## Key Management and Distribution

- In real world cryptography, key management is the most difficult vulnerability to protect against, simply because managing and keeping the keys secret is very difficult.
- Cryptanalysts often attack both symmetric and public-key cryptosystems through their key management.
- Key management refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key
- For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others
- Frequent key changes are desirable to limit the amount of data compromised if an attacker learns the key
- Alice and Bob can achieve key distribution in a number of ways:
    1. Alice can select a key and physically deliver it to Bob
    2. A trusted third party, Trent can select the key and physically deliver it to Alice and Bob
    3. If Alice and Bob have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key
    4. If Alice and Bob each have an encrypted connection to a third party Carol, Carol can deliver a key on the encrypted links to Alice and Bob
- A very important security measure is to have **frequent key changes** in order to limit the amount of data compromised if an attacker acquires the key.
- The first two options require manual delivery of keys. Using an encrypted link is reasonably secure since each link encryption device is going to be exchanging data only with its partner on the other end of the link.
- However, for end-to-end encryption over a network, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time.
- Thus, each device needs a number of keys supplied dynamically. The problem is especially difficult in a wide-area distributed system.
- For end-to-end encryption, variations on option 4 have been developed. In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.

- Each user must share a unique key with the key distribution center for purposes of key distribution. The use of a key distribution center is based on the use of a hierarchy of keys, depicted by the following diagram (text):

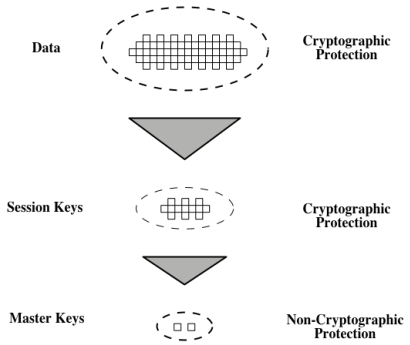


**Figure 14.2 The Use of a Key Hierarchy**

- Communication between end systems is encrypted using a **temporary key**, often referred to as a **session key**.
- Typically, the session key is used for the duration of a logical connection, and then discarded.
- Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication.
- Session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.
- For each end system or user, there is a unique master key that it shares with the key distribution center.
- The master keys must be distributed in some way, however the scale of the problem is vastly reduced.
- If there are N entities that wish to communicate in pairs, then, as many as [N (N - 1)]/2 session keys are needed at any one time.
- However, only N master keys are required, one for each entity. Thus, master keys can be distributed in some non-cryptographic way, such as physical delivery.

- The key distribution concept can be deployed in a number of ways. A typical scenario is illustrated in diagram below, which is based on a diagram in the Popek and Kline paper (figure 1) posted on the course website.

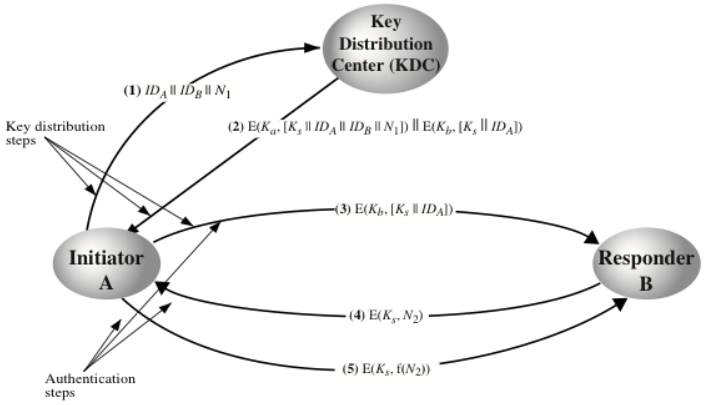


**Figure 14.3 Key Distribution Scenario**

- The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

- Let us assume that Alice wishes to establish a logical connection with Bob and requires a one-time session key to protect the data transmitted over the connection.

