Slides for Chapter 11: Security

From **Coulouris, Dollimore, Kindberg and Blair**

Distributed Systems: Concepts and Design

Edition 5, © Addison-Wesley 2012

Overview of Chapter

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptographic pragmatics
- Case studies: Needham-Schroeder,Kerberos, TLS, 802.11 WiFi

Introduction

Resources:

- Processes encapsulate resources (objects and system resources)
- Some clients are authorized to access these resources through interfaces
- Some principals (users or other processes) are authorized to operate on resources
- Resources must be protected against unauthorized access/operations

Network:

- Processes interact through a shared network
- Enemies (attackers) can also access the network
- Attackers can copy/read messages being transmitted
- Attackers can inject arbitrary messages purporting to come from any source addressed to any destination

Introduction (cont.)

Policies vs. mechanisms:

- Security *policies* provide the rules for accessing/operating on resources and for protecting messages
- Security *mechanisms* provide the means and technology for enforcing the security policies
- Focus is on security mechanisms

Cryptography:

- Distinct subject from computer security
- Basis for most security mechanisms
- Encodes information so that only intended recipients can decode
- Can also be used to authenticate document senders

Figure 11.1
Familiar names for the protagonists in security protocols

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

Introduction (cont.)

Types of security threats:

- *Leakage*: acquiring information by unauthorized recipient
- *Tampering*: unauthorized alteration of information
- *Vandalism*: interference with system operation (without gain)

Types of attacks on communication channels:

- *Eavesdropping*: obtaining copies of messages
- *Masquerading*: sending/receiving messages using identity of another
- *Message tampering*: intercepting and altering message contents
- *Replaying*: storing intercepted messages and sending them later
- *Denial of service*: flooding a channel or resource with messages so others cannot access it

Introduction (cont.)

Threats from mobile code:

- Some programming languages allow programs from a remote server to be loaded into a local process and executed locally
- *Example:* Java applets
- *Threat*: internal interfaces and objects within local executing process may be attacked by mobile code

Java Virtual Machine (JVM) tries to protect against malicious mobile code:

- *Security manager*: determines which resources are available to an application, cannot be replaced
- *Browsers*: specify that applets cannot access local files, network sockets
- Downloaded classes stored separately from local classes
- Newer versions of Java close loopholes that come up

Introduction (cont.)

Electronic transactions that depend on security:

- *Email*: confidential emails or emails that include protected info
- *Internet purchases*: require secure credit card/bank info transmission
- *Internet banking*: requires secure access/update/transmission

Requirements for secure web purchasing and banking:

- Authenticate vendor to buyer
- Keep buyers payment details secure
- Downloaded goods (e.g. music) delivered without alteration and without interception by third parties
- For banking transactions, authenticate identity of customer by the banking server (non-repudiation of identity)

Introduction (cont.)

Designing secure systems:

- Construct list of threats, show that security mechanisms can prevent these threats
- Use formal verification methods and exhaustive testing – may not be possible in complex systems
- Use auditing to detect unforeseen violations

Security log:

- Includes info on each transaction: principal id, resource accessed, operation, timestamp

Tradeoffs:

- Balance the cost overhead versus the threat level
- Inappropriate security mechanisms may exclude legitimate users

Overview of Chapter

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptographic pragmatics
- Case studies: Needham-Schroeder,Kerberos, TLS, 802.11 WiFi

Overview of Security Techniques

- Worst-case assumptions
- Cryptography
- Digital signatures
- Certificates
- Access control
- Credentials
- Firewalls

Worst-case assumptions and design guidelines

- Process interfaces can be *exposed to attackers*
- Networks are *insecure* – message sources can be *falsified*, host addresses can be *spoofed*
- Secrets (passwords, encryption keys) lifetimes should be time-limited
- Make encryption/authentication algorithms public to challenge scrutiny by third parties (only keys kept secret)
- Attackers may have large resources
- Minimize trusted computing base (hardware nodes, software components) that is responsible for security implementation

Cryptography

- Process of encoding a message to hide its content
- Many algorithms based on concept of secret and public *keys* used in encryption/decryption

Classes of cryptographic algorithms:

- *Shared secret keys* between sender/receiver – same key used to encrypt and decrypt
- *Public/private key pairs* – sender uses public key published by receiver to encrypt message, receiver uses private key to decrypt (require 100 to 1000 times processing power as shared secret key algorithms but more secure in some cases)

Figure 11.2
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

Three major roles:

- Secrecy and integrity
- Authentication
- Digital signatures

Assumption:

- The participants (Bob, Alice, etc.) have agreed on the encryption algorithms to use and have copies of these algorithms

Secrecy and integrity

- Secrecy of encrypted message is maintained as long as decryption key is not compromised
- Integrity of the encrypted information by including checksum in encrypted message

Scenario 1:

- Alice sends encrypted messages $\{M_i\}$ to Bob using encryption algorithm $E(K_{AB}, M)$ with shared secret key – Bob decrypts each received message using same key $D(K_{AB}, M)$
- How does Alice send K_{AB} to Bob securely?
- How does Bob know that a message was to “captured” by Mallory and replayed later? (e.g. send payment message multiple times)

Authentication within an organization

Scenario 2 (Authentication protocol using trusted local server):

- Alice wants to access files held by Bob on company server
- Sara is the *secure authentication server* – knows Alice's encryption key K_A and Bob's encryption key K_B
- Encrypted *Ticket* is issued by Sara containing identity of requester (Alice) plus a shared key K_{AB} to be used in one communication session
- Sara sends to Alice $\{\{\text{Ticket}\}_{KB}, K_{AB}\}_{KA}$
- Alice sends Ticket to Bob with request: Ticket_{KB}, Alice, R
- Bob decrypts ticket to get $\{\text{Alice}, K_{AB}\}$ – *session key* K_{AB} can be used to encrypt/decrypt messages between Alice and Bob
- Works in a single organization with trusted authentication server Sara – not for general ecommerce

