Chapter 11: Security

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

١

Introduction

- **History**
- **The emergence of cryptography into the public domain**
  - Public-key cryptography
  - Much stronger DES
  - Protagonist in security (principles)
- **Security policies**
  - Provide for the sharing resource within limited rights
- **Security mechanisms**
  - Implement security policies

٢

Historical context: the evolution of security needs

	1965-75	1975-89	1990-99	Current
Platforms	Multi-user timesharing computers	Distributed systems based on local networks	The Internet, wide-area services	The Internet + mobile devices
Shared resources	Memory, files	Local services (e.g. NFS), local networks	Email, web sites, Internet commerce	Distributed objects, mobile code
Security requirements	User identification and authentication	Protection of services	Strong security for commercial transactions	Access control for individual objects, secure mobile code
Security management environment	Single authority, single authorization database (e.g. /etc/ passwd)	Single authority, delegation, repli- cated authorization databases (e.g. NIS)	Many authorities, no network-wide authorities	Per-activity authorities, groups with shared responsibilities

Cryptography notations

Alice	First participant
Bob	Second participant
Carol	Participant in three- and four-party protocols
Dave	Participant in four-party protocols
Eve	Eavesdropper
Mallory	Malicious attacker
Sara	A server

Threats and attacks

- **Security threats**
  - *Leakage*: acquisition of information by unauthorized recipients
  - *Tampering*: unauthorized alteration of information
  - *Vandalism*: interference with the proper operation of a system without gain to the perpetrator
- **Methods of attack (dangers in theory)**
  - *Eavesdropping*: obtain copies of messages without authority
  - *Masquerading*: send or receive messages using the identity of another principal without their authority
  - *Message tampering*: intercept messages and alter their contents before pass them on to the intended recipient
  - *Replaying*: store intercepted messages and send them at a later data
  - *Denial of service*: flood a channel or other resources with messages in order to deny access for others

Threats and attacks

- **Attacks in practice**
  - discover loopholes
    - E.g. Guess password

Threats from mobile code

- **Sandbox model in Java**
  - Each environment has a security manager that determines which resources are available to the application
    - most applets can not access local files, printers or network sockets
  - The JVM takes two further measures to protect the local environment :
    - The downloaded classes are stored separately from the local classes, preventing them from replacing local classes with spurious versions
    - The bytecodes are checked for validity,
      - e.g. avoiding accessing illegal memory address

Information Leackage

- If transmission of a message between two processes can be observed, some information can be gleaned from its mere existence.
  - E.g. flood of messages to a dealer in a particular stock might indicate a high level of trading in that stock.

Securing electronic transactions

- **Examples depending crucially on security**
  - Email, purchase of goods and services, banking transactions, micro-transactions

Securing electronic transactions

Requirements for securing web purchases

- Authenticate the vendor to the buyer
- Keep the buyer’s credit number and other payment details from falling into others’ hands and ensure that they are unaltered from the buyer to vendor
- Ensure downloadable contents are delivered without alteration and disclosure
- Authenticate the identity of the account holder to the bank before giving them access to their account
- Ensure account holder can’t deny they participated in a transaction (non-repudiation)

Designing secure systems

- **The analogy between designing secure systems and producing bug-free programs.**
- **Construct a list of threats**, and show that each of them is prevented by the mechanisms employed
  - By informal argument, or logical proof
- **Auditing methods**
  - Secure log: record security-sensitive system actions with details of the actions performed and their authority
- **Balance cost and inconvenience**
  - Cost in computational effort and in network usage
  - Inappropriately specified security measures may exclude legitimate users from performing necessary actions

Worst-case assumptions and design guidelines

- **Interfaces are exposed**
- **Networks are insecure**
- **Limit the lifetime and scope of each secret**
- **Algorithms and program code are available to attackers**
  - Publish the algorithms used for encryption and authentication, relying only on the secrecy of cryptographic keys
- **Attackers may have access to large resources**
- **Minimize the trusted base**
  - Trusted computing base: the portion of a system that are responsible for the implementation of its security, and all the hardware and software components upon which they rely

Chapter 11:Overview of security techniques

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

١٣

Overview of security techniques

1. *Cryptography :* is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication.
2. *Cryptanalysis :* is the study of mathematical techniques for attempting to defeat cryptographic techniques, and, more generally, information security services.