- Alice has a master key, $K_a$, known only to herself and the KDC. Bob also shares the master key $K_b$ with the KDC.

- The following sequence illustrates all of the messages and interaction:

    - Alice sends a request to the KDC requesting a connection to Bob ($ID_B$) using a session key. The message includes her identifier ($ID_A$), a unique (nonce) number $N_1$ to identify this particular transaction.

    - The KDC responds with a message encrypted using $K_a$, which includes the following additional items intended for **Alice**:

        - A one-time session key, $K_s$, to be used for this session
        - A copy of the original message including the nonce, to enable Alice to match the received response with the original request.

    - Since Alice is the only one who can decrypt the message, she can authenticate the contents and ensure that the message was not tampered with during transmission from the KDC.

- In addition there are two more items intended for **Bob**:
    - A one-time session key, $K_s$, to be used for this session
    - An identifier for Alice, $ID_A$.
- The last two items for Bob are encrypted using $K_b$, which Bob can decrypt and authenticate Alice's identity.
- Alice then the data sends the data from the KDC intended for Bob to Bob.
- The data includes the one-time session key, $K_s$, as well as Alice's identity ($ID_A$), all encrypted by Bob's secret key ($K_b$).
- Only Bon can decrypt this message and thus discover the new session key and confirm that Alice is the other party, and that all this came from the KDC.
- However Bob does not yet confirm that the message he just received is not a replay of some previous message.
- Using the new session key, Bob sends a nonce $N_2$ to Alice.
- Alice performs some function on $N_2$ and returns the result to Bob, who decrypts the result and thus confirms that there has not been a replay of previous messages.

## Hierarchical Key Control

- For very large networks, limiting the key distribution function to a single KDC it may not be practical.
- As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building. For communication among entities within the same local domain, the local
- If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC.
- In this case, any one of the three KDCs involved can actually select the key.
- The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork.
- A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities.
- Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

## Session Key Lifetime

- The more frequently session keys are exchanged, the more secure they are, because the cryptanalyst has less ciphertext to work with for any given session key.

- On the other hand, the distribution of session keys delays the start of any exchange and places a increases the load on the network.

- For connection-oriented protocols, an easy choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session.

- However, if a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.

- For connectionless protocols, there is no explicit connection initiation or termination. Thus, it is not obvious how often one needs to change the session key.

- The most secure approach is to use a new session key for each exchange. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction.

- A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

## Symmetric Key Distribution using Asymmetric Encryption

- Public-key cryptosystems tend to be very inefficient when encrypting large blocks of data, therefore their use is limited to smaller block transfers.

- However, one of the most important uses of public-key cryptosystems is to encrypt secret keys for distribution.

- The following diagram depicts such a system:

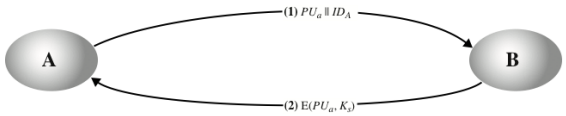


**Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key**

- If Alice wishes to communicate with Bob, the following exchange takes place:

    - Alice generates a public/private key pair ($PU_A$, $PR_A$) and sends a message to Bob consisting of $PU_A$, and an identifier, $ID_A$.
    - Bob generates a secret key $K_s$, and transmits a message to Alice, encrypted with her public key, $PU_A$.
    - Alice decrypts Bob's message $D(PR_A, E(PU_A, K_s))$ to recover the secret key. Since only Alice can decrypt the message, only Alice and Bob know $K_s$.
    - Alice and Bob can then discard $PU_A$, $PR_A$ and $PU_B$ respectively.

