Symmetric Cryptography

Alessandro Armando

Computer Security Laboratory (CSec)
DIBRIS, Università di Genova

Computer Security

Outline

Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on a very large alphabet
- 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
- broader range of applications

Outline

- Block ciphers look like an extremely large substitution
- Would need table of 2^n entries for a n-bit block and hence a "key" size of $n \times 2^n$
- A total of $2^n!$ tranformations are possible
- Ideal cipher as statistical information of the plainttext is lost, but infeasible.

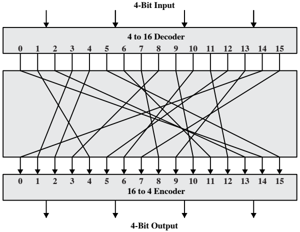


Figure 3.1 General n-bit-n-bit Block Substitution (shown with $n = 4$)

Table 3.1 Encryption and Decryption Tables for Substition Cipher of Figure 3.4

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

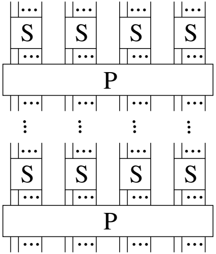
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher Structure

- **Idea:** approximate the ideal block cipher by utilizing the concept of a *product cipher*, i.e. a combination of simple ciphers in such a way that the final result or product is cryptographically stronger that any of the component ciphers.
- In practice: develop a block cipher with a key legth of k bits and a block length of n bits, allowing a total of 2^k possible transformations, rather than the $2^n!$ transformations available with the ideal block cipher
- Most symmetric block ciphers are based on this idea
- Needed since must be able to decrypt ciphertext to recover messages efficiently

Claude Shannon and Substitution-Permutation Ciphers

- Claude Shannon introduced idea of substitution-permutation (S-P) networks in 1949 paper
- form basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations seen before:
  - substitution (S-box)
    confuse input bits.
  - permutation (P-box)
    diffuse bits across S-box inputs
- provide confusion & diffusion
  of message & key



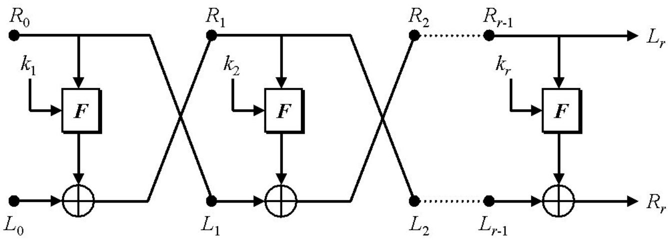
Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message
- a one-time pad does this
- more practically Shannon suggested combining S & P elements to obtain:
  - *diffusion* dissipates statistical structure of plaintext over bulk of ciphertext
  - *confusion* makes relationship between ciphertext and key as complex as possible

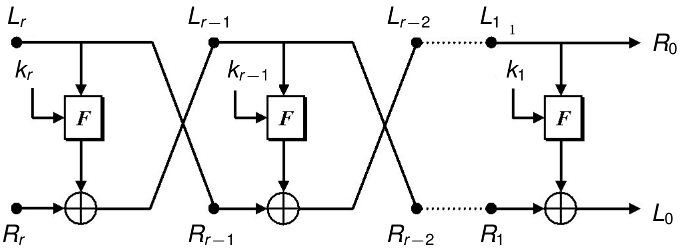
Feistel Cipher Structure

- Horst Feistel devised the feistel cipher
  - based on concept of invertible product cipher
- Partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves
- implements Shannon's S-P net concept

Feistel Cipher Structure

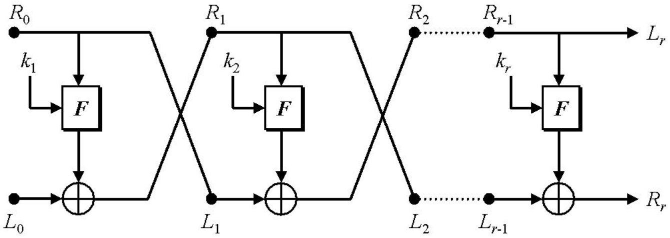


- The round function F can be an S-P network, or any (not necessarily invertible) cipher.
- *Encryption and decryption are structurally identical*, though the subkeys used during encryption at each round are taken in reverse order during decryption.
- More precisely, the input in the decryption algorithm is the pair (R_r, L_r) instead of the pair (L_0, R_0), and the i-th subkey is k_{r-i+1}, not k_i. This means that we obtain (R_{r-i}, L_{r-i}) instead of (L_i, R_i) after the i-th round.