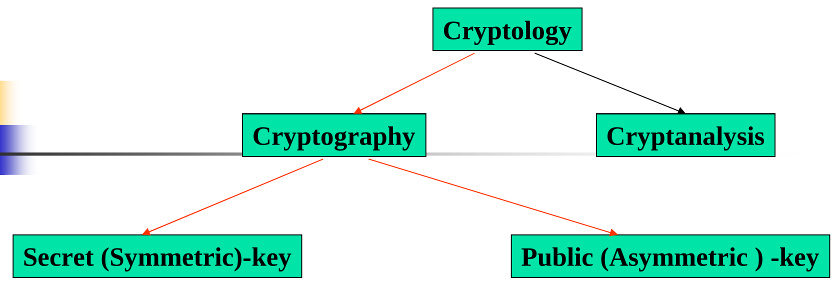
A *cryptanalyst* is someone who engages in cryptanalysis.

3. *Cryptology :* is the study of cryptography and cryptanalysis.

A *cryptosystem* is a general term referring to a set of cryptographic primitives used to provide information security services. Most often the term is used in conjunction with primitives providing confidentiality, i.e., encryption.

4. **Cryptographic** techniques are typically divided into two generic types: *symmetric-key* and *public-key.*

١٤



- Confidentiality
- Data integrity
- Entity authentication
- Non-repudiable (data origin authentication)

١٥

Cryptography

- **Encryption**
  - the process of encoding a message in such a way as to hide its contents
- **Cryptographic key**
  - a parameter used in an encryption algorithm in such a way that the encryption can not be reversed without a knowledge of the key

١٦

Cryptography

- **Shared secret keys**
  - The sender and the recipient must share a knowledge of the key and it must not be revealed to anyone else
- **Public/private key pairs**
  - The sender of a message uses a *public key* – one that has already been published by the recipient – to encrypt the messages; the recipient uses a corresponding *private key* to decrypt the message.

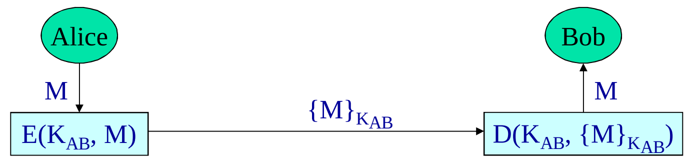
Cryptography notations

K_A	Alice's secret key
K_B	Bob's secret key
K_{AB}	Secret key shared between Alice and Bob
K_{Apriv}	Alice's private key (known only to Alice)
K_{Apub}	Alice's public key (published by Alice for all to read)
$\{M\}_K$	Message M encrypted with key K
$[M]_K$	**Message** M **signed with key** K

Uses of cryptography – secrecy and integrity

Scenario 1: secret communication with a shared secret key

Alice wishes to send some information secretly to Bob. Alice and Bob share a secret key K_{AB}



Problem 1: how can *Alice* send a shared key K_{AB} to *Bob* securely?

Problem 2: How does *Bob* know that any $\{M\}_{K_{AB}}$ is not a copy of an earlier encrypted message from *Alice* that was captured by *Mallory* and replayed later?

١٩

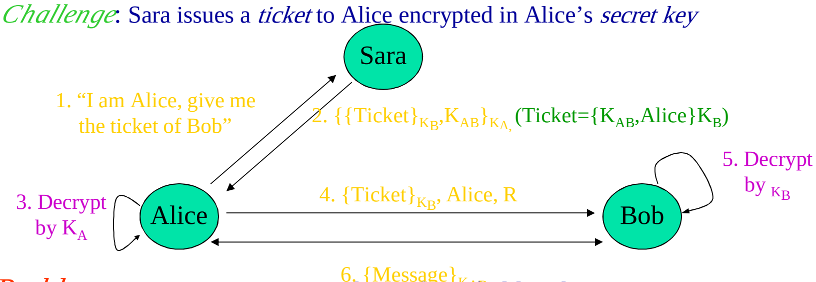
Uses of cryptography – authentication (1)

Scenario 2: Authenticated communication with a server

Alice wishes to access files held by *Bob*. *Sara* is an authentication server that is securely managed, and it knows Alice's key K_A and Bob's key K_B

