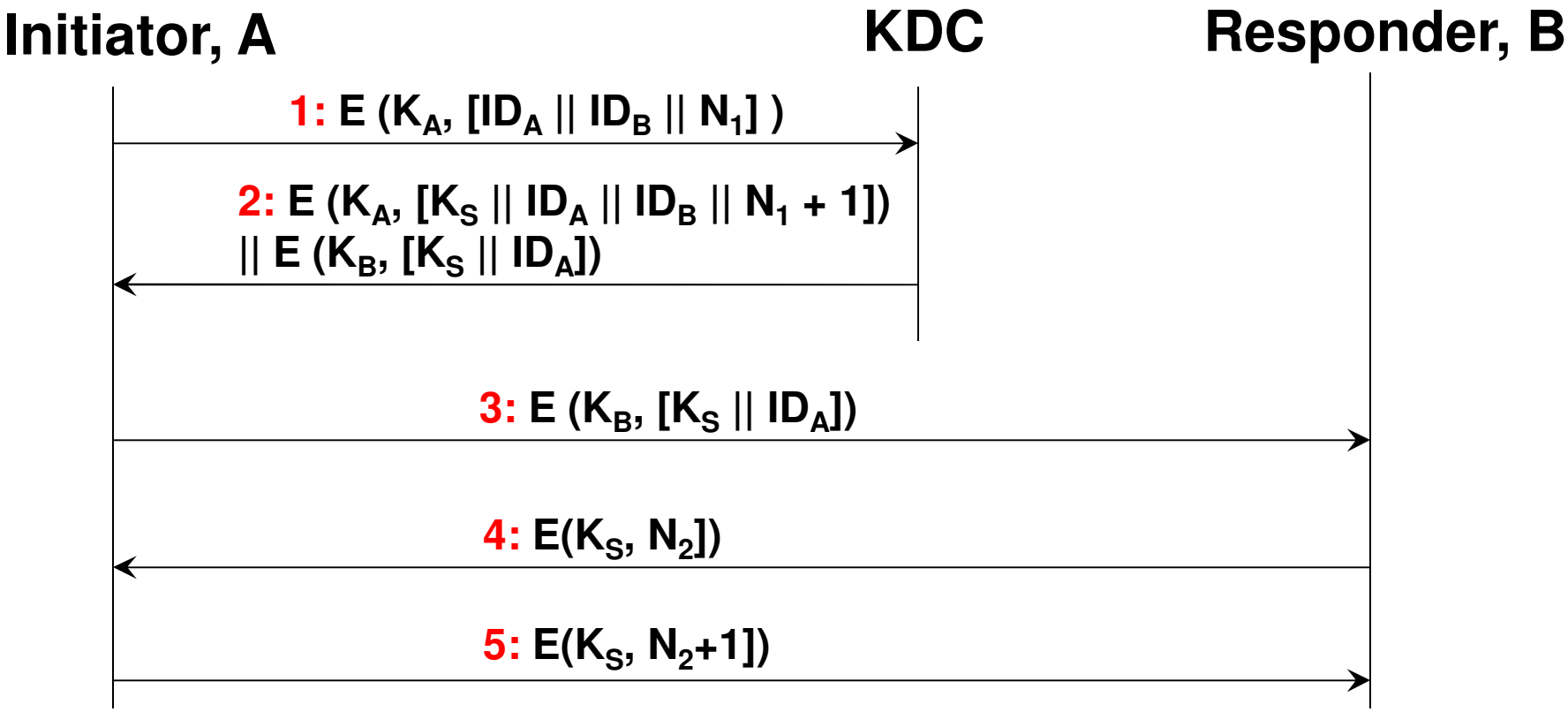
- Both symmetric key schemes and public key schemes require both parties to acquire valid keys.
- In symmetric key schemes, the shared key should be securely distributed between the source and destination, while protecting it from others.

- Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.
- A new session key should be used for each new connection-oriented session. For a connectionless protocol, a new session key is used for a certain fixed period only or for a certain number of transactions.

- On many occasions systems have been broken, not because of a poor encryption algorithm, but because of poor key selection or management.
- Preferred Approach, especially for scalability - A third party, whom all parties trust, can be used as a trusted intermediary to mediate the establishment of secure communications between users.