Authentication cryptographic challenge

- Challenge step is message send by Sara to Alice encrypted using K_A

- Sara sends to Alice $\{\{Ticket\}_{KB}, K_{AB}\}_{KA}$
- Only Alice can decrypt it using K_A and send Ticket to Bob

Authentication using public keys

Scenario 3 (Authentication protocol using trusted key distribution server):

- Alice requests *public-key certificate* from trusted key distribution server to get Bob's public key K_{Bpub}
- Alice creates session key K_{AB} and sends it to Bob encrypted using K_{Bpub}
- Only Bob can decrypt the message using his private decryption key
- Bob and Alice can set up encrypted communication session using K_{AB}

- This scheme is vulnerable to man-in-the-middle attack – Mallory can intercept initial message and return his own public key
- Bob's certificate is signed by well-known authority to prevent this attack

Digital signatures

- An irreversible binding to message or document of a secret known only to signer
- Example: Encrypt message *digest* (created from full message using a *secure digest function*) using signer secret key
- Typically use signer's private key to encrypt message – receive can decrypt message using receiver's public key

Scenario 4:

- Alice computes fixed-length digest *Digest(M)* of the message M
- Alice encrypts digest using her private key, and appends it to M
- Bob receives signed document, extracts M, computes Digest(M)
- Bob decrypts appended encrypted signature using Alice's public key, compares the decrypted and computed digest to see if they match

Certificates

Scenario 5:

- Bob is a bank
- Upon Alice's request, Bob provides a signed certificate with Alice's bank account number using Bob's private key
- Certificate can be authenticated using Bob's public key – obtained from a trusted authority

- Certificates are used extensively in ecommerce
- Banking authority can issue certificates to validate banks

Figure 11.3
Alice's bank account certificate

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

Figure 11.4
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:*	$\{Digest(field\ 2 + field\ 3)\}_{K_{Fpriv}}$

Access control

- Protects access to resources by processes
- Access matrix: processes x resources
- Capability: A binary value that acts as an access key to invoke certain operations on a resource – each process has a set of capabilities to the resources it is allowed to access

- Access control lists: each resource has a list of processes that are allowed to access it

Credentials

- A set of evidence provided by a principal when accessing a resource
- Example: A certificate from a trusted authority stating the principal’s identity
- Speaks for: credential can *speak for* a principal – e.g. private key speaks for a user

- Delegation: allows a principal to use credentials delegated from the authority of another principal

Firewalls

- Firewall: a setup to protect an organization’s intranets by performing filtering actions on incoming and outgoing communications
- All external communication is intercepted – if approved, they communication is forwarded
- Protects against external attacks

- Delegation: allows a principal to use credentials delegated from the authority of another principal

Overview of Chapter

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptographic pragmatics
- Case studies: Needham-Schroeder,Kerberos, TLS, 802.11 WiFi

Cryptographic algorithms

- Encryption: Transforms *plaintext* message to *ciphertext*
- Decryption: Transforms *ciphertext* message back to *plaintext*
- Encryption algorithm E and an encryption key K are used to encrypt – should be easy (efficient)
- Decryption algorithm D and an decryption key K' are used to decrypt – should be difficult
- Symmetric algorithms: use same key K to encrypt and decrypt
- Asymmetric algorithms: use public/private key pair
- Block ciphers: fixed-length blocks of data bits operated on by E, D
- Cipher block chaining: apply XOR with preceding block when encrypted – apply XOR after decrypting (XOR is its own inverse)
- Stream cipher: incremental algorithms

Figure 11.5
Cipher block chaining

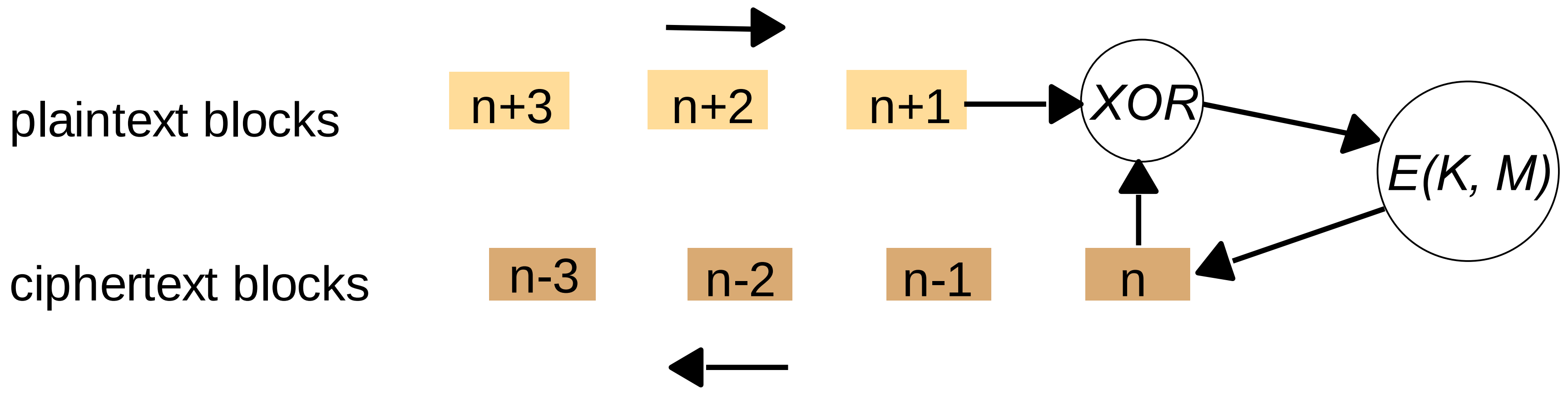
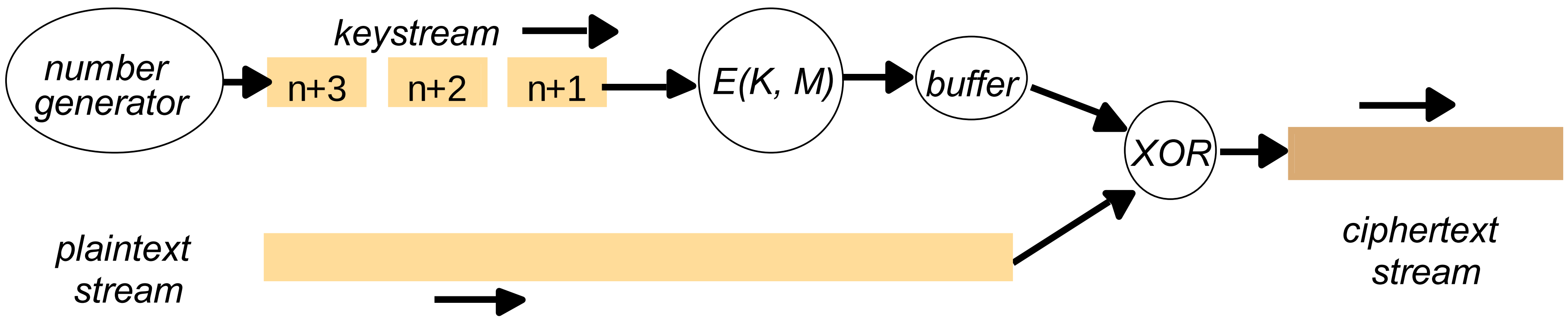