Ticket: an encrypted item issued by an authentication server, containing the *identity of the principal* to whom it is issued and a *shared key* that has been generated for the current communication session

Challenge: Sara issues a *ticket* to Alice encrypted in Alice's *secret key*



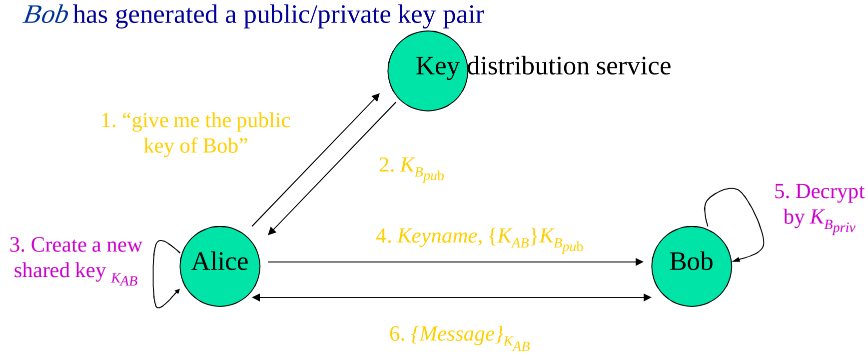
Problem: no protection against the replay of old authentication messages

٢٠

Uses of cryptography – authentication (2)

Scenario 3: Authenticated communication with public keys

Bob has generated a public/private key pair



Problem: *Mallory* may intercept *Alice*'s initial request to the key distribution service for *Bob*'s public-key certificate and send a response containing his own public key ٢١

Uses of cryptography – digital signatures

Digital signature: verify to a third party that a message is an unaltered copy of one produced by the signer

Digest: a fixed-length compressed message

Secure digest function: similar to checksum function, unlikely to produce a similar digest value for two different messages

Scenario 4: digital signatures with a secure digest function

Alice want to sign a document M that she is the originator of it

1. Alice computes *Digest(M)*
2. Alice make *M*, $\{Digest(M)\}K_{A_{priv}}$ available
3. Bob obtains the signed document, extracts *M* and computes *Digest(M)*
4. Bob decrypts $\{Digest(M)\}K_{A_{priv}}$ using Alice's public key $K_{A_{pub}}$ and compares the result with his calculated *Digest(M)*

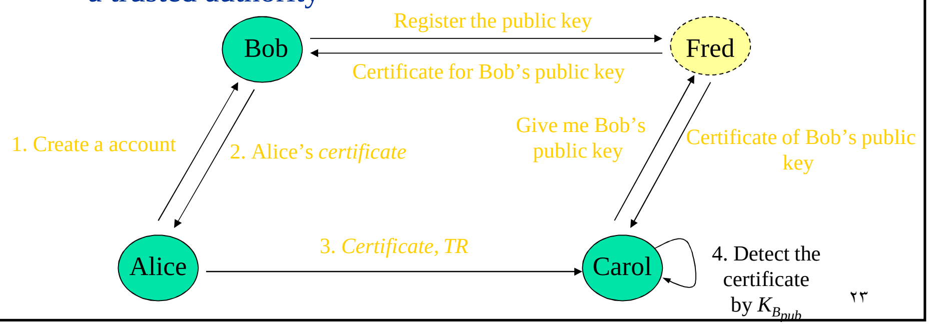
٢٢

Certificates

Digital certificate: a statement signed by a principal

Scenario 5: The use of certificates

Bob: a bank, *Alice*: a customer who has an account with Bob's bank, *Carol*: a vendor who accept Alice's transaction, *Fred*: a trusted authority



Alice's bank account certificate

1. *Certificate type*:	Account number
2. *Name*:	Alice
3. *Account*:	6262626
4. *Certifying authority*:	Bob's Bank
5. *Signature*:	$\{Digest(field\ 2 + field\ 3)\}_{K_{Bpriv}}$

Public-key certificate for Bob's bank

1. *Certificate type*	Public key
2. *Name*:	Bob's Bank
3. *Public key*:	K_{Bpub}
4. *Certifying authority*:	Fred – The Bankers Federation
5. *Signature*:	$\{Digest(field\ 2 + field\ 3)\}_{K_{Fpriv}}$