# Needham-Schroeder Protocol for Secure Key Distribution and Authentication

- We assume a Key Distribution Center (KDC) shares a unique key with each party/ user.




**Note: Steps 1, 2 and 3 are related to "Key Distribution," while steps 3, 4 and 5 are related to providing "Authentication" for the initiator A at the responder B**

# Key Distribution Issues

- For very large networks, a hierarchy of KDCs can be established.
- For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a (hierarchy of) global KDC(s)
- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized (not easy though).

- Hybrid Key Distribution Scheme: This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public key scheme is used to distribute the master keys.
- The addition of a public-key layer provides a secure, efficient means of distributing master keys.

- Distribution of Public Keys: Using Public-Key Authority (centralized approach, needs real-time access to the authority) and Public-Key Certificates (no need for real-time access to the certificate authority).

# Public-Key Certificates

- Motivation:
- Need to verify that the public key advertised for a person actually belongs to that person. Why?
- An evil person C may know a public key-private key pair and advertise the public key as belonging to another entity A.
- Person C may then send a message encrypted using this private key (as if the message comes from A) to another person B.
- B will decrypt the message using the public key of A advertised (by C).
- As a result, B thinks that it is communicating with A, but B is actually communicating with C.

- Each of us adopt a “trust threshold” – a degree to which we are willing to believe an unidentified individual.
- We will use the concept of “vouching for” by a third party as the basis of thrust in settings where two parties do not know about each other.

- Certification Authority (CA): Is an entity that issues digital certificates that contain a public key and the identity of the owner.
- The CA attests that the public key contained in the digital certificate belongs to the person (CA is a sort of digital notary).

# Certificates

- How it works?
- Assume the users of a network have a CA. The users are aware of the public key of the CA.
- The users basically believe something notarized by the CA
- Before communicating with any other user, each user needs to communicate with the CA and obtain a digital certificate for their public key.
- The user sends all its identification information to a registration authority that captures and authenticates the identity of the user.
- The generates a public-key/ private-key pair and then submits a certificate request to the appropriate CA.
- The CA, after getting the identification information authenticated by the registration authority, authenticates the public key submitted by the user, will compute a hash of the identification information and the public key of the user.
- The CA encrypts the identification information, public key and the hash with its private key and sends the encrypted message to the user.
- This encrypted message is now the digital certificate for the user.
- Note that a digital certificate may also contain a start date and stop date, indicating the time period during which the certificate is to be considered valid.

# Certificates

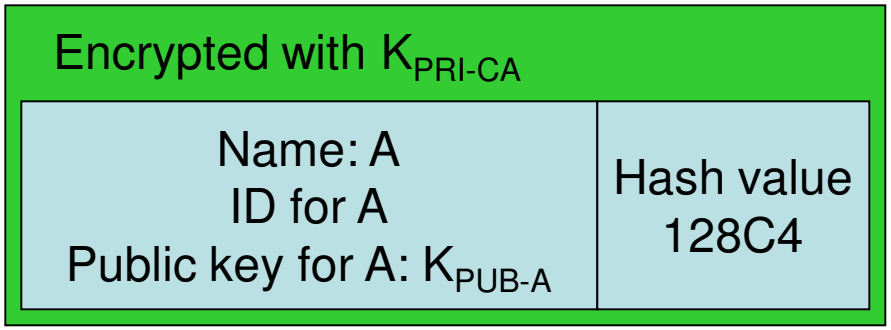
- User A, now wishing to communicate with a user B, will
  - First encrypt the message with the private key of A
  - Second, use the public key of B to further encrypt the above encrypted message and the digital certificate issued by the CA.
- User B will first decrypt the message using its private key and extract the digital certificate issued by the CA for the public key of A.
- User B will decrypt the digital certificate using the public key of the CA and extract the public key of A.
- User B will then decrypt the message sent by A using this extracted public key.

- Note that the private key generated for the user should be stored in a secure key store (created by the application registering for a certificate such as a web browser). The key store would be encrypted based on a password that the user can set.