Figure 11.6
Stream cipher



Cryptographic algorithms

- TEA: Tiny Encryption Algorithm
- Symmetric algorithm using single secret ket to encrypt/decrypt

Figure 11.7
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
}
```

Figure 11.8
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;
}
```

Figure 11.9
TEA in use

```
void tea(char mode, FILE *infile, FILE *outfile, unsigned long k[]) {
/* mode is ’e’ for encrypt, ’d’ for decrypt, k[] is the key.*/
    char ch, Text[8]; int i;
    while(!feof(infile)) {
        i = fread(Text, 1, 8, infile);          /* read 8 bytes from infile into Text */
        if (i <= 0) break;
        while (i < 8) { Text[i++] = ' ';}       /* pad last block with spaces */
        switch (mode) {
        case 'e':
            encrypt(k, (unsigned long*) Text); break;
        case 'd':
            decrypt(k, (unsigned long*) Text); break;
        }
        fwrite(Text, 1, 8, outfile);            /* write 8 bytes from Text to outfile */
    }
}
```

Cryptographic algorithms

Some well-known symmetric algorithms:

- IDEA: International Data Encryption Algorithm
- RC4: stream cipher
- AES: Advanced Encryption Standard

Cryptographic algorithms

Asymmetric algorithms (public-private key pairs):

- RSA
- Elliptic curve algorithms

RSA Encryption - 1

To find a key pair *e*, *d*:

1. Choose two large prime numbers, *P* and *Q* (each greater than 10100), and form:
   $N = P \times Q$
   $Z = (P–1) \times (Q–1)$
2. For *d* choose any number that is relatively prime with *Z* (that is, such that *d* has no common factors with *Z*).
   We illustrate the computations involved using small integer values for *P* and *Q*:
   $P = 13, Q = 17 \rightarrow N = 221, Z = 192$
   $d = 5$
3. To find *e* solve the equation:
   $e \times d = 1 \bmod Z$

That is, $e \times d$ is the smallest element divisible by *d* in the series $Z+1, 2Z+1, 3Z+1, \ldots$.

$e \times d = 1 \bmod 192 = 1, 193, 385, \ldots$
385 is divisible by *d*
$e = 385/5 = 77$

RSA Encryption - 2

To encrypt text using the RSA method, the plaintext is divided into equal blocks of length k bits where $2^k < N$ (that is, such that the numerical value of a block is always less than N; in practical applications, k is usually in the range 512 to 1024).

$k = 7$, since $27 = 128$

The function for encrypting a single block of plaintext M is:

$E'(e,N,M) = M^e \bmod N$

for a message M, the ciphertext is $M^{77} \bmod 221$

The function for decrypting a block of encrypted text c to produce the original plaintext block is:

$D'(d,N,c) = c^d \bmod N$

Rivest, Shamir and Adelman proved that E' and D' are mutual inverses (that is, $E'(D'(x)) = D'(E'(x)) = x$) for all values of P in the range $0 \leq P \leq N$.

The two parameters e,N can be regarded as a key for the encryption function, and similarly d,N represent a key for the decryption function.

So we can write $K_e = <e,N>$ and $K_d = <d,N>$, and we get the encryption function:

$E(K_e, M) = \{M\}_K$ (the notation here indicating that the encrypted message can be decrypted only by the holder of the private key K_d) and $D(K_d, = \{M\}_K) = M$.