- The round function F can be an S-P network, or any (not necessarily invertible) cipher.
- *Encryption and decryption are structurally identical*, though the subkeys used during encryption at each round are taken in reverse order during decryption.
- More precisely, the input in the decryption algorithm is the pair (R_r, L_r) instead of the pair (L_0, R_0), and the i-th subkey is k_{r-i+1}, not k_i. This means that we obtain (R_{r-i}, L_{r-i}) instead of (L_i, R_i) after the i-th round.

Feistel Cipher Structure: Decryption



$$L_r = R_{r-1} \quad (1)$$

$$R_r = L_{r-1} \oplus F(k_r, R_{r-1}) \quad (2)$$

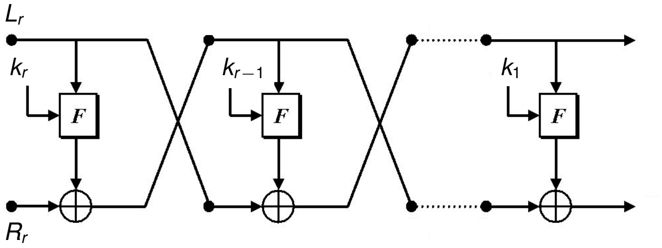
Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$
\begin{aligned}
x \oplus x &= 0 \\
x \oplus 0 &= x \\
x \oplus y &= y \oplus x \\
x \oplus (y \oplus z) &= (x \oplus y) \oplus z
\end{aligned}
$$

Feistel Cipher Structure: Decryption



$$L_r = R_{r-1} \quad (1)$$
$$R_r = L_{r-1} \oplus F(k_r, R_{r-1}) \quad (2)$$

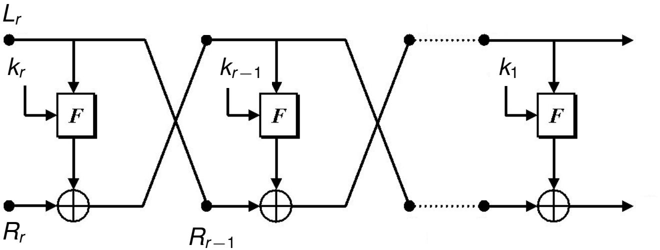
Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$\begin{aligned} x \oplus x &= 0 \\ x \oplus 0 &= x \\ x \oplus y &= y \oplus x \\ x \oplus (y \oplus z) &= (x \oplus y) \oplus z \end{aligned}$$

Feistel Cipher Structure: Decryption



$$
\begin{aligned}
L_r &= R_{r-1} && (1)\\
R_r &= L_{r-1} \oplus F(k_r, R_{r-1}) && (2)
\end{aligned}
$$

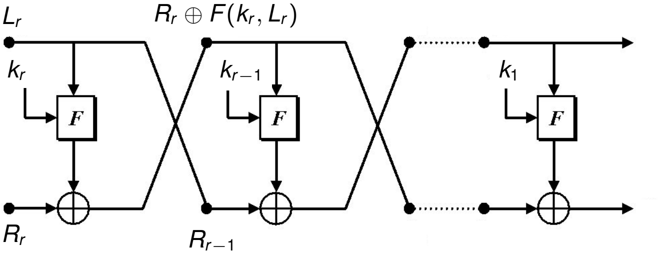
Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$
\begin{aligned}
x \oplus x &= 0\\
x \oplus 0 &= x\\
x \oplus y &= y \oplus x\\
x \oplus (y \oplus z) &= (x \oplus y) \oplus z
\end{aligned}
$$

Feistel Cipher Structure: Decryption



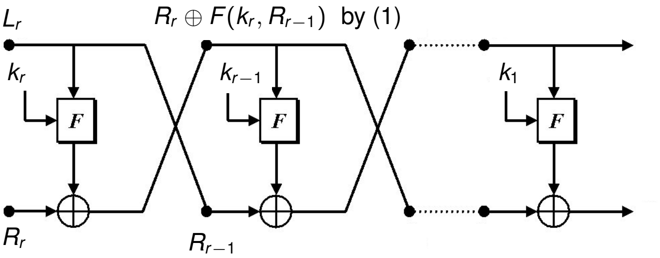
$$L_r = R_{r-1} \quad (1)$$
$$R_r = L_{r-1} \oplus F(k_r, R_{r-1}) \quad (2)$$

Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$x \oplus x = 0$$
$$x \oplus 0 = x$$
$$x \oplus y = y \oplus x$$
$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$



$$\begin{aligned} L_r &= R_{r-1} && (1) \\ R_r &= L_{r-1} \oplus F(k_r, R_{r-1}) && (2) \end{aligned}$$

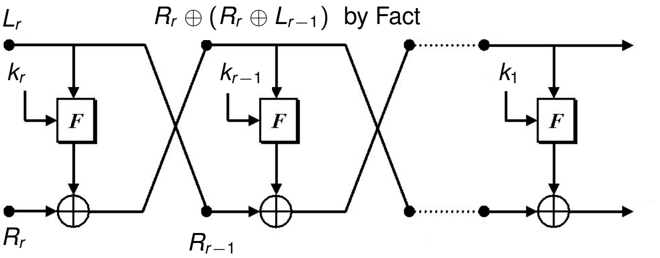
Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$\begin{aligned} x \oplus x &= 0 \\ x \oplus 0 &= x \\ x \oplus y &= y \oplus x \\ x \oplus (y \oplus z) &= (x \oplus y) \oplus z \end{aligned}$$

Feistel Cipher Structure: Decryption



$$
\begin{aligned}
L_r &= R_{r-1} && (1)\\
R_r &= L_{r-1} \oplus F(k_r, R_{r-1}) && (2)
\end{aligned}
$$

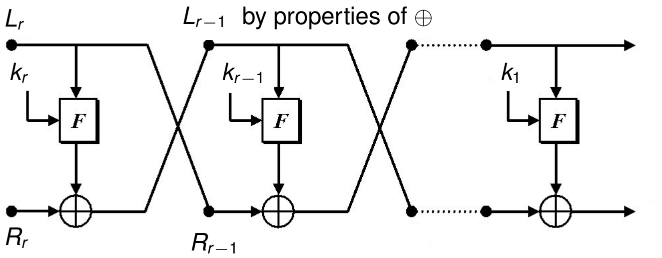
Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$
\begin{aligned}
x \oplus x &= 0\\
x \oplus 0 &= x\\
x \oplus y &= y \oplus x\\
x \oplus (y \oplus z) &= (x \oplus y) \oplus z
\end{aligned}
$$

Feistel Cipher Structure: Decryption



$$
\begin{aligned}
L_r &= R_{r-1} && (1)\\
R_r &= L_{r-1} \oplus F(k_r, R_{r-1}) && (2)
\end{aligned}
$$

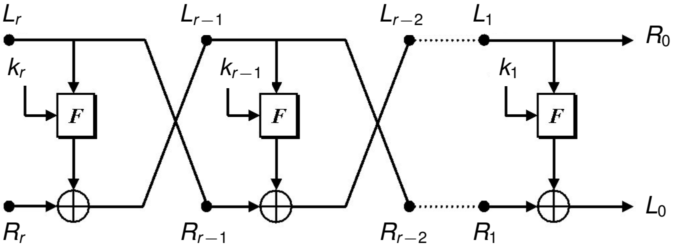
Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

$$
\begin{aligned}
x \oplus x &= 0\\
x \oplus 0 &= x\\
x \oplus y &= y \oplus x\\
x \oplus (y \oplus z) &= (x \oplus y) \oplus z
\end{aligned}
$$

Feistel Cipher Structure: Decryption



$$L_r = R_{r-1} \tag{1}$$

$$R_r = L_{r-1} \oplus F(k_r, R_{r-1}) \tag{2}$$

Fact: $R_r \oplus L_{r-1} = F(k_r, R_{r-1})$

Proof: By putting both sides of (2) in $\oplus$ with L_{r-1} we get:
$R_r \oplus L_{r-1} = L_{r-1} \oplus F(k_r, R_{r-1}) \oplus L_{r-1}$
This equation simplifies to the above fact by exploiting the properties of $\oplus$.