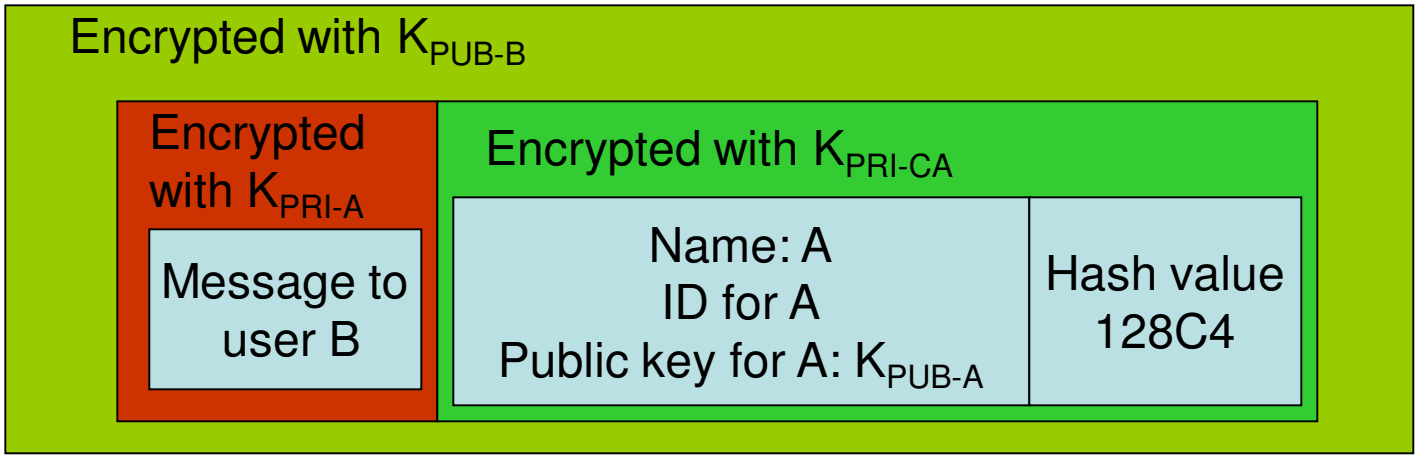
- The above protocol guarantees the following:
  - The message really came from A
  - No body other than B sees the digital certificate for A's public key issued by the CA.

# Certificates

Digital Certificate for the Public Key of A




User A sending to user B




Note: The certificates are created and formatted based on the X.509 standard, which outlines the necessary fields of a certificate and the possible values that can be inserted into these fields. The latest X.509 version is v.3.

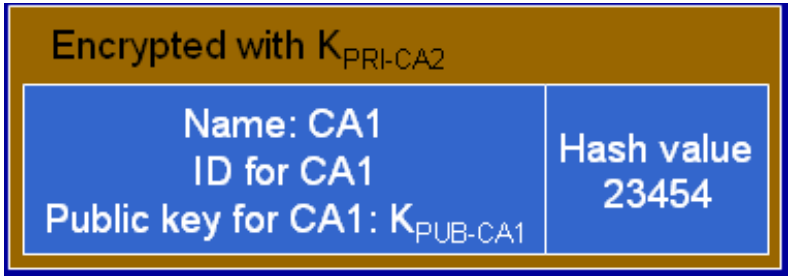
# Certificates

- What if user B is in another network and cannot directly accept the attestation done by the CA of A, and needs another CA to attest the public key of the CA of A?