Overview of Chapter

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptographic pragmatics
- Case studies: Needham-Schroeder,Kerberos, TLS, 802.11 WiFi

Digital signatures

Traditional signatures used to verify that a document is:

- authentic
- unforgeable
- Non-repudiable

Digital signatures used to:

- Irrevocably bind a signer's identity to entire document
- Signed document consists of: M (electronic document), A (signer's identity), $[M]_K$ (encrypted copy of M with key K_A of A)
- Key can be secret shared key for symmetric encryption or it can be private key of A for asymmetric encryption
- Digest functions: secure hash function H(M)

Figure 11.10
Digital signatures with public keys

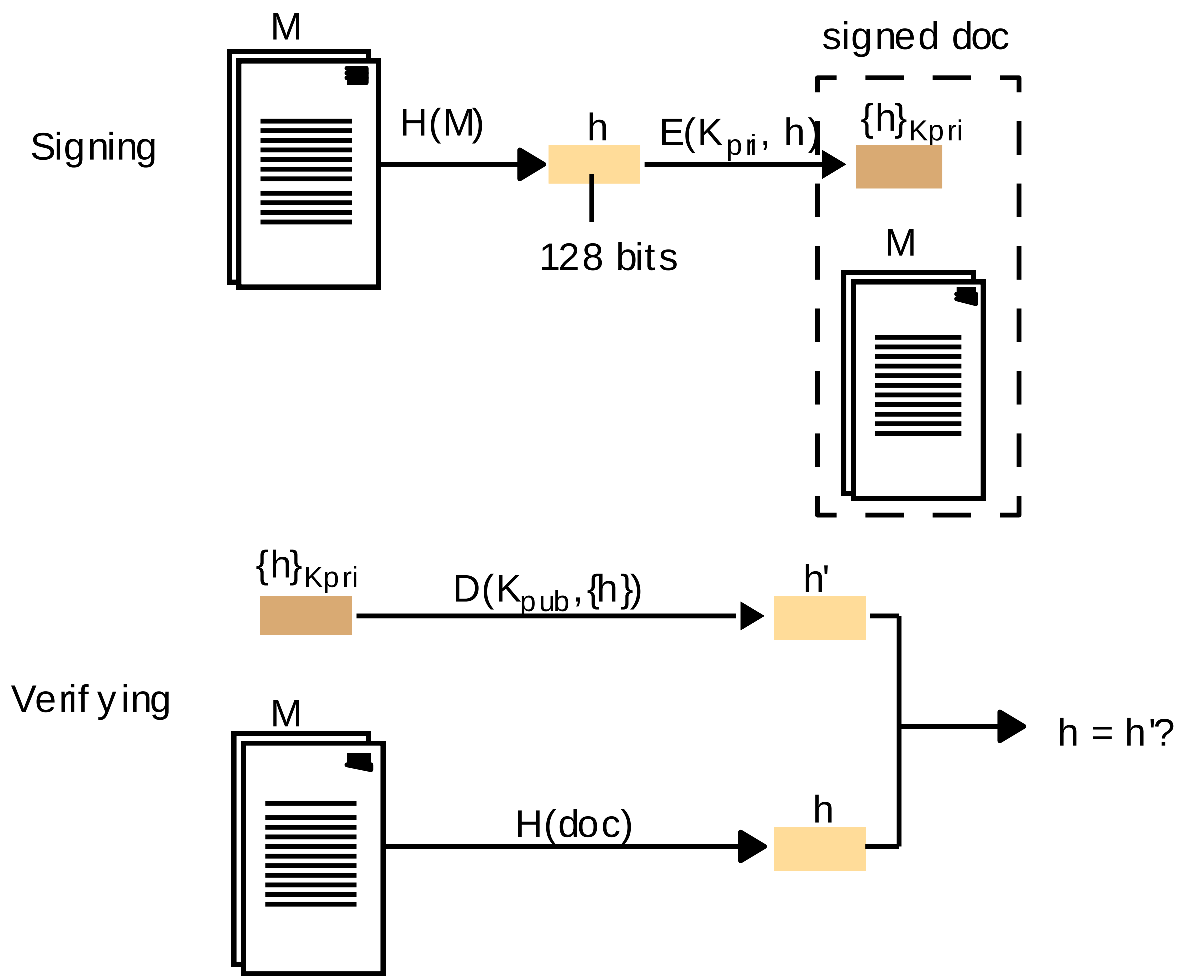
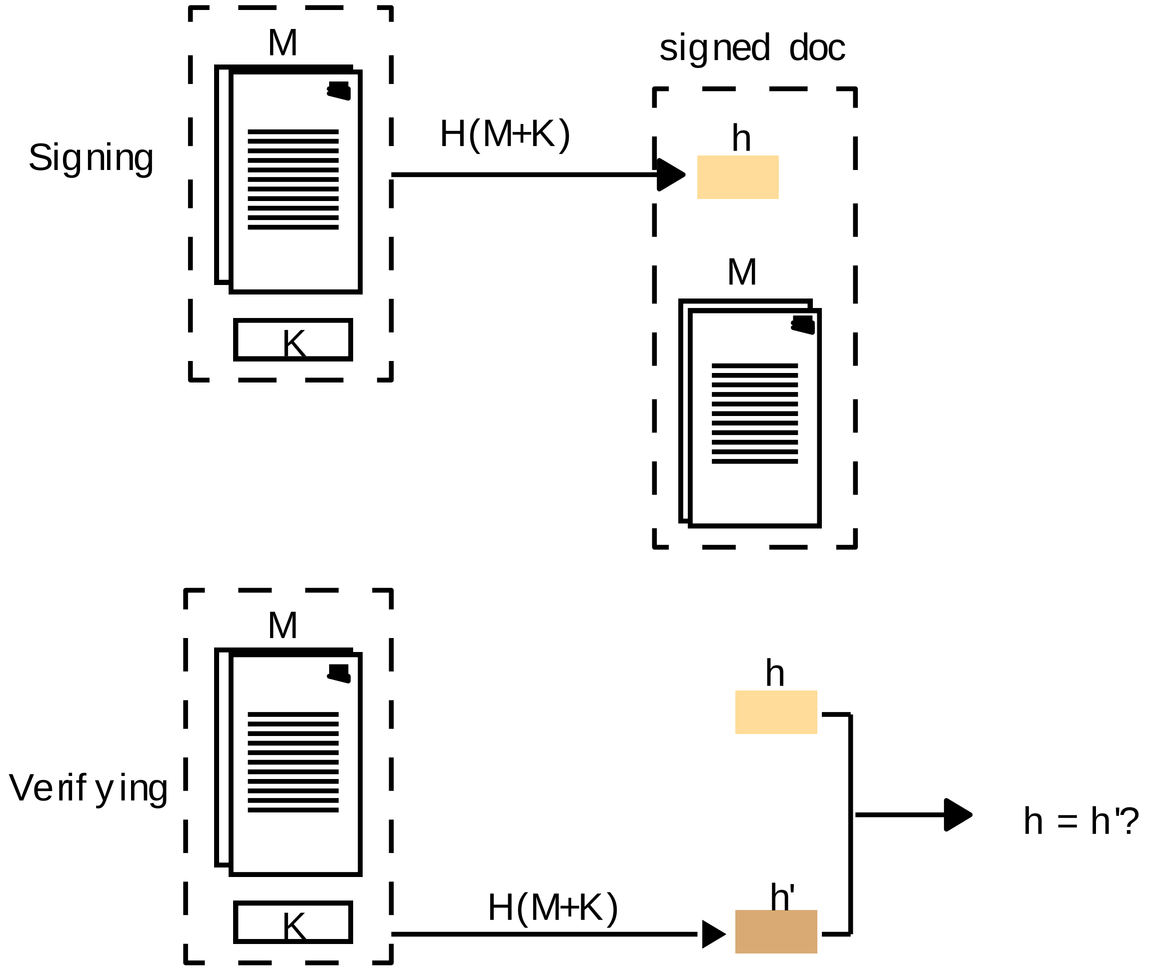