Properties of $\oplus$

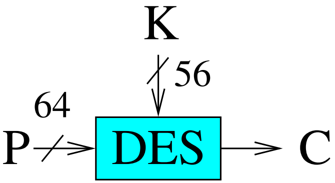
$$\begin{aligned} x \oplus x &= 0 \\ x \oplus 0 &= x \\ x \oplus y &= y \oplus x \\ x \oplus (y \oplus z) &= (x \oplus y) \oplus z \end{aligned}$$

Feistel Cipher Design Elements

- block size
- key size
- number of rounds
- subkey generation algorithm
- round function
- fast software en/decryption
- ease of analysis

DES

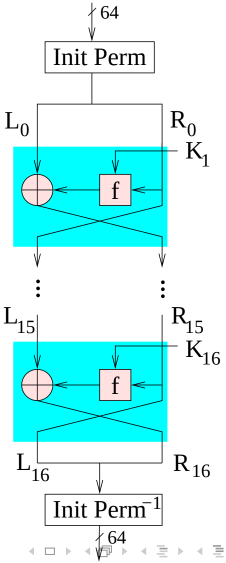
- **Data Encryption Standard**, 1993
  Based on IBM Invention, LUCIFER



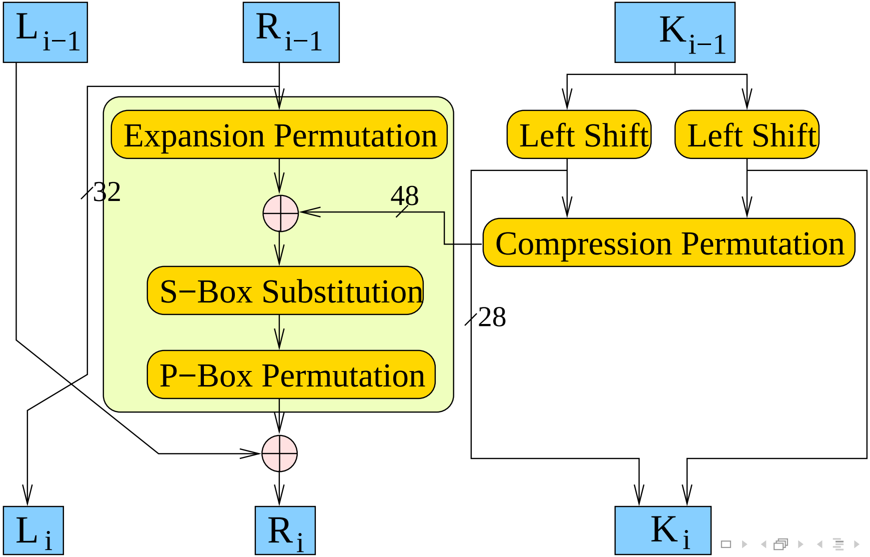
- Block cipher, encrypting 64-bit blocks
  Uses 56 bit keys
  Expressed as 64 bit numbers (8 bits parity checking)
- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard
- Heavily used in banking applications.
  Extensions like triple-DES used to overcome short key-length.

DES – overall form

- 16 rounds Feistel cipher + key-scheduler.
- Key scheduling algorithm derives subkeys K_i from original key K.
- Initial permutation at start, and inverse permutation at end.
- f consists of two permutations and an s-box substitution.



DES – 1 round



L_{i-1}
R_{i-1}
K_{i-1}
Expansion Permutation
Left Shift
Left Shift
32
48
Compression Permutation
S−Box Substitution
28
P−Box Permutation
L_i
R_i
K_i

Avalanche Effect

- key desirable property of encryption algorithm
- where a change of one input or key bit results in changing approx half output bits
- DES exhibits strong avalanche effect

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible
  - in 1997 on Internet in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- must now consider alternatives to DES

Strength of DES – Analytic Attacks

- now have several analytic attacks on DES
- these utilise some deep structure of the cipher
  - by gathering information about encryptions
  - can eventually recover some/all of the sub-key bits
  - if necessary then exhaustively search for the rest
- generally these are statistical attacks
- include
  - differential cryptanalysis
  - linear cryptanalysis
  - related key attacks

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive
- information about some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

Security of DES

People have long questioned the security of DES. There has been much speculation on the key length, number of iterations, and design of the S-boxes. The S-boxes were particularly mysterious – all those constants, without any apparent reason as to why or what they're for. Although IBM claimed that the inner workings were the result of 17 man-years of intensive cryptanalysis, some people feared that the NSA embedded a trapdoor into the algorithm so they would have an easy means of decrypting messages.

