Chapter 9 KEY MANAGEMENT

- **Key Management**
  - **Distribution of cryptographic keys**
  - **Mechanisms for binding an identity to a key**
  - **Generation, maintenance, and revoking of keys**
- **Notation**
  - $X \rightarrow Y$: $\{Z\}_k$ **means that** X **sends** Y **a message** Z **enciphered with key** k.
  - k_{Alice} **and** k_{Bob} **refer to keys belonging to Alice and Bob.**
  - **If Alice and Bob share a key, the key is** $k_{Alice,Bob}$.
  - k **represents a <u>symmetric</u> key.**
  - e **represents a <u>public</u> key and** d **represents a <u>private</u> key.**
  - **The operator || means that if** a **and** b **are the operands,** $a||b$ **is the <u>concatenation</u> of** a **and** b.

Chapter 9 KEY MANAGEMENT

9.1 Session and Interchange Keys

- A <u>interchange</u> key is a cryptographic key associated with a principal to a communication.
  - Alice has a cryptographic key used to exchange information with Bob.
  - This key does not need to change over interactions with Bob.
- A <u>session</u> key is a cryptographic key associated with the communication itself.
  - If Alice communicates twice with Bob, she should not use the same key to encipher messages.
  - Hence, Alice needs to generate a key for the session in question.
  - The session key enciphers data only. It does not authenticate either principal.
- EXAMPLE:
  - Alice is a client of Bob's stockbrokering firm.
  - She needs to send Bob one of 2 messages: BUY or SELL.
  - The attacker (Cathy) enciphers both messages with Bob's public key.
  - When Alice sends her message to Bob, Cathy compares it with her two messages to check which one it matches.

Chapter 9 KEY MANAGEMENT

9.2 Key Exchange

- The goal of key exchange is to enable secret communication between Alice and Bob.
- Solutions to this problem must <u>meet</u> the following criteria:
  - The key that Alice and Bob are to share cannot be transmitted in the clear.
  - This key could either be enciphered or derived after exchanging some data.
  - Alice and Bob may decide to trust a 3rd party (Cathy).
  - The cryptosystems and protocols are publicly known. The only secret data is the cryptographic keys.

Chapter 9 KEY MANAGEMENT

- **Classical Cryptographic Key Exchange and Authentication**
  - **Suppose Alice and Bob wish to communicate. How do they agree on a common key?**
  - **If Alice sends the key in the clear, an eavesdropper (Eve) will see it.**
  - **To avoid this <u>bootstrapping</u> problem, classical protocols rely on a trusted 3rd party, Cathy.**
    - **Alice and Cathy share a secret key.**
    - **Bob and Cathy share a different secret key.**
  - **The following simple protocol provides a starting point:**
    1. **Alice $\rightarrow$ Cathy: {request for session key to Bob}k_{Alice}**
    2. **Cathy $\rightarrow$ Alice: $\{k_{session}\}\ k_{Alice} \| \{k_{session}\}\ k_{Bob}$**
    3. **Alice $\rightarrow$ Bob: $\{k_{session}\}\ k_{Bob}$**
    4. **Now Bob deciphers the message from Alice and uses $k_{session}$ to communicate with Alice.**
  - **This protocol is the <u>basis</u> for many more sophisticated protocols. However, Bob does not know to whom he is talking.**

Chapter 9 KEY MANAGEMENT

– **EXAMPLE:**
  - **Alice sends Bob the message "Deposit $500 in Dan's bank account today" enciphered under $k_{session}$.**
  - **If Eve records the message "$\{k_{session}\}\ k_{Alice} \| \{k_{session}\}\ k_{Bob}$," and the message**

    {Deposit $500 in Dan's bank account today}$_{ksession}$,

    she can send Bob the message $\{k_{session}\}\ k_{Bob}$ followed by

    {Deposit $500 in Dan's bank account today}$_{ksession}$.
  - **Bob will not know who is sending the message.**
  - **Avoiding problems such as this replay attack adds considerable complexity.**
  - **Key exchange protocols typically add, at a minimum, some sort of authentication and defense against <u>replay</u> attack.**
  - **One of the best known protocols is the <u>Needham-Schroeder</u> protocol.**

Chapter 9 KEY MANAGEMENT

Needham-Schroeder protocol

1. Alice $\xrightarrow{\text{Alice} \,||\, \text{Bob} \,||\, r_1}$ Cathy

   $r1$ and $r2$ are two random numbers, and they cannot repeat between different protocols.