٢٥

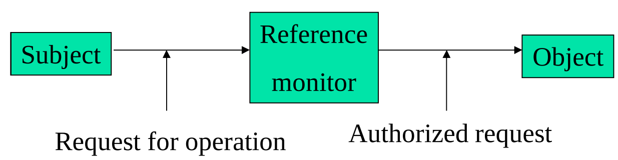
Certificates *… continued*

- To make certificates useful
  - A standard format and representation
  - Agreement on the manner of certificates chain
  - a trusted authority
- Time failure
  - include an expire data

٢٦

Access control

- A request can be carried out only if the client has sufficient access rights for the invocation.
- Formally, verifying access rights is referred to as access control, where authorization is about granting access right.
- General model of controlling access to objects



٢٧

Access control...continued

- Access control matrix, each subject is represented by a row in this matrix, each object is represented by a column.
- M[s,o] lists precisely which operations subject s can request to be carried out on object o.
- Each object maintains a list of the access rights of subjects that want to access the object. It is the column-wise of M, called Access Control List
- Another approach is to distribute matrix row-wise by giving each subject a list of capabilities.
- Protection domain: an execution environment shared by a collection of processes, contains a set of <object, access rights>

٢٨

Chapter 11: Security

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

٢٩

Cryptographic Algorithms

Message M, key K, published encryption functions E, D

- **Symmetric (secret key)**

  $E(K, M) = \{M\}_K$ $D(K, E(K, M)) = M$

  Same key for E and D

  M must be hard (infeasible) to compute if **K** is not known.

  Usual form of attack is brute-force: try all possible key values for a known pair M, $\{M\}_K$. Resisted by making **K** sufficiently large ~ 128 bits

- **Asymmetric (public key)**

  Separate encryption and decryption keys: K_e, K_d

  $D(K_d. E(K_e, M)) = M$

  depends on the use of a *trap-door function* to make the keys. E has high computational cost. Very large keys > 512 bits

- **Hybrid protocols - used in SSL (Transport layer security TLS)**

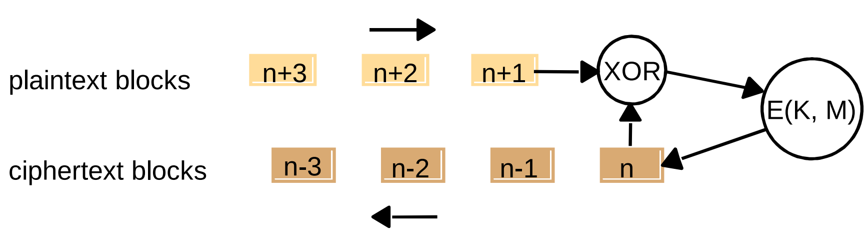
  Uses asymmetric crypto to transmit the symmetric key that is then used to encrypt a session.

٣٠ *

Different Ciphers

- **Block ciphers**
  - Fixed size blocks of data, e.g. 64 bits is popular
    - Recognize repeated patterns, short of integrity guarantee
  - Cipher block chaining (CBC)
    - Each plaintext block is combined with the preceding ciphertext block using the exclusive-or operation before it is encrypted
    - restricted to reliable connection

Cipher block chaining

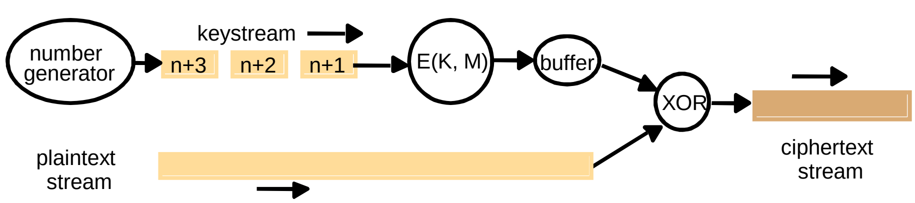


Different Ciphers

- **Stream ciphers**
  - Convert plaintext to ciphertext one bit at a time
  - Keystream
    - an arbitrary-length sequence of bits, Encrypt the keystream, XOR the keystream with the data stream
    - Keystream is secure, so is the data stream
  - Keystream generator
    - E.g. a random number generator which is agreed between sender and receiver