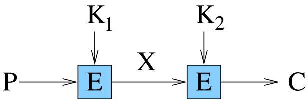
– Bruce Schneier, Applied Cryptography p278.

The National Security Agency also provided technical advice to IBM. And Konheim has been quoted as saying "we sent the S-boxes off to Washington. They came back and were all different. We ran our tests and they passed." People have pointed to this as evidence that the NSA put a trapdoor in DES.

– Bruce Schneier, Applied Cryptography p279.

Increasing DES Security – Double DES

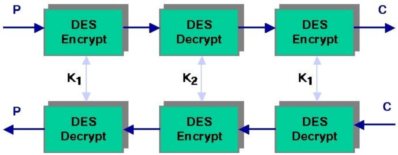
- **Idea**: Perform two encryptions



Equivalent to 112 bit keys?

- **Attack**: Meet-in-the-Middle
  - For $C = E_{K_2}(E_{K_1}(P))$ let $X = E_{K_1}(P) = D_{K_2}(C)$.
  - Given known P and C encrypt P for all 2^{56} possible K_1.
  - Store in table, sorted by X.
  - Decrypt C with all 2^{56} possible K_2 and look for a match.
  - Each hit must be validated against additional plain/cipher-text pair. (For a given plaintext P the average number of different 112-bit keys that will produce a given ciphertext C is $2^{112}/2^{64} = 2^{48}$.)
  - A known plaintext attack against double DES (112 bit keys) succeeds with effort on the order of 2^{56} operations.

Increasing DES Security – Triple DES



- Use three stages of encryption instead of two.
- Notice that K_1 is used twice $\Rightarrow$ 112-bit key.
- Compatibility is maintained with standard DES if $K_2 = K_1$.
- No known practical attack $\Rightarrow$ brute-force search with 2^{112} operations.
- For additional security three-key 3DES is used (e.g. in PGP and S/MIME):

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$

- Backward compatibility with DES with $K_3 = K_2$ and $K_1 = K_2$.

Advanced Encryption Standard (AES)

- Selected as NIST standard in 2001 (out of 15 competing ciphers)
- Based on the Rijndael cipher developed by Belgian cryptographers, Joan Daemen and Vincent Rijmen
- Rijndael is a family of ciphers with different key and block sizes.
- For AES, NIST selected three members of the Rijndael family:
  - block size of 128 bits,
  - three different key lengths: 128, 192 and 256 bits.
- Fast in both software and hardware.
- AES is now used worldwide and supersedes DES

Advanced Encryption Standard (continued)

- AES is based on substitution-permutation network
- Unlike DES, AES does not use a Feistel network.
- While AES has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits, Rijndael works with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.
- The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the plaintext into ciphertext:
  - 10 cycles of repetition for 128-bit keys.
  - 12 cycles of repetition for 192-bit keys.
  - 14 cycles of repetition for 256-bit keys.

Outline

Modes of operation – Electronic Codebook

- How is a block cipher used when messages exceed block-width?
  Different possible modes of operation. Let's consider (just) two!
- Simplest is Electronic Codebook Mode
  Message split into m blocks. Each encrypted individually.
- Limitations:

  Information leak: identical ciphertext blocks map to identical plaintext blocks
  Limited integrity: decryption doesn't indicate if ciphertext blocks have been changed, deleted, or duplicated.

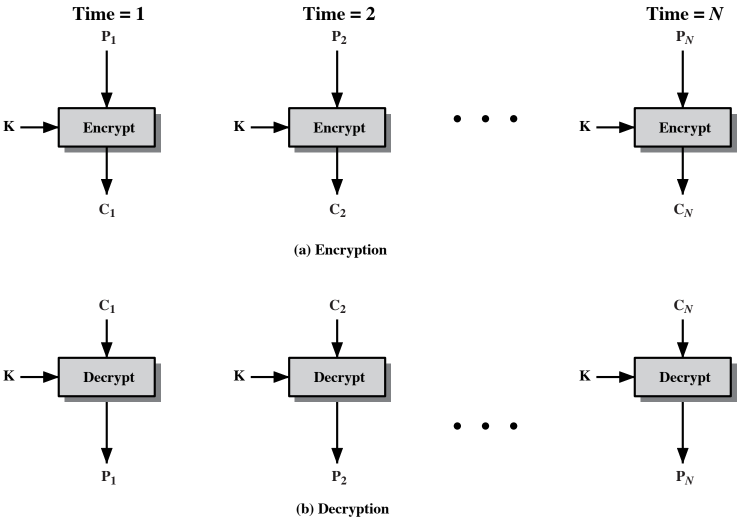


Figure 6.3 Electronic Codebook (ECB) Mode

Modes of operation – Cipher-block Chaining

- Cipher input is XOR of plaintext block with preceding ciphertext.
- For $C_0 = IV$ (an initialization vector), $i = 1..m$:

$$\begin{array}{ll} \text{Encryption:} & C_i = E_K(P_i \oplus C_{i-1}) \\ \text{Decryption:} & P_i = C_{i-1} \oplus D_K(C_i) \end{array}$$

- Correctness?

$$\begin{aligned} P_i &= C_{i-1} \oplus D_K(C_i) \\ &= C_{i-1} \oplus D_K(E_K(P_i \oplus C_{i-1})) \\ &= C_{i-1} \oplus (P_i \oplus C_{i-1}) \\ &= P_i \end{aligned}$$

- Properties
  - Identical plaintext blocks mapped to different ciphertext
  - *Chaining dependencies:* C_j depends on all preceding plaintext.
  - *Self-synchronizing:* if an error occurs (changed bits, dropped blocks) in C_j but not C_{j+1}, then C_{j+2} is correctly decrypted.

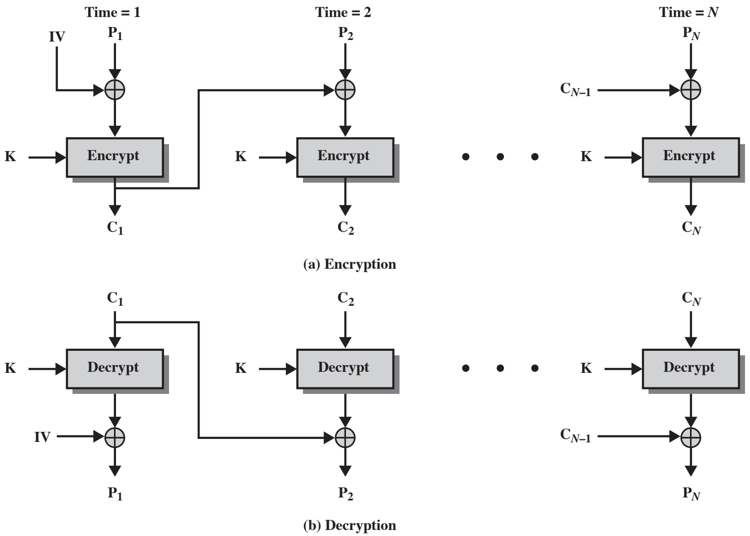


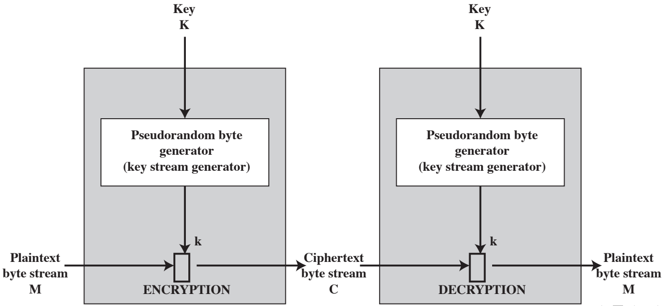
Figure 6.4 Cipher Block Chaining (CBC) Mode

Stream Ciphers

- Same idea of Vernam cipher but use pseudorandom generator (in place of a truly random generator), i.e.

$$
\begin{aligned}
E_{k_1\cdots k_n}(m_1\cdots m_n) &= (m_1\oplus k_1)\cdots(m_n\oplus k_n)\\
D_{k_1\cdots k_n}(c_1\cdots c_n) &= (c_1\oplus k_1)\cdots(c_n\oplus k_n)
\end{aligned}
$$

- Use seed as key



Cipher	Key Length	Speed (Mbps)
DES	56	9
3DES	168	3
RC4	variable	45

- Stream ciphers are usually faster and easier to implement
- With a properly designed pseudorandom number generator, a stream cypher can be as secure as a block cipher of comparable key length
- With block ciphers keys can be reused
- With stream ciphers if two plaintexts, p_i and p'_i with $i = 1, 2, \ldots$, are encrypted with the same key using a stream cipher, i.e. $c_i = p_i \oplus k_i$ and $c'_i = p'_i \oplus k_i$, then $c_i \oplus c'_i = p_i \oplus p'_i$.