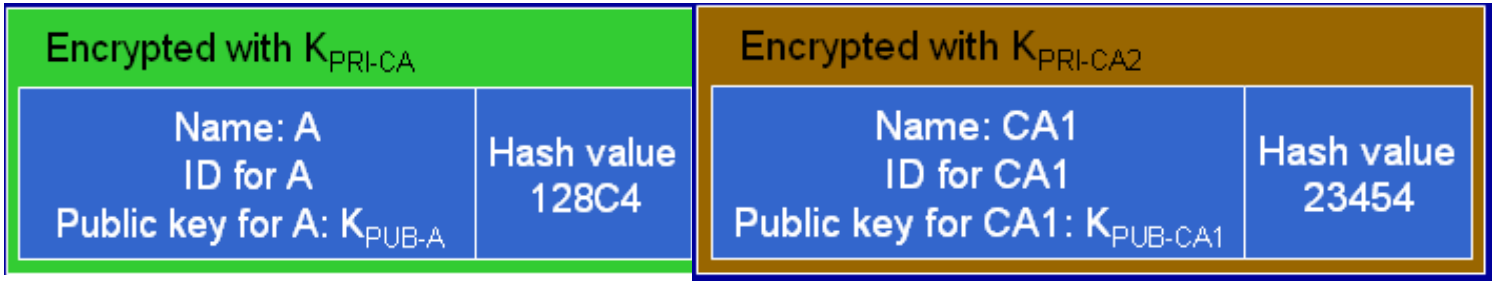
- Let CA1 be the CA that could attest A.
- Let CA2 be the CA that needs to attest CA1 and this attestation would be believed by B.
- Along with the digital certificate issued by CA1 for A, CA1 needs to append the digital certificate it received from CA2 for the public key of CA1
- User B need to send both these digital certificates to B.
- User B will first extract the public key of CA1 from the digital certificate issued by CA2, using the public key of CA2.
- User B will then extract the public key of user A from the digital certificate issued by CA1, using the extracted public key of CA1.
- User B will then use this certified public key of user A to extract the message.

# Certificates

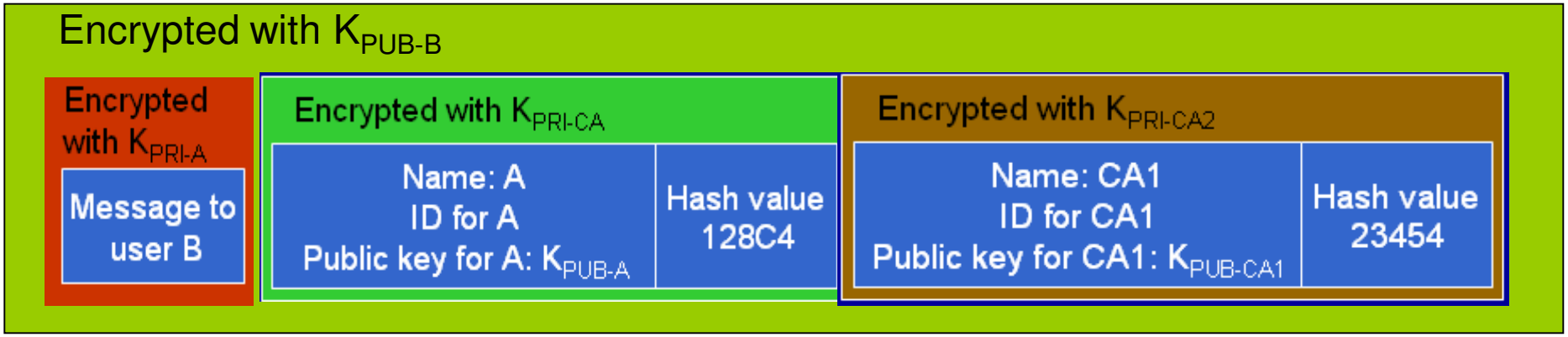
Digital Certificate for the Public Key of CA1