٣٣

Stream Cipher



٣٤

Design of cryptographic algorithms

- **Based on** *Information Theory*
- *Confusion*
  - Combine each block of plaintext with the key
  - Non-destructive operations, e.g. XOR, circular shifting
  - Obscure the relationship between M and $\{M\}_K$
- *Diffusion*
  - Dissipate the regular patterns
  - transpose portions of each plaintext block

٣٥

Symmetric encryption algorithms

These are all programs that perform confusion and diffusion operations on blocks of binary data

TEA: a simple but effective algorithm developed at Cambridge U (1994) for teaching and explanation. *128-bit key, 700 kbytes/sec*

DES: The US Data Encryption Standard (1977). No longer strong in its original form. *56-bit key, 350 kbytes/sec.*

Triple-DES: applies DES three times with two different keys. *112-bit key, 120 Kbytes/sec*

IDEA: International Data Encryption Algorithm (1990). Resembles TEA. *128-bit key, 700 kbytes/sec*

AES: A proposed US Advanced Encryption Standard (1997). *128/256-bit key.*

There are many other effective algorithms. See Schneier [1996].

The above speeds are for a Pentium II processor at 330 MHZ.

٣٦ *

Secret-key (symmetric) algorithms

- TEA (Tiny Encryption Algorithm) [CU1994]
  - Cipher block: 64 bits, 2 integer
  - Encryption key: 128 bits, 4 integer
    - Against brute-force attack
  - Confusion: XOR (^) and shift (<<, >>)
  - Diffusion: shift and swap
  - *delta*: obscure the key
  - Two very minor weaknesses [Wheeler and Needham 1997]
  - Application Example

TEA encryption function

```
void encrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];                          1
    unsigned long delta = 0x9e3779b9, sum = 0; int n;                2
    for (n= 0; n < 32; n++) {                                        3
        sum += delta;                                                4
        y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);        5
        z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);        6
    }
    text[0] = y;  text[1] = z;                                       7
}
```

TEA decryption function

```
void decrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = delta << 5;  int n;
    for (n= 0; n < 32; n++) {
        z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
        y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        sum -= delta;
    }
    text[0] = y; text[1] = z;
}
```

٣٩

Secret-key (symmetric) algorithms…*continued*

- **Data Encryption Standard DES [IBM1977]**
  - Adopted as a US national standard
  - 64-bit cipher block, 56-bit key
  - Be cracked in a widely publicized brute-force attack
  - Triple-DES: $E_{3DES}(K_1,K_2,M)=E_{DES}(K_1,D_{DES}(K_2,E_{DES}(K_1,M)))$
- **The International Data Encryption Algorithm IDEA [1990]**
  - Successor to DES, 128-bit key, 3 times faster than DES
- **Advanced Encryption standard Algorithm AES [NIST1999]**
  - 128-bit, 192-bit, 256-bit key
  - Like to be the most widely used symmetric encryption algorithms

٤٠

Public-key (asymmetric) algorithms

- $D(K_d, E(K_e,M)) = M$
  - K_e is public, K_d is secret
- The Rivest, Shamir, and Adelman RSA
  - based on the use of two very large prime numbers
  - $K_e = <e, N>, K_d = <d, N>$
  - factorization of N is so time consuming
  - In application, N’s length should be at least 768 bits

٤١

Hybrid cryptographic protocols

- Public-key cryptography
  - Pros: no need for a secure key-distribution mechanism
  - Cons: high processing cost
- Secret-key cryptography
  - Pros: effective
  - Cons: need for a secure key-distribution mechanism
- Hybrid encryption scheme
  - Secret key distribution: public-key cryptography
  - Data transmission: secret-key cryptography