Figure 11.11
Low-cost signatures with a shared secret key



Certificate standards and authorities

X.509 standard is widely used:

- Subject
- Issuer
- Period of validity

- Used in ecommerce

Figure 11.12
X509 Certificate format

Subject	Distinguished Name, Public Key
Issuer	Distinguished Name, Signature
Period of validity	Not Before Date, Not After Date
Administrative information	Version, Serial Number
Extended Information	.

Overview of Chapter

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptographic pragmatics
- Case studies: Needham-Schroeder,Kerberos, TLS, 802.11 WiFi

Figure 11.13
Performance of symmetric encryption and secure digest algorithms

	Key size/hash size (bits)	*PRB optimized 90 MHz Pentium 1 (Mbytes/s)*	*Crypto++ 2.1 GHz Pentium 4 (Mbytes/s)*
TEA	128	–	23.801
DES	56	2.113	21.340
Triple-DES	112	0.775	9.848
IDEA	128	1.219	18.963
AES	128	–	61.010
AES	192	–	53.145
AES	256	–	48.229
MD5	128	17.025	216.674
SHA-1	160	–	67.977

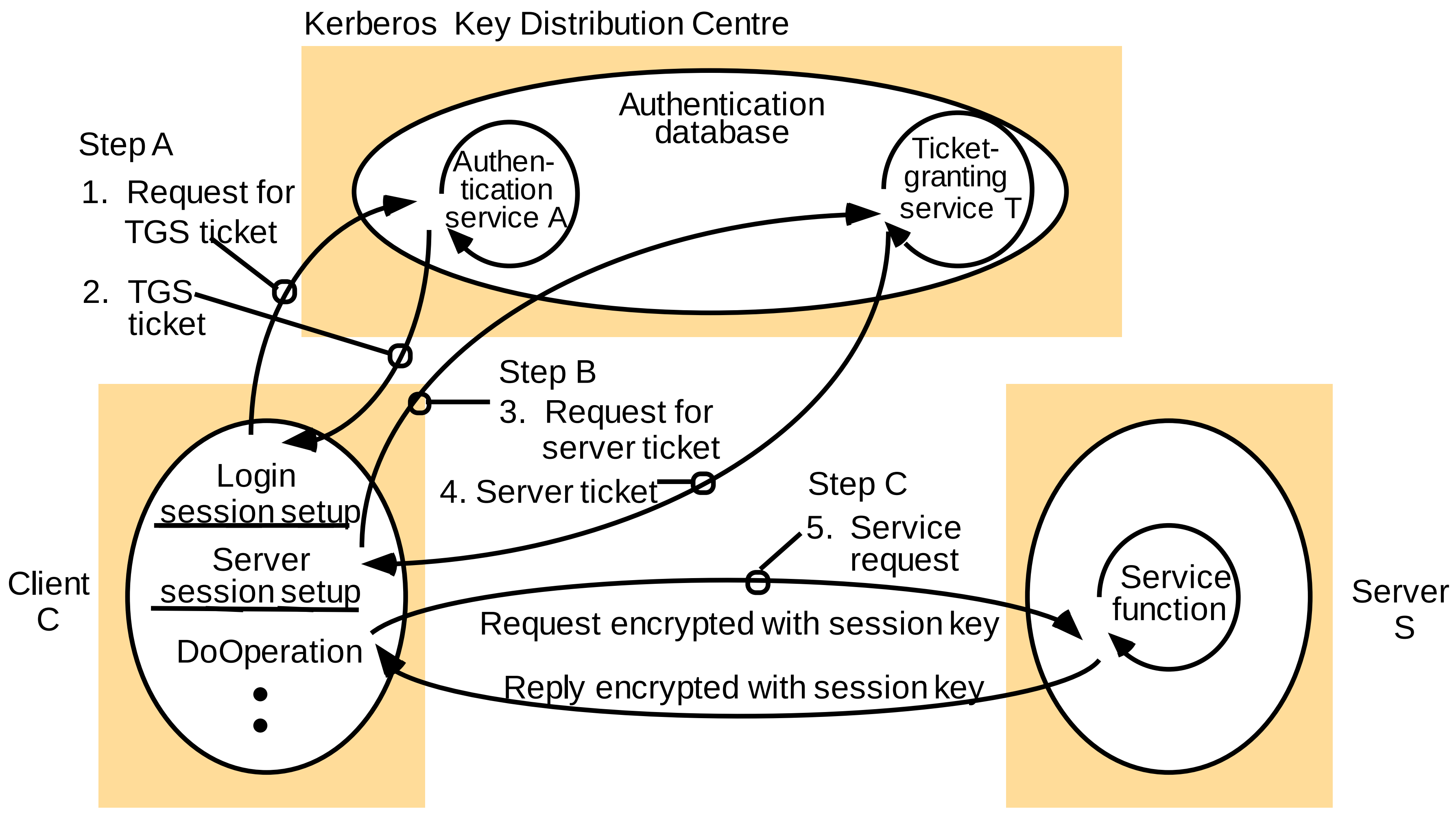
Overview of Chapter

- Introduction
- Overview of security techniques
- Cryptographic algorithms
- Digital signatures
- Cryptographic pragmatics
- Case studies: Needham-Schroeder,Kerberos, TLS, 802.11 WiFi

Figure 11.14
The Needham–Schroeder secret-key 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.

Figure 11.15
System architecture of Kerberos



Kerberos Key Distribution Centre
Authentication database
Authen-tication service A
Ticket-granting service T
Step A
1. Request for TGS ticket
2. TGS ticket
Step B
3. Request for server ticket
4. Server ticket
Step C
5. Service request
Client C
Login session setup
Server session setup
DoOperation
Request encrypted with session key
Reply encrypted with session key
Service function
Server S