Digital Certificate for the Public Key of A




User A sending to user B



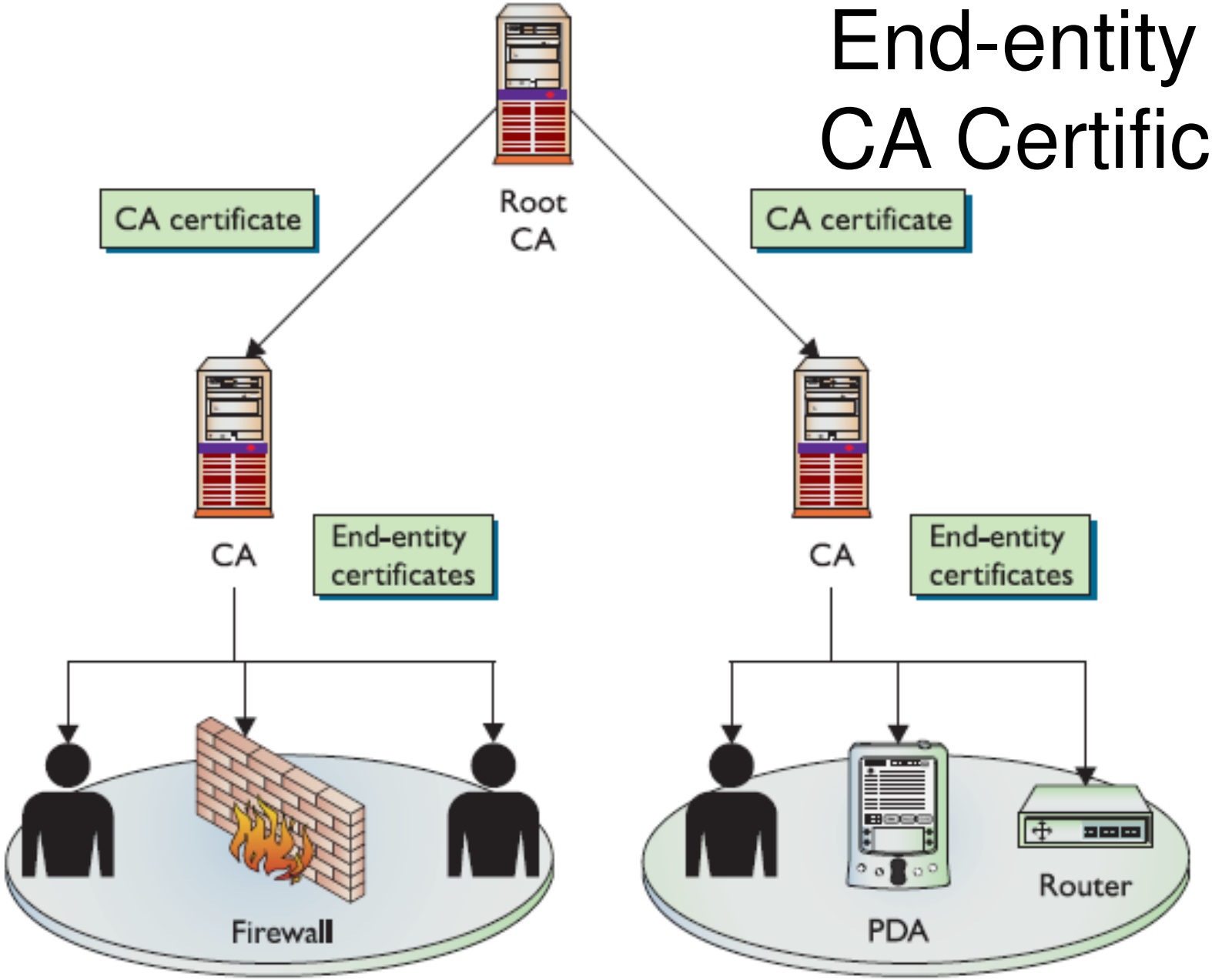
# Certificates

- In general, in addition to issuing the digital certificate for the public key of a user, a CA may send its digital certificate and the digital certificates of all its predecessors in the CA hierarchy.
- Depending upon the authentication level required by the other side of the communication (say user B), it may be then up to user A to only include the relevant CA digital certificates in a message to user B.

# Classes of Digital Certificates

- The types of certificates available can vary between CAs; but, all CAs should at least support the following three classes of certificates:
- Class 1 – A Class 1 certificate is usually used to verify an individual's identity through e-mail. A person who receives a Class 1 certificate can use his public/ private key pair to digitally sign e-mail and encrypt message contents.
- Class 2 – A Class 2 certificate can be used for software signing. A software vendor would register for this type of certificate so that it could digitally sign the software. This provides integrity for the software after it is developed and released, and it allows the receiver software to verify where the software actually came from before installation.
- Class 3 – A Class 3 certificate can be used by a company to set up its own CA which will allow it to carry out its own identification verification and internally generate certificates.
- End-entity certificates (Class 1 and 2 are referred to as End-entity certificates)
- CA certificate – Class 3 can also be referred to as CA certificates.
- Note: An entity can have multiple public/ private key pairs and corresponding digital certificates, used for different purposes.