RC4

- Designed in 1987 by Rivest, trade secret anounymously posted on the Internet in 1994
- Used in SSL/TLS, WEP, and WiFi Protected Access (WPA)

```
#define SWAP(a,b) (((a) ^ (b)) && ((b) ^= (a) ^= (b), (a) ^= (b)))
int main() {
  unsigned char S[256], key[]="Key";
  unsigned short i, j, keylength=3;
  for(i=0;i<256;i++)
    S[i]=i;
  j=0;
  for(i=0;i<256;i++) {
    j=(j+S[i]+key[i%keylength])%256;
    SWAP(S[i],S[j]);
  }
  int K;
  i=j=0;
  while(1) {
    i=(i+1)%256; j=(j+S[i])%256; SWAP(S[i],S[j]);
    K=S[(S[i]+S[j])%256];
    printf("%X",K);
```

Outline

Placement of Encryption

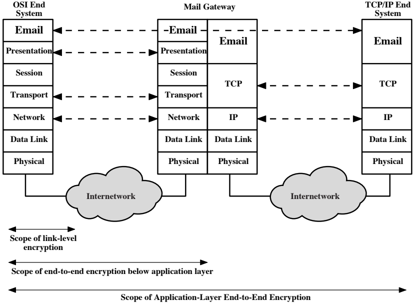
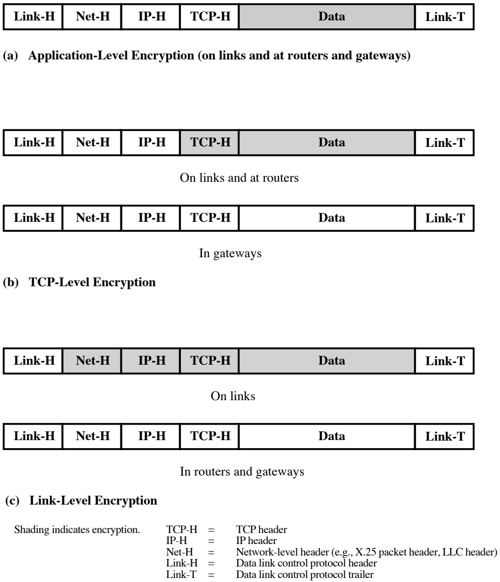


Figure 7.4 Encryption Coverage Implications of Store-and-Forward Communications

- Encryption can be placed at various layers in the OSI Reference Model
- link encryption at layers 1 or 2
- end-to-end encryption at layers 3, 4, 6, 7
- as we move higher in the OSI stack
  - less information is encrypted
  - more secure, but
  - more complex and with more entities and keys

Placement of Encryption (continued)

- When using end-to-end encryption headers must be left in clear
  - so network can correctly route information
- Hence, although contents protected, traffic pattern flows are not.
- Ideally we want both at once
  - end-to-end encryption protects data contents over entire path and provides authentication
  - link protects traffic flows from monitoring


Link-H
Net-H
IP-H
TCP-H
Data
Link-T
(a) Application-Level Encryption (on links and at routers and gateways)
Link-H
Net-H
IP-H
TCP-H
Data
Link-T
On links and at routers
Link-H
Net-H
IP-H
TCP-H
Data
Link-T
In gateways
(b) TCP-Level Encryption
Link-H
Net-H
IP-H
TCP-H
Data
Link-T
On links
Link-H
Net-H
IP-H
TCP-H
Data
Link-T
In routers and gateways
(c) Link-Level Encryption
Shading indicates encryption.
TCP-H = TCP header
IP-H = IP header
Net-H = Network-level header (e.g., X.25 packet header, LLC header)
Link-H = Data link control protocol header
Link-T = Data link control protocol trailer