Figure 11.16
SSL protocol stack

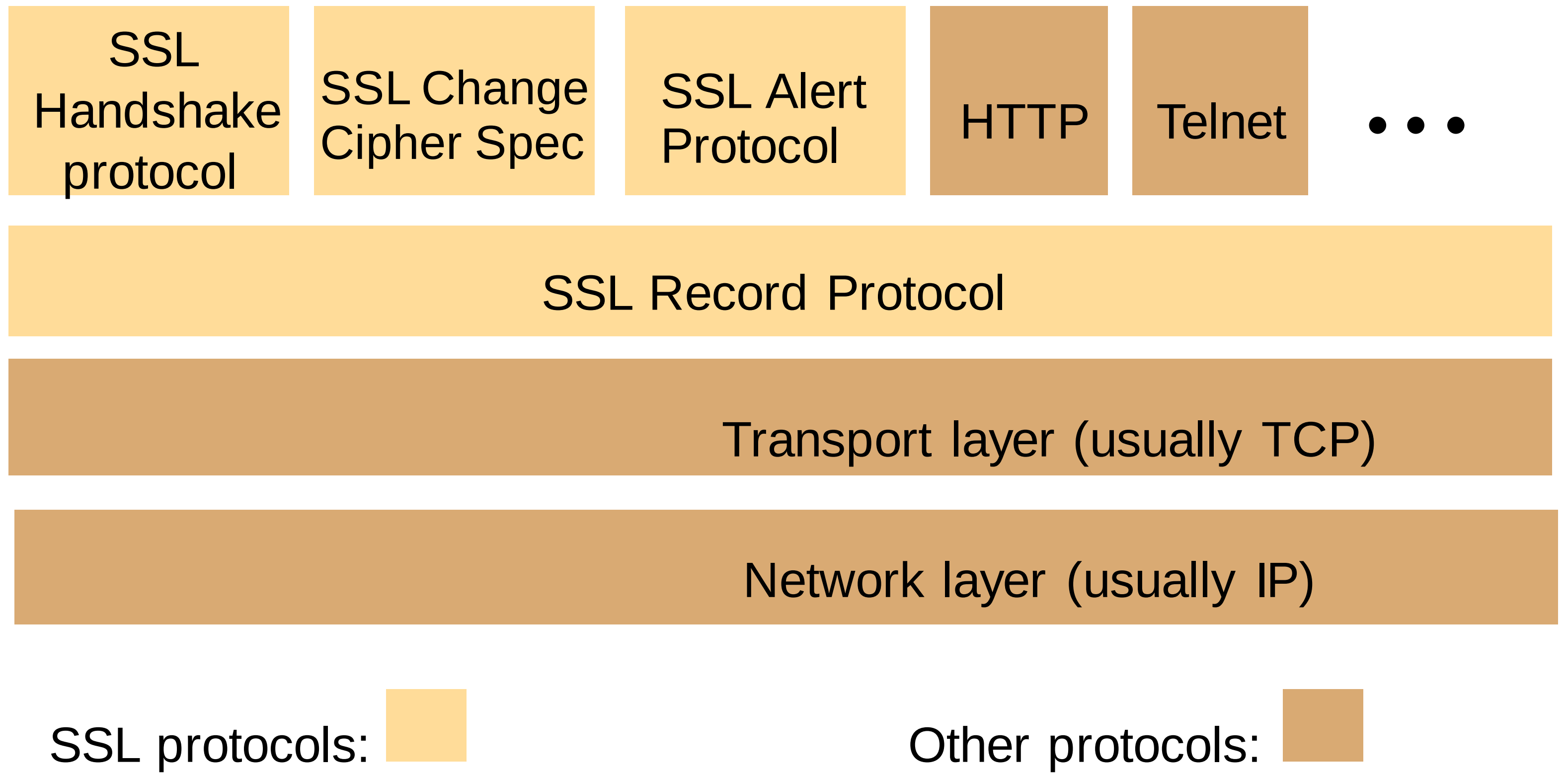
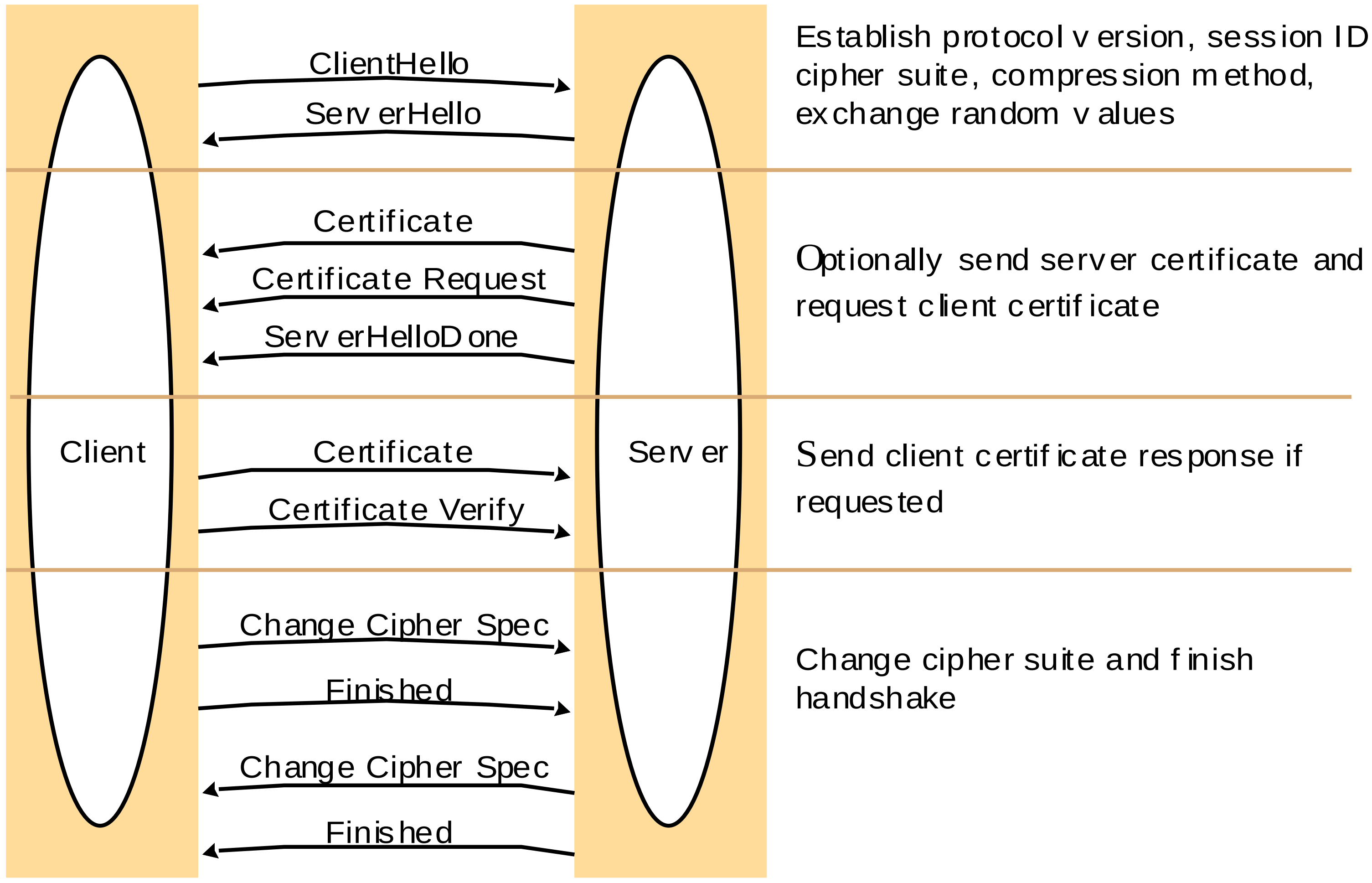


Figure 11.17
TLS handshake protocol



ClientHello
ServerHello
Certificate
Certificate Request
ServerHelloDone
Client
Certificate
Certificate Verify
Server
Change Cipher Spec
Finished
Change Cipher Spec
Finished
Establish protocol version, session ID cipher suite, compression method, exchange random values
Optionally send server certificate and request client certificate
Send client certificate response if requested
Change cipher suite and finish handshake