2. Alice $\xleftarrow{\{\, \text{Alice} \,||\, \text{Bob} \,||\, r_1 \,||\, k_s \,||\, \{\, \text{Alice} \,||\, k_s \,\}\, k_B \,\}\, k_A}$ Cathy

   Alice deciphers this message from Cathy, and knows that Alice, Bob, and r_1 are present. r_1 is in both the first and second messages. It is an indication that Cathy must have sent the second message.

3. Alice $\xrightarrow{\{\, \text{Alice} \,||\, k_s \,\}\, k_B}$ Bob

   When Bob deciphers this message, he sees Alice's name. The message was enciphered using a key Bob shares with Cathy. $\therefore$ the message was enciphered by Cathy.

4. Alice $\xleftarrow{\{\, r_2 \,\}\, k_s}$ Bob

   Bob sends a random number enciphered by the session key. If Eve intercepts the message, she would not know what to return to Bob.

5. Alice $\xrightarrow{\{\, r_2 - 1 \,\}\, k_s}$ Bob

   Since Alice has initiated the communication, she is able to decipher the message sent by Bob. Alice then applies a fixed function to r_2 (in this case, it is r_2-1). Bob will understand that he is talking to Alice.

Chapter 9 KEY MANAGEMENT

- **Public Key Cryptographic Key Exchange and Authentication**
  - **Public key cryptography makes key exchange very easy.**
    - **Alice → Bob: $\{k_{session}\}e_{Bob}$, where e_{Bob} is Bob's public key.**
  - **Now, Alice and Bob can communicate securely using the session key.**
  - **Eve can forge such a message, and Bob does not know who the message comes from.**
  - **One obvious fix is for Alice to sign the session key:**
    - **Alice → Bob: $\{\{k_{session}\}d_{Alice}\}e_{Bob}$, where d_{Alice} is Alice's private key.**
  - **These protocols assume that Alice has a copy of Bob's public key. If not, she must get it from a public server, Peter.**
  - **If Eve manages to get Bob's public key, she can read any traffic between Alice and Bob:**
    - **Alice → Peter: {send me Bob's public key}, which is intercepted by Eve.**
    - **Eve → Peter: {send me Bob's public key}**
    - **Peter → Eve: e_{Bob}**
    - **Eve → Alice: e_{Eve}**
    - **Alice → Bob: $\{k_{session}\}e_{Eve}$, which is intercepted by Eve.**
    - **Eve → Bob: $\{k_{session}\}e_{Bob}$**
  - **This is called a "<u>man-in-the-middle-attack.</u>"**
  - **Hence, there must be a <u>binding</u> of identity to a public key!**

Chapter 9 KEY MANAGEMENT

9.3 Cryptographic Key Infrastructures

- Classical cryptosystems use shared keys, and it is not possible to bind an identity to a symmetric key.
- Public key cryptosystems use a pair of keys for each communicating party.
  - The association between the cryptographic key and the principle is critical.
  - It determines the public key used to encipher messages for secrecy.
- The question is "How is it possible to bind the public key to a principal to represent her identity?"
- The originator cannot sign the public key with her private key!
  - This pushes the problem to another level.
  - The recipient would know that whoever generated the public key also signed it.
  - No identity is present!
- Kohnfelder's suggestion is to create a message containing the following: $C_{Alice} = \{e_{Alice} \| \text{Alice} \| T\}d_{Cathy}$.
  - A representation of identity.
  - The corresponding public key.
  - A time stamp.
  - A trusted authority's signature on the message.

- A certificate is a token that binds an identity to a cryptographic key.

Chapter 9 KEY MANAGEMENT

- **Certificate Signature Chains**
  - **How does a user <u>validate</u> a certificate? He must obtain the issuer's public key.**
  - **If the issuer has a <u>certificate</u>, the user can get the public key from the issuer's certificate.**
  - **This pushes the problem to another level: How can the issuer's certificate be <u>validated</u>?**
  - **One major approach to this problem is to construct a <u>tree-like hierarchy</u>.**
  - **<u>X.509 certificate</u> provides a solution.**
    - **Defines certificate formats and certificate validation in a generic context.**
    - **Originally issued in 1988.**
    - **I'Anson and Mitchell found problems with the protocols and the certificate structure.**
    - **These problems were corrected in the 1993 version, X.509v3.**