٤٢

Chapter 11: Security

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

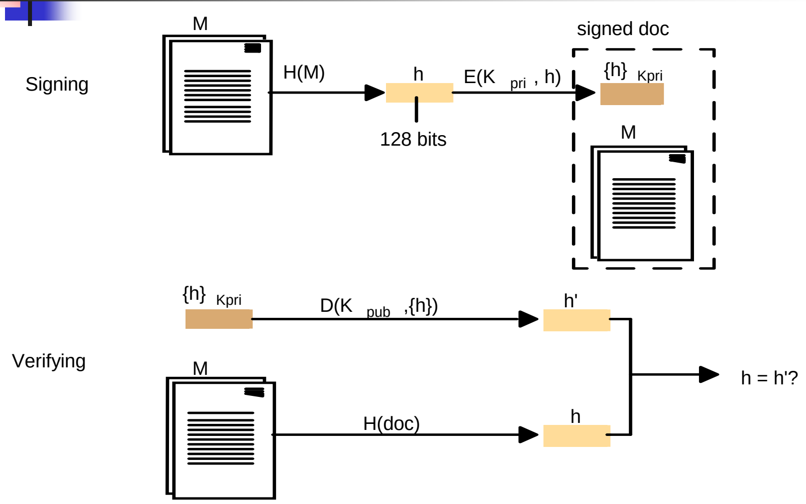
Handwritten Signature and Digital Signature

- **Handwritten signatures**
  - Authentic: no alteration
  - Unforgeable: can not be copied and placed other doc.
  - Non-repudiable (signer can not deny that the document was signed by them)
- **Digital signatures**
  - Bind a unique and secret attribute of the signer to the document
  - Digital signing
    - Example of a singed doc: $M,A,[H(M)]_{K_A}$, A: signer ID, K_A: signer's key
  - Digest functions
    - secure hash functions: ensure H(M)<>H(M`)

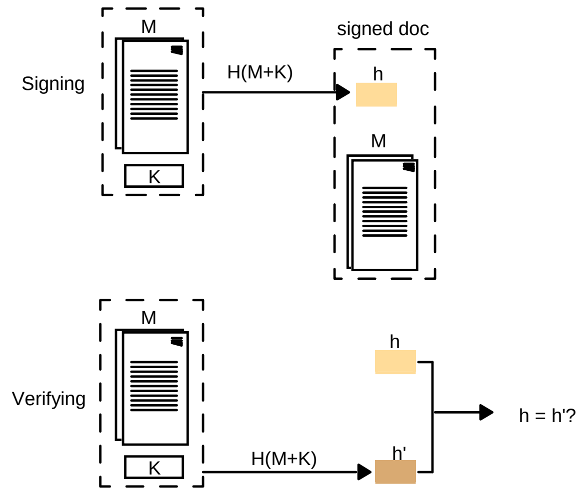
Digital signature Keys

- Digital signatures with public keys
  - convenient solution in most situations
- Digital signatures with secret keys
  - Problems caused by secret key digital signature
    - Secure secret key distribution mechanism
    - Verifier could forge signers signature
  - MAC (*message authentication code*)
    - Sender sends receiver a shared key via secure channel
    - No encryption, 3-10 times faster than symmetric encryption
    - Suffer problems also

Digital signatures with public keys



Low-cost signatures with a shared secret key



٤٧

Secure digest functions

- **Requirements on secure digest function *h* = *H(M)***
  - Given *M*, it is easy to compute *h*
  - Given *h*, it is hard to compute *M*
  - Given *M*, it is hard to find another message *M`*, such that *H(M) = H(M`)*
- **Vulnerable to Birthday attack**
- **Two widely used digest function :**
  - **MD5** [Rivest 1992], 128-bit digest
  - **Secure Hash Algorithm SHA** [NIST 1995], 160-bit digest

٤٨

Certificate standards and certificate authorities

- X.509
  - The most widely used standard format for certificates[CCITT 1988b]
  - Based on the global uniqueness of distinguished names
  - E.g. www.verisign.com , www.cern.net issue public key certificates.
- SPKI (Simple Public-key Infrastructure)
  - Creation and management of sets of public certificates
  - Chains of certificates

٤٩

Chapter 11: Security

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

٥٠

Cryptography pragmatics

- **Performance of cryptographic algorithms**
- **Applications of cryptography and political obstacles**
  - NSA (National Security Agency): restrict the strength of cryptography
  - FBI: privileged access to all cryptographic keys
  - PGP (Pretty Good Privacy)
    - A example of cryptographic method which is not controlled by US government
    - generate and manage public and secret keys
    - RSA for authentication and secret key transmission
    - IDEA or 3DES for data transmission

Cryptography pragmatics

	Key size/hash size (bits)	*Extrapolated 2.1 GHz (kbytes/sec.)*	*PRB optimized (kbytes/s) 90 Mhz*
TEA	128	700	-
DES	56	350	7746
Triple-DES	112	120	2842
IDEA	128	700	4469
RSA	512	7	-
RSA	2048	1	-
MD5	128	1740	62425
SHA	160	750	25162

Chapter 11: Security

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

Needham and Schroeder with secret keys

The K_{AB} is supplied to A in two forms, one that A can use to encrypt the messages that it sends to B and one that it can transmit securely to B.