Figure 11.18
TLS 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-1

Figure 11.19
TLS record protocol

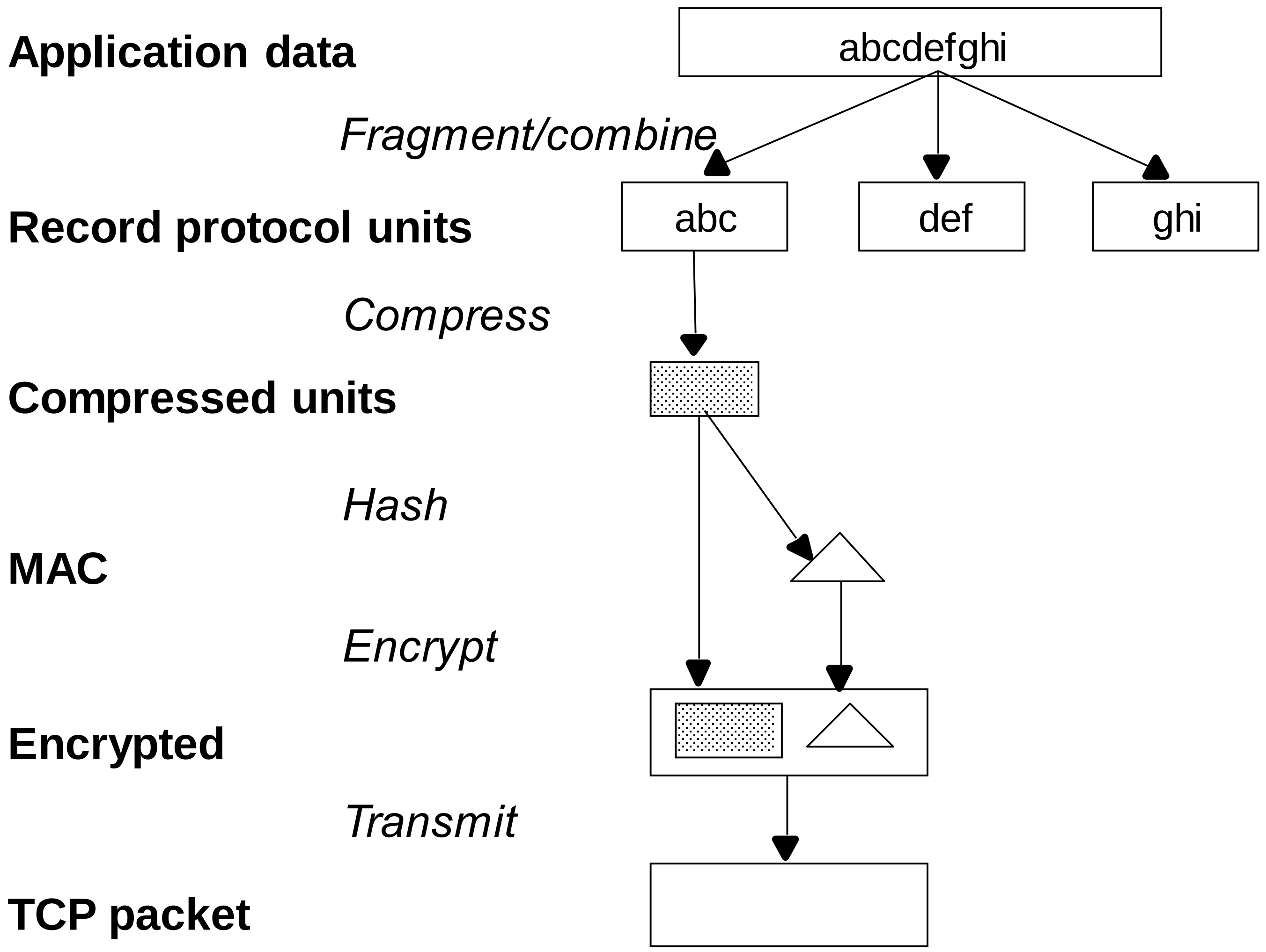
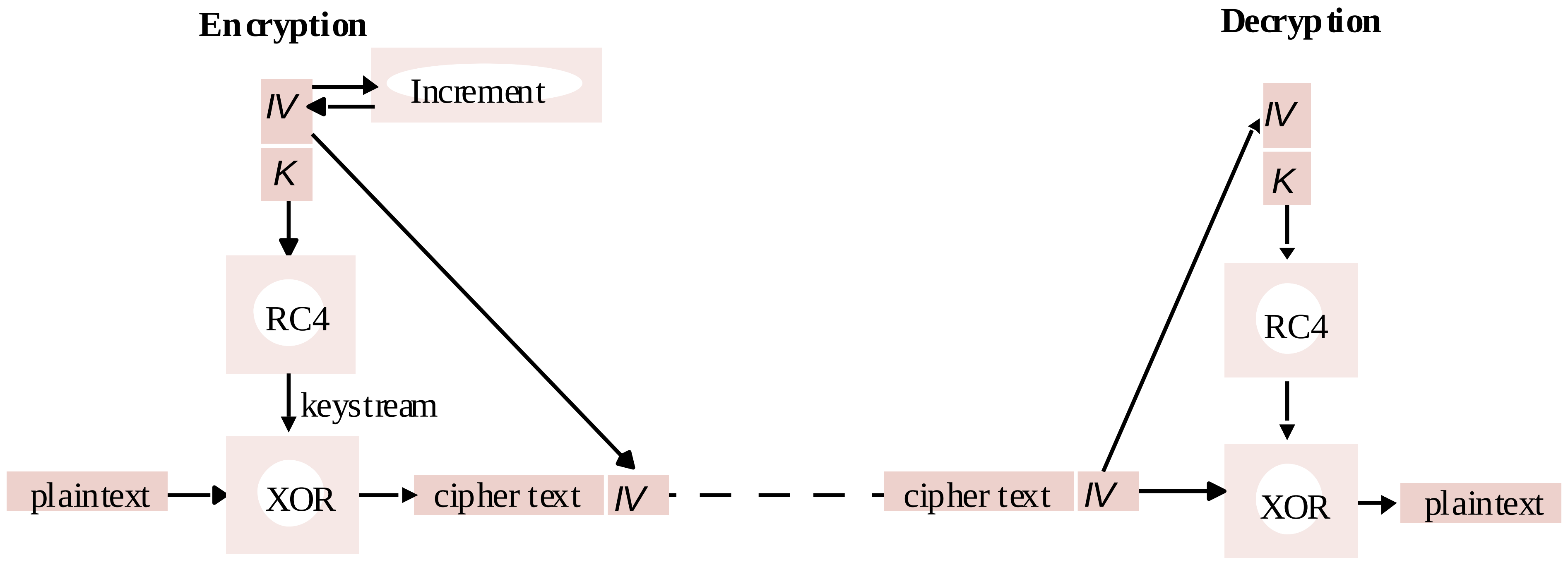


Figure 11.20
Use of RC4 stream cipher in IEEE 802.11 WEP



IV: initial value
K: shared key