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host Message exposed in intermediate nodes	Message encrypted in sending/receiving host Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host Transparent to user Host maintains encryption facility One facility for all users Can be done by hardware All or no messages encrypted	Applied by sending/receiving process User applies encryption User must determine algorithm Users selects encryption scheme Software implementation User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair Provides host authentication	Requires one key per user pair Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host Message exposed in intermediate nodes	Message encrypted in sending/receiving host Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host Transparent to user Host maintains encryption facility One facility for all users Can be done by hardware All or no messages encrypted	Applied by sending/receiving process User applies encryption User must determine algorithm Users selects encryption scheme Software implementation User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair Provides host authentication	Requires one key per user pair Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host Message exposed in intermediate nodes	Message encrypted in sending/receiving host Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host Transparent to user Host maintains encryption facility One facility for all users Can be done by hardware All or no messages encrypted	Applied by sending/receiving process User applies encryption User must determine algorithm Users selects encryption scheme Software implementation User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair Provides host authentication	Requires one key per user pair Provides user authentication

Link vs End-to-End Encryption: Summary

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending/receiving host	Message encrypted in sending/receiving host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending/receiving host	Applied by sending/receiving process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done by hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

Traffic Analysis

- Monitoring of communications flows between parties
  - useful both in military & commercial spheres
  - can also be used to create a covert channel
- Link encryption obscures header details
  - but overall traffic volumes in networks and at end-points is still visible
- Traffic padding can further obscure flows
  - but at cost of continuous traffic

Outline

Key Distribution

- Symmetric schemes require both parties to share a common secret key.
- The issue is how to securely distribute this key.
- Often secure system failure is due to a break in the key distribution scheme.

Approches to Key Distribution

Given parties A and B have various key distribution alternatives:

- A can select key and physically deliver to B
- Third party can select and deliver key to A and B
- If A and B have communicated previously can use previous key to encrypt a new key
- If A and B have secure communications with a third party C, C can relay key between A and B

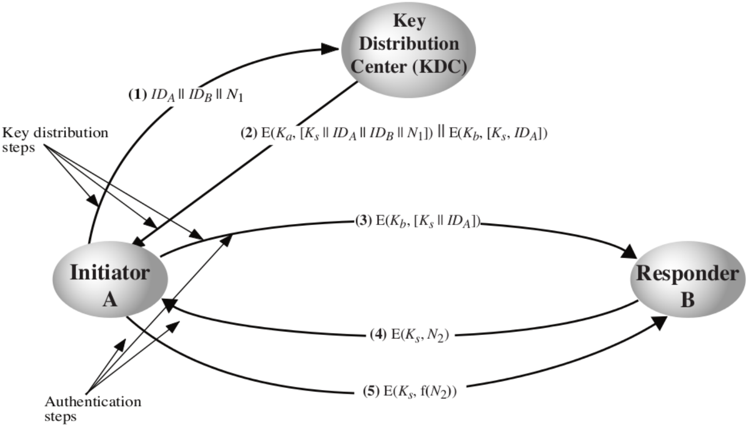
Key Hierarchy

Typically a hierarchy of keys is used:

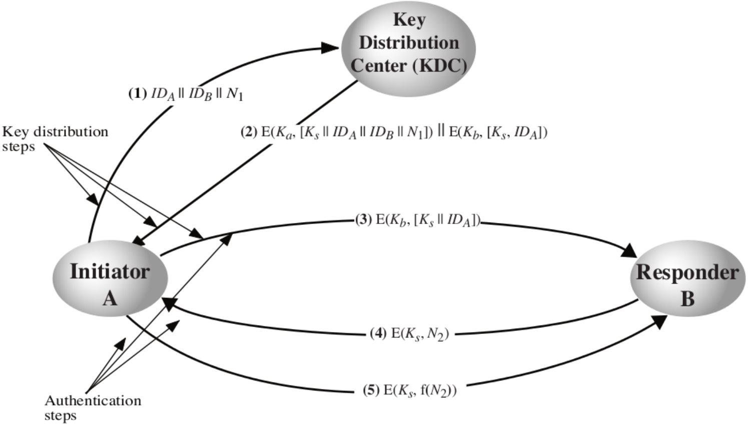
Session key: used for encryption of data between users for one logical session then discarded

Master key: used to encrypt session keys
shared by user and key distribution center

Key Distribution Scenario



Key Distribution Scenario



Weakness: B cannot check freshness of K_s. If K_s is compromised, then it can be replayed by the attacker.

Key Distribution Issues

- Hierarchies of KDC's required for large networks, but must trust each other
- Session key lifetimes should be limited for greater security
- Use of automatic key distribution on behalf of users, but must trust system
- Use of decentralized key distribution
- Controlling key usage