The authentication server S maintains a table containing a name and a secret key for each principal known to the system.

A ticket is an encrypted message containing a secret key for use in communication between A and B

A nonce is an integer value that is added to a message to demonstrate its freshness.

Needham and Schroeder authentication protocol

Header	*Message*	*Notes*
1. A->S:	A, B, N_A	A requests S to supply a key for communication with B.
2. S->A:	$\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_B}\}_{K_A}$	S returns a message encrypted in A's secret key, containing a newly generated key K_{AB} and a 'ticket' encrypted in B's secret key. The nonce N_A demonstrates that the message was sent in response to the preceding one. A believes that S sent the message because only S knows A's secret key.
3. A->B:	$\{K_{AB}, A\}_{K_B}$	A sends the 'ticket' to B.
4. B->A:	$\{N_B\}_{K_{AB}}$	B decrypts the ticket and uses the new key K_{AB} to encrypt another nonce N_B.
5. A->B:	$\{N_B - 1\}_{K_{AB}}$	A demonstrates to B that it was the sender of the previous message by returning an agreed transformation of N_B.

- Remedy for stale 3: $\{K_{AB}, A, t\}_{K_B}$

٥٥

Kerberos

- **Three kinds of security objects**
  - *ticket*: a token that verifies the sender has recently been authenticated
  - *authentication*: a token that proves user's identity and currency of communication with a server
  - *session key*: encrypt communication
- **System architecture of Kerberos**
- **Kerberos protocol**

٥٦

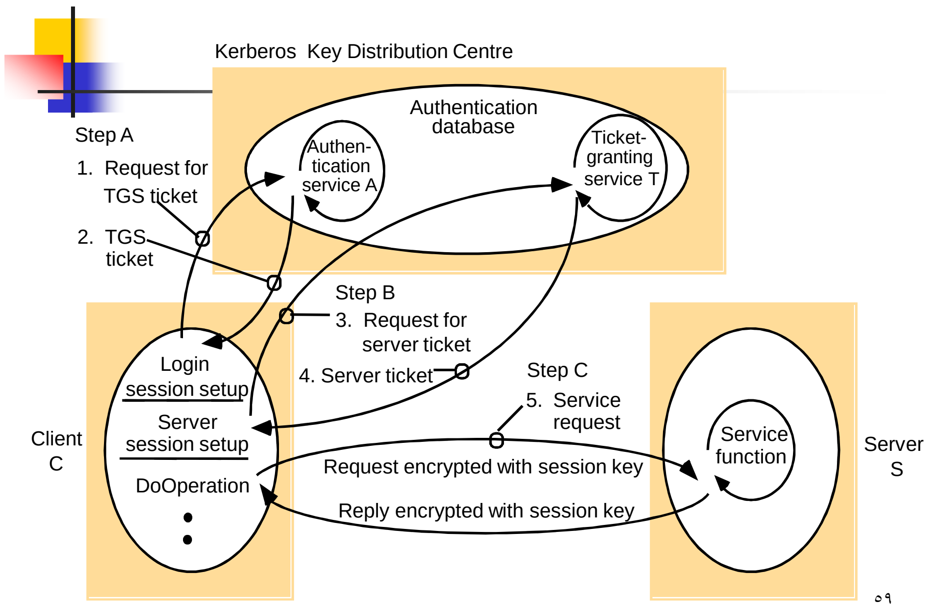
Application of Kerberos

- Campus network[MIT 1990]
- Users’ passwords and services secrets
  - Be known by owner and authentication server
- Login with Kerberos
  - password is prevented from eavesdropping
- Access servers with kerberos
  - ticket containing expire time

Secure Socket Layer (SSL)

- **SSL protocol stack**
  - hybrid scheme: public-key cryptography for authentication, secret-key cryptography for data communication
- **Negotiable encryption and authentication algorithms**
  - requirement of an open network environment
  - handshake protocol
- **Application**
  - [netscape 1996], *de facto*, *https*, integrated in web browsers and web servers
  - ticket: verify the sender has recently been authenticated