Chapter 9 KEY MANAGEMENT

The X.509v3 certificate has the following components:

1. **Version: Each successive version of the certificate has new fields added.**
2. **Serial number: Must be unique among the certificates issued by the issuer.**
3. **Signature Algorithm Identifier: Identifies the algorithm, and any parameters, used to sign the certificate.**
4. **Issuer's Distinguished Name: A name that uniquely identifies the issuer.**
5. **Validity Interval: Gives the time at which the certificate becomes valid and expires.**
6. **Subject's Distinguished Name: A name that uniquely identifies the subject to whom the certificate is issued.**
7. **Subject's Public Key Information: Identifies the algorithm, its parameters, and the subject's public key.**
8. **Issuer's Unique Identifier: Allows the issuer to disambiguate among entities with the same name.**
9. **Subject's Unique Identifier: Similar to field 8.**
10. **Extensions: Defines certain extensions.**
11. **Signature: Identifies the algorithm and parameters used to sign the certificate, followed by the signature (an enciphered hash of fields 1 to 10).**

Chapter 9 KEY MANAGEMENT

- A <u>Certificate Authority (CA)</u> is an entity that issues certificates.
  - If all certificates have a common issuer, the issuer's public key can be distributed out of band. This is not <u>feasible</u>!
  - There is a need for <u>multiple</u> issuers.
  - Suppose Alice has a certificate from her local CA, Cathy.
  - Alice wants to communicate with Bob, whose local CA is Dan.
  - The problem for Alice and Bob is to <u>validate</u> each other's certificates.
  - Certificates:

Cathy<<Alice>>:	Alice's certificate signed by Cathy.
Dan<<Bob>:	Bob's certificate signed by Dan.
Cathy<<Dan>>:	Dan's certificate signed by Cathy.
Dan<<Cathy>>:	Cathy's certificate signed by Dan.

  - Dan and Cathy are said to be <u>cross-certified</u>.
  - Alice forms the signature chain: Cathy<<Dan>>Dan<<Bob>.
  - Bon forms the signature chain: Dan<<Cathy>>Cathy<<Alice>>.
  - Signature chains could be of arbitrary length.
  - The only requirement is that each certificate can be validated by the one before it in the chain.

Chapter 9 KEY MANAGEMENT

9.4 Storing and Revoking Keys

- Key Storage
  - Protecting cryptographic keys sounds simple: Just put the key into a file and use OS access control mechanisms to protect it.
  - Unfortunately, OS access control mechanisms can often be evaded or defeated, or may not apply to some users.
  - On a networked system:
    - An attacker could trick the owner into downloading a program that would send keystrokes and files to the attacker.
    - If the file is enciphered, the key will be resident in memory at some point.
  - A feasible solution is to put the key onto one or more physical devices such as a special terminal, ROM, or a smartcard.
    - The key never enters the computer's memory.
    - To encipher a message, the user inserts a smartcard into a special device that can read from, and write to, a computer.
      - » The computer sends to the smartcard the message to be protected.
      - » The smartcard uses the key to encipher the message.
      - » Finally, the message is sent to the computer.
      - » Hence, at no point is the cryptographic key exposed.

Chapter 9 KEY MANAGEMENT

9.4 Storing and Revoking Keys

- Key Revocation
  - Certificate formats contain a key <u>expiration</u> date.
  - If a key becomes <u>invalid</u> before the expiration date, it must be revoked.
  - This means that the key is either compromised or the binding between the subject and the key has changed.
  - There are <u>two</u> problems with revoking a public key:
    - The first is to ensure that revocation is correct (i.e., the entity revoking the key is authorized to do so).
    - The second is to ensure timeliness of the revocation throughout the infrastructure.
      - » This problem depends on reliable and highly connected servers.
      - » Ideally, notice of revocation will be sent to all parties, but invariably there will be a time lag.
  - A <u>certificate revocation list</u> (CRL) is a list of certificates that are no longer valid.
    - A CRL contains:
      - » The serial numbers of the revoked certificates and the dates on which they were revoked
      - » The name of the issuer
      - » The date on which the list was issued
      - » When the next list is expected to be issued
    - The issuer also signs the list.