# End-entity and CA Certificates




Source: Figure 6.8 from Conklin and White – Principles of Computer Security, 2nd Edition

# Kerberos

- Kerberos is used for authentication between intelligent processes, such as client-to-server tasks, or a user's workstation to other hosts in a distributed system.
- Kerberos is based on the idea that a central server provides authenticated tokens called "tickets" to requesting applications.
- A ticket is an unforgeable, non-replayable, authenticated object.
- The security of the protocol relies heavily on the participants maintaining loosely synchronized time.
- Entities involved:
  - Authentication Server (AS)
  - Ticket Granting Server (TGS)
  - Service Server (SS)
  - Ticket Granting Ticket (TGT)
- Brief Idea
  - The client authenticates to AS using a long-term shared secret and receives a ticket from the AS.
  - The client can use this ticket to get additional tickets for SS without resorting to using the shared secret.
  - These tickets are used to prove authentication to the SS.

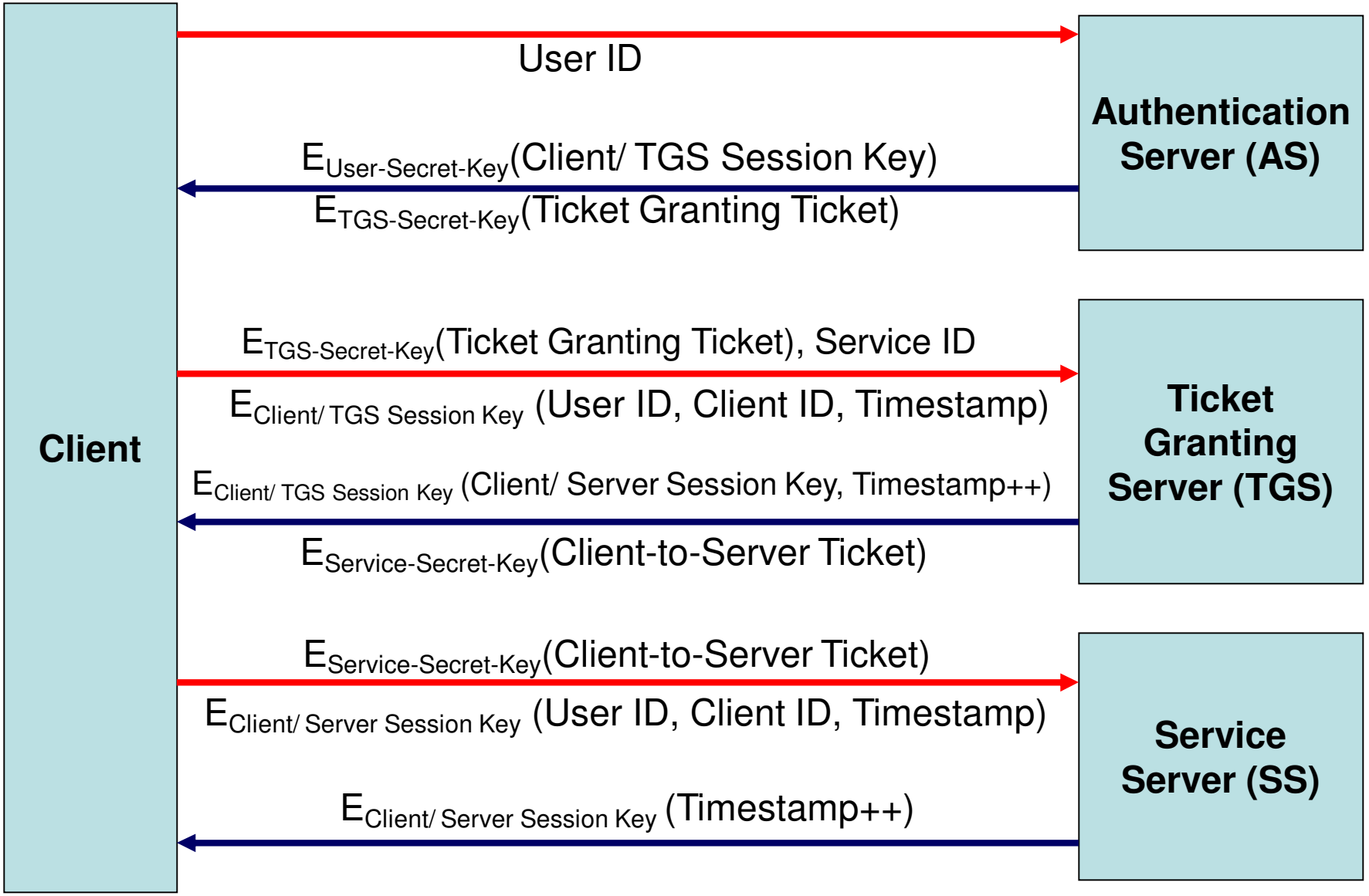
# Kerberos Protocol Steps

- **User Client-based Logon Steps:**
  - A user enters username and password on the client.
  - The client performs a one-way function on the entered password, and this becomes the secret key of the client.
- **Client Authentication Steps:**
  - The client sends a cleartext message containing the user's identity to the Kerberos Authentication Server (AS).
  - The AS checks to see it the client is in its database. If it is, the AS sends back the following two messages to the client:
    - Message A: Client/TGS session key encrypted using the secret key of the user
    - Message B: Ticket-Granting Ticket (which includes the user ID, client network address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS.
  - Once the client receives messages A and B, it decrypts message A with its secret key and extracts the Client/TGS session key
- **Client Service Authorization Steps:** (can be done for each service)
  - The client sends the following two messages to the TGS:
    - Message C: Composed of the TGT from message B and the ID of the requested service.
    - Message D: Authenticator (which is composed of the user ID, client network address and the timestamp), encrypted using the Client/ TGS session key.

# Kerberos Protocol Steps

- Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key. This gives it the Client/TGS session key.
- Using the Client/TGS session key, the TGS decrypts message D and sends the following two messages to the client:
  - Message E: Client-to-Server Ticket (which includes the user ID, client network address, validity period and Client/ Server session key), encrypted using the service's secret key.
  - Message F: Client/ Server session key and timestamp in Message D incremented by 1, encrypted with the Client/ TGS session key.
- Once the client receives message F, it decrypts the message with the Client/ TGS session key and extracts the Client/ Server session key.
- **Client Service Request Steps:**