System architecture of Kerberos



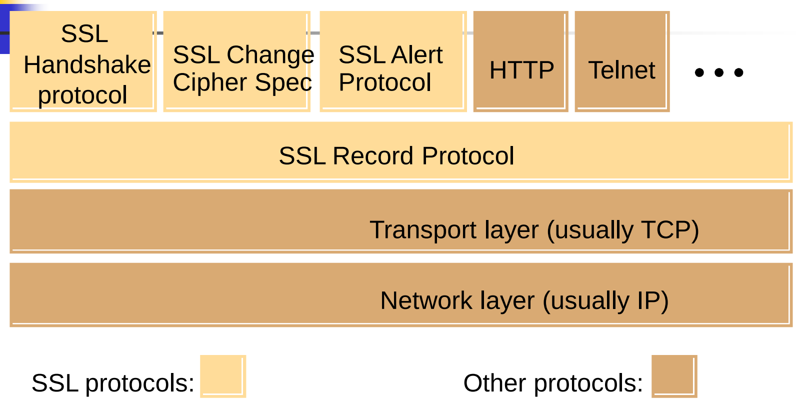
٥٩

Kerberos protocol

Header	Message
1. C- A: Request for TGS ticket	*C, T, n*
2. A- C: TGS session key and ticket	$\{K_{CT}, n\}K_C$, $\{ticket(C,T)\}K_T$ containing $C,T,t1,t2,K_{CT}$
3. C-T: Request ticket for service S	$\{auth(C)\}K_{CT}$, $\{ticket(C,T)\}K_T$, S, n containing $\{C,t\}K_{CT}$
4. T-C: Service ticket	$\{K_{CS}, n\}K_{CT}$, $\{ticket(C,S)\}K_S$
5. C-S: Service request	$\{auth(C)\}K_{CS}$, $\{ticket(C,S)\}K_S$, *request,n*
6. S-C: Server authentication	$\{n\}K_{CS}$

٦٠

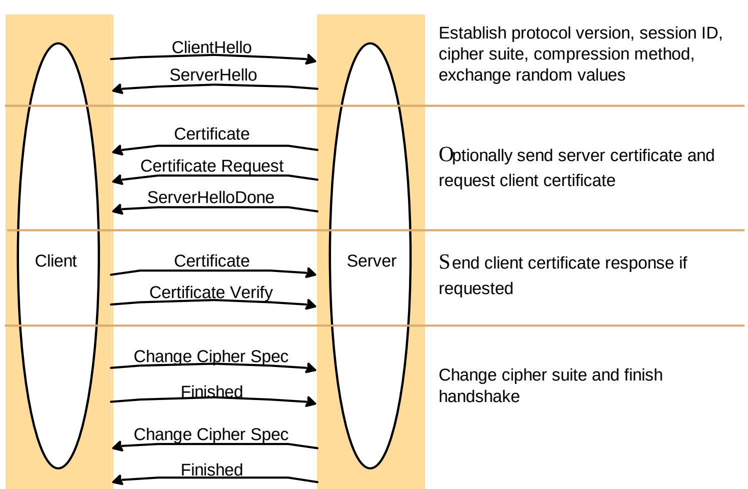
SSL protocol stack



SSL Record Protocol: implements a secure channel, encrypting and authenticating messages

SSL Handshake protocol and the other related protocols establish and maintain an SSL session

SSL handshake protocol



SSL handshake configuration options

Component	*Description*	*Example*
Key exchange method	the method to be used for exchange of a session key	RSA with public-key certificates
Cipher for data transfer	the block or stream cipher to be used for data	IDEA
Message digest function	for creating message authentication codes (MACs)	SHA

Chapter 11: Security

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptography pragmatics
- Case studies: Needham-Schroeder, Kerberos, SSL&Millicent
- Summary

Summary

- **Guide for designing a secure system**
  - worst case assumptions
- **Public-key and secret-key cryptography**
  - TEA
  - RSA
- **Access control mechanisms**
  - capability and ACL
- **Needham-Schroeder authentication protocol**
  - Challenge, Ticket
- **Kerberos**
  - Ticket, Authenticator, Session key

٦٥