Chapter 9 KEY MANAGEMENT

9.5 Digital Signatures

- As <u>e-commerce</u> grows, so does the need for a provably high degree of authentication.
- A <u>digital signature</u> is a construct that authenticates both the origin and contents of a message in a manner provable to a <u>disinterested third party</u>.
- Classical Signatures
  - All classical signature schemes rely on a trusted third party.
  - The judge must trust the third party.
  - Let Cathy be the trusted third party.
  - Alice shares a cryptographic key with Cathy: k_{Alice}
  - Bob shares a cryptographic key with Cathy: k_{Bob}
  - The communication between Alice, Bob, and Cathy is as follows:

$$\text{Alice} \xrightarrow{\{m\}k_{Alice}} \text{Bob}$$

$$\text{Cathy} \xleftarrow{\{m\}k_{Alice}} \text{Bob}$$

$$\text{Cathy} \xrightarrow{\{m\}k_{Bob}} \text{Bob}$$

Chapter 9 KEY MANAGEMENT

9.5 Digital Signatures

– **Public Key Signatures**

- **A digital signature scheme based on the <u>RSA system</u>.**
- **Suppose Alice wants to trick Bob into signing a message m.**
- **She computes two other messages m_1 and m_2 such that $m_1 m_2 \bmod n_{Bob} = m$.**
- **She has Bob sign m_1 and m_2.**
- **Alice multiplies the two signatures together and reduces mod n_{Bob}.**
- **She now has Bob's signature on m.**
- **The defense is <u>not</u> to sign random documents (which is obvious!), and <u>not</u> to sign the document itself. A cryptographic <u>hash</u> of the document should be signed!**
- **EXAMPLE:**
  - **Alice and Bob are communicating.**
  - **$n_{Alice} = 95$, $e_{Alice} = 59$, $d_{Alice} = 11$**
  - **$n_{Bob} = 77$, $e_{Bob} = 53$, $d_{Bob} = 17$**
  - **26 contracts, numbered 00 to 25**
  - **Alice has Bob sign 05 and 17:**
    - » **$c = m^{dBob} \bmod n_{Bob} = 05^{17} \bmod 77 = 3$**
    - » **$c = m^{dBBob} \bmod n_{Bob} = 17^{17} \bmod 77 = 19$**
  - **Alice computes $05 \times 17 \bmod 77 = 08$; the corresponding signature is $03 \times 19 \bmod 77 = 57$. She claims Bob signed 08.**
  - **The judge computes $c^{eBob} \bmod n_{Bob} = 57^{53} \bmod 77 = 08$. Alice has tricked Bob!**

Chapter 9 KEY MANAGEMENT

– **A second problem demonstrates that messages that are both enciphered and signed should be signed first!**

- **Suppose Alice is sending Bob her signature on a confidential contract *m*.**
- **She enciphered it first, then signs it, and sends the result to Bob:**

$$c = (m^{e_{Bob}} \bmod n_{Bob})^{d_{Alice}} \bmod n_{Alice}$$

- **Bob wants to claim that Alice sent him the contract *M*.**
- **Bob computes a number *r* such that M^r mod n_{Bob}= *m.***
- **He then publishes his public key as (re_{Bob}, n_{Bob}).**
- **Note that the modulus does not change.**
- **Now Bob claims that Alice sent him *M*.**
- **The judge verifies this using Bob's current public key.**
- **The simplest way to fix this problem is to require all users to use the <u>same exponent</u> but vary the <u>moduli</u>.**

Chapter 9 KEY MANAGEMENT

- **EXAMPLE:**
  - **Bob seeks revenge!**
  - **Bob and Alice agree to sign the contract *G*(06).**
  - **Alice first enciphers it, then signs it, and sends it to Bob:**

    $$(06^{53} \bmod 77)^{11} \bmod 95 = 63$$
  - **Bob wants the contract to be *N*(13).**
  - **He computes an *r* such that $13^r \bmod 77 = 6$. One such *r* is 59.**
  - **Bob then computes a new public key:**

    $$re_{Bob} \bmod \varnothing(n_{Bob}) = 59 \times 53 \bmod 60 = 7$$
  - **He replaces his current public key with (7,77), and resets his private key to 43.**
  - **Bob now claims that Alice sent him contract *N*, signed by her.**
  - **The judge is called, and she computes:**

    $$(63^{59} \bmod 95)^{43} \bmod 77 = 13$$
  - **The conclusion is that Bob is correct!**
  - **This attack will not work if one <u>signs</u> first and then <u>enciphers</u>.**