- The protocol depicted above is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message using an attack known as a man-in-the-middle attack:

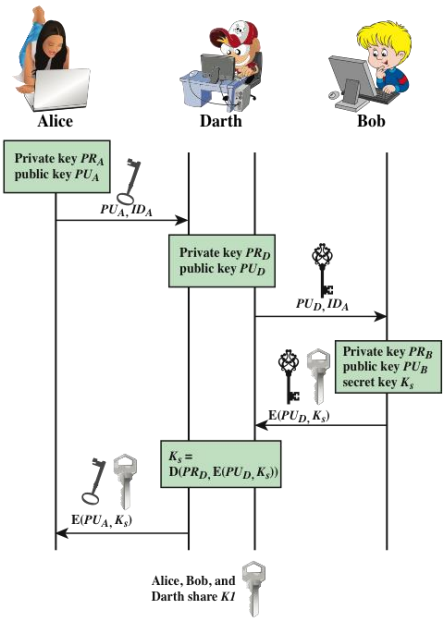


**Figure 14.8 Another Man-in-the-Middle Attack**

- if an adversary, Darth, has control of the intervening communication channel, then D can compromise the communication in the following fashion without being detected:
    - Alice generates a public/private key pair ($\boldsymbol{PU_A}$, $\boldsymbol{PR_A}$) and sends a message to Bob consisting of $\boldsymbol{PU_A}$, and an identifier, $\boldsymbol{ID_A}$.
    - Darth intercepts and creates his own public/private key pair ($\boldsymbol{PU_D}$, $\boldsymbol{PR_D}$) and sends a message to Bob consisting of $\boldsymbol{PU_D}$, and an identifier, $\boldsymbol{ID_A}$.
    - Bob generates a secret key $\boldsymbol{K_s}$, and transmits a message to Alice, encrypted with the malicious public key, $\boldsymbol{PU_D}$.
    - Darth intercepts and acquires the secret key $\boldsymbol{K_s}$ by decrypting $\boldsymbol{D(PR_D, E(PU_D, K_s))}$
    - Darth sends $\boldsymbol{E(PU_A, K_s)}$ to Alice.
- The result is that both Alice and Bob know $\boldsymbol{K_s}$, and are unaware that Darth also know the secret key, and can simply eavesdrop on their message exchanges.

## Distribution of Public Keys

- Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the general schemes described below.

### Public Announcement (Uncontrolled) of Public Keys

- The whole point of public-key encryption is that if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large:


Figure 14.10 Uncontrolled Public Key Distribution

- PGP (pretty good privacy), is a very good example of a cipher which makes use of RSA. Many PGP users have adopted the practice of appending their public key to messages that they send to public forums.

- Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user Alice and send a public key to another participant or broadcast such a public key.

- Until such time as Alice discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for Alice and can use the forged keys for authentication.

**Publicly available Directory**

- A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys:

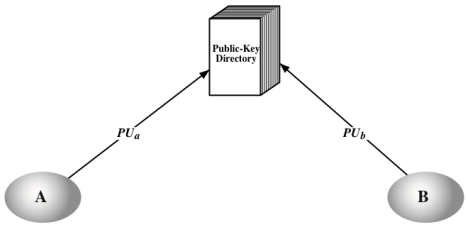


**Figure 14.11 Public Key Publication**

- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization as shown above.

- Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

- This scheme is more secure than individual public announcements but still has vulnerabilities.

- If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant.

- Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

**Public-Key Authority**

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.

- The following is a typical scenario based on a figure in the Popek and Kline paper:

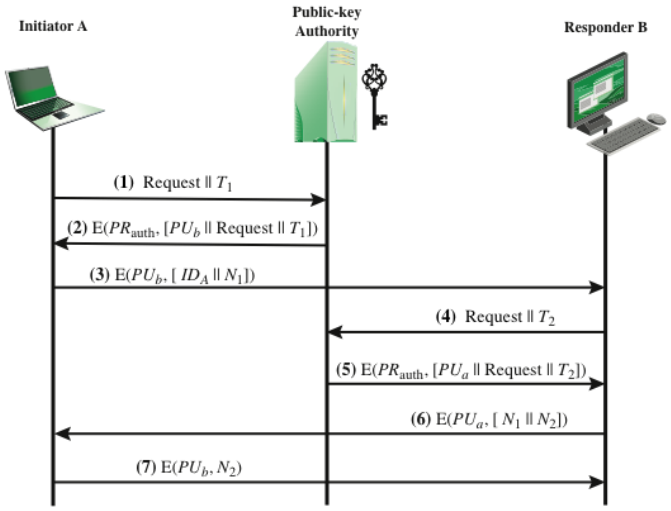


**Figure 14.12 Public-Key Distribution Scenario**

- The central Public-Key authority maintains a dynamic directory of public keys of all participants.

- In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

- A total of seven messages are required:

    1. Alice begins by sending a time-stamped message to the public-key authority requesting Bob's public key.
    2. The public-key authority sends Alice Bob's public key, a copy of the original request, and the time stamp, all encrypted using the authority's private key $PR_{AUTH}$. Alice can decrypt this message using the public key of the authority and is thus also sure of the source of the message. The message includes the following:

        a. Bob's public key, $PU_B$ which Alice can use to encrypt message to Bob.
        b. The original timestamp so Alice can guarantee that this is not an old message from the authority containing a key other than B's current public key
        c. A copy of the original to allow Alice to verify that her original request was not altered.

    3. Alice can now send messages to Bob because she knows Bob's public key. However, to identify herself, as well as to prevent a replay of previous transmissions, Alive now sends an identifier ($ID_A$) and a nonce ($N_1$) to Bob, encrypted with Bob's public key.
    4. Bob now performs the first two steps above with the public-key authority to retrieve Alice's public key.
    5. Then Bob sends Alice the identifier just received, and the nonce ($N_1$), as well as a new nonce ($N_2$) all encrypted with Alice's public key ($PU_A$ ).
    6. Alice can decrypt that message and is now sure that she is talking to Bob.
    7. Alice must now send back $N_2$, encrypted with Bob's public key ($PU_B$) to Bob so that Bob can be sure he is talking to Alice.

- The above protocol contains seven messages; however the initial five messages can be dispensed with by local caching of public keys.

- As a security measure as well as to ensure currency, each user should periodically request fresh copies of public keys.

**Public-Key Certificates**

- The previous has several drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact.

- As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

- An alternative approach is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority.

- In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.

- Typically, the third party is a certificate authority, such as a government agency or a financial institution that is trusted by the user community.

- A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate.

- Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.

- A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.

- This scheme has the following requirements:

    1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
    2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
    3. Only the certificate authority can create and update certificates.

- Subsequent to the original requirements above, the following additional requirement was added:

    4. Any participant can verify the currency of the certificate.

- The following diagram illustrates a certificate scheme described above:

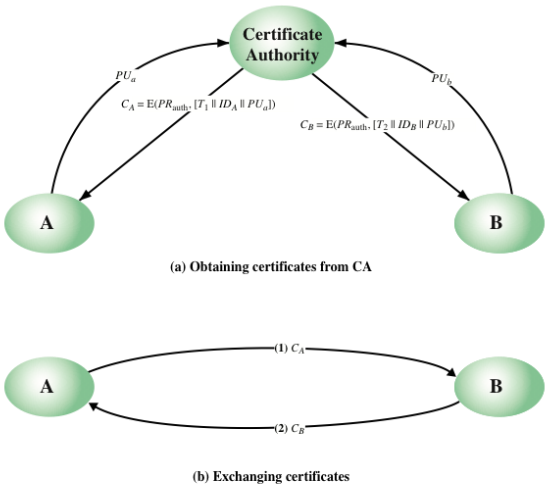


**Figure 14.13 Exchange of Public-Key Certificates**

- Each participant applies to the certificate authority to obtain the necessary certificates by supplying a public key and using some form of secure authenticated communication.

- **Reading Assignment: Read the section (14.4) on X.509 Certificates in your textbook.**