  - Upon receiving messages E and F from TGS, the client sends to the SS, message E and the following message G: contains the user ID, client network address, timestamp and encrypted with the Client/ Server session key.
  - The SS on receives messages E and G, decrypts message E using its secret key and extracts the Client-to-Server ticket and the Client/ Server session key
  - The SS decrypts message G using the Client/ Server session key, compares the client ID with that in the Client-to-Server ticket, increments the timestamp incremented by 1, encrypts using the Client/ Server session key and sends as message H.
  - The client decrypts message H using the Client/ Server session key and if valid, the client can trust the server and start issuing service requests to the servers, which will be attended by the server.

# Kerberos Protocol Steps



# Kerberos Advantages

- No passwords communicated on the network:
  - A user's password is not sent on the wire (either in clear text or cipher text) during session initiation
- Cryptographic protection against spoofing:
  - Each service access request is mediated by the TGS, which knows the identity of the requested based on the authentication performed by the Kerberos AS, and the fact that the user submitted a service request by encrypting it with the Client/ TGS session key
- Limited validity of the ticket:
  - Long-term attacks that require lots of cryptanalysis cannot be launched
- Timestamps to prevent replay attacks:
  - Kerberos is based on the assumption that the clocks across the client and the servers are synchronized.
  - Each user's request is stamped with a timestamp. A server responds back only if the timestamp is close to the current time at the server.
- Mutual authentication:
  - The user attempts to authenticate the TGS by sending the {user ID, client ID, timestamp} authentication message encrypted with the Client/ TGS.
  - The TGS can decrypt this message only after decrypting the message containing the TGT and the Client/ TGS session key with its secret key.
  - Similar approach is also adopted by the user to authenticate the Service Server.

# Kerberos Weaknesses

- Kerberos requires continuous availability of a trusted ticket-granting server for all access control and authentication.
- Authenticity of servers requires a trusted relationship between the TGS and every service server.
- Kerberos requires timely transactions to reduce the chances of a user with the genuine ticket being denied service
- Password guessing could still work to get the valid secret key for a user.
  - The whole system is still dependent on the user password
- Kerberos does not scale well as the number of service servers is increased, the TGS has to maintain a trustworthy relationship and maintain the secret key for each SS. Adding backup service servers further complicates the situations.
- All applications run by the users in the network need to go through Kerberos authentication. Access to certain services without obtaining Kerberos authentication is not possible.