COM 5335 Network Security
Lecture 2
Symmetric Cryptography

Scott CH Huang

Overview

1. Basic concepts & Block ciphers
2. Case study (block cipher) - DES
3. Stream ciphers
4. Case study (stream cipher) - RC4
5. Block cipher modes of operation
6. Cryptanalysis

Basic Concepts & Block Ciphers

Modern Block Ciphers

- One of the most widely used types of cryptographic algorithms
- Provide secrecy /authentication services
- Messages are processed in blocks

Block vs Stream Ciphers

- Block ciphers process messages in blocks, each of which is encrypted/decrypted.
- They behave like a substitution table on very big characters. ~ 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

Block Cipher Principles

- Many symmetric block ciphers are based on a **Feistel Cipher Structure**
- Feistel structure: **decrypt** ciphertext is very similar to **encrypt** plaintext
- Block ciphers look like an extremely large substitution
- Would need table of 2^{64} entries for a 64-bit block
- Instead create from smaller building blocks
- Using idea of a product cipher

Substitution & Permutation Networks

- 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)
  - *permutation* (P-box)
- Provide *confusion* & *diffusion* of message & key

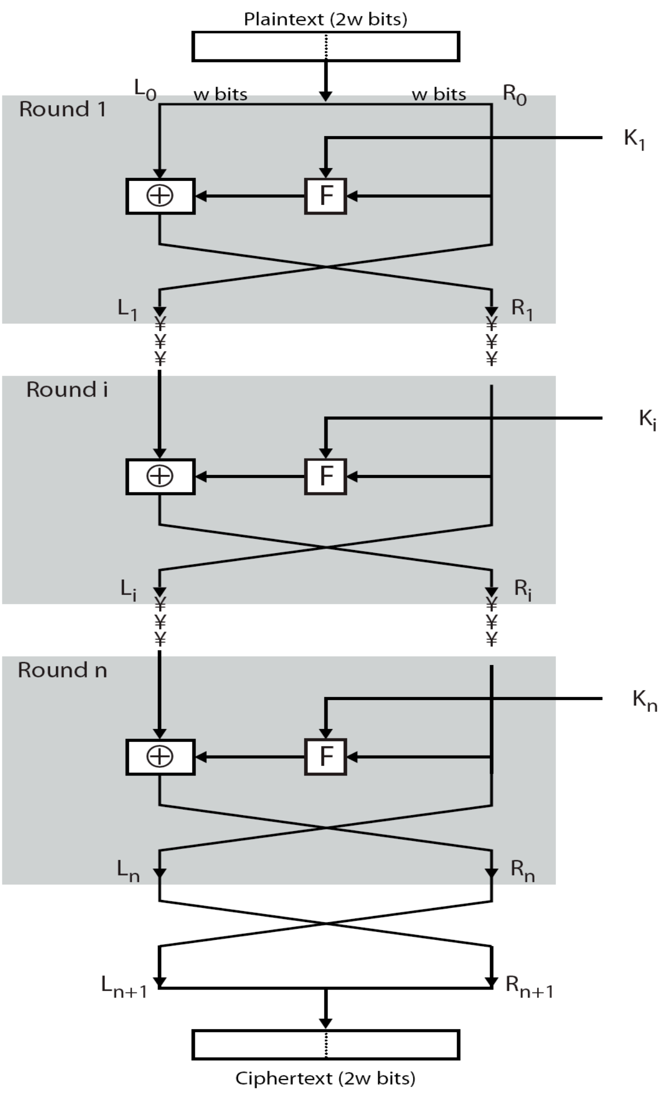
Confusion and Diffusion

- 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 structure**
  - 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

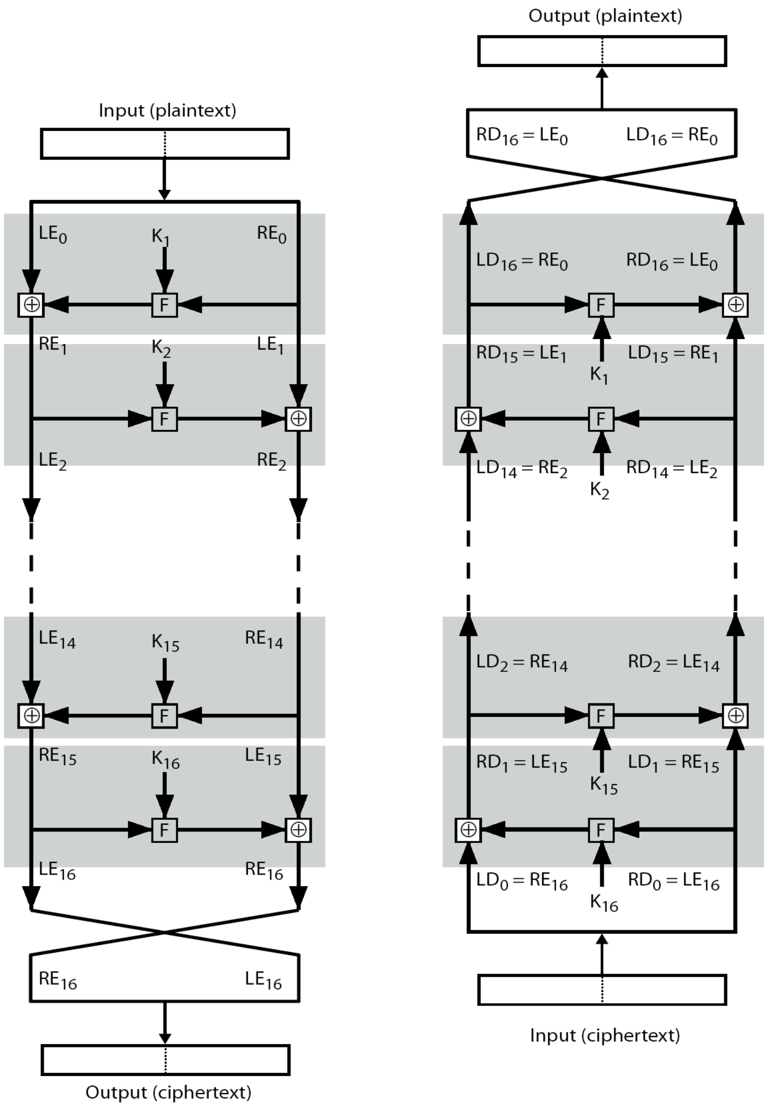
Feistel Cipher Structure



Feistel Cipher Design Elements

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

Feistel Cipher Decryption



Case Study – DES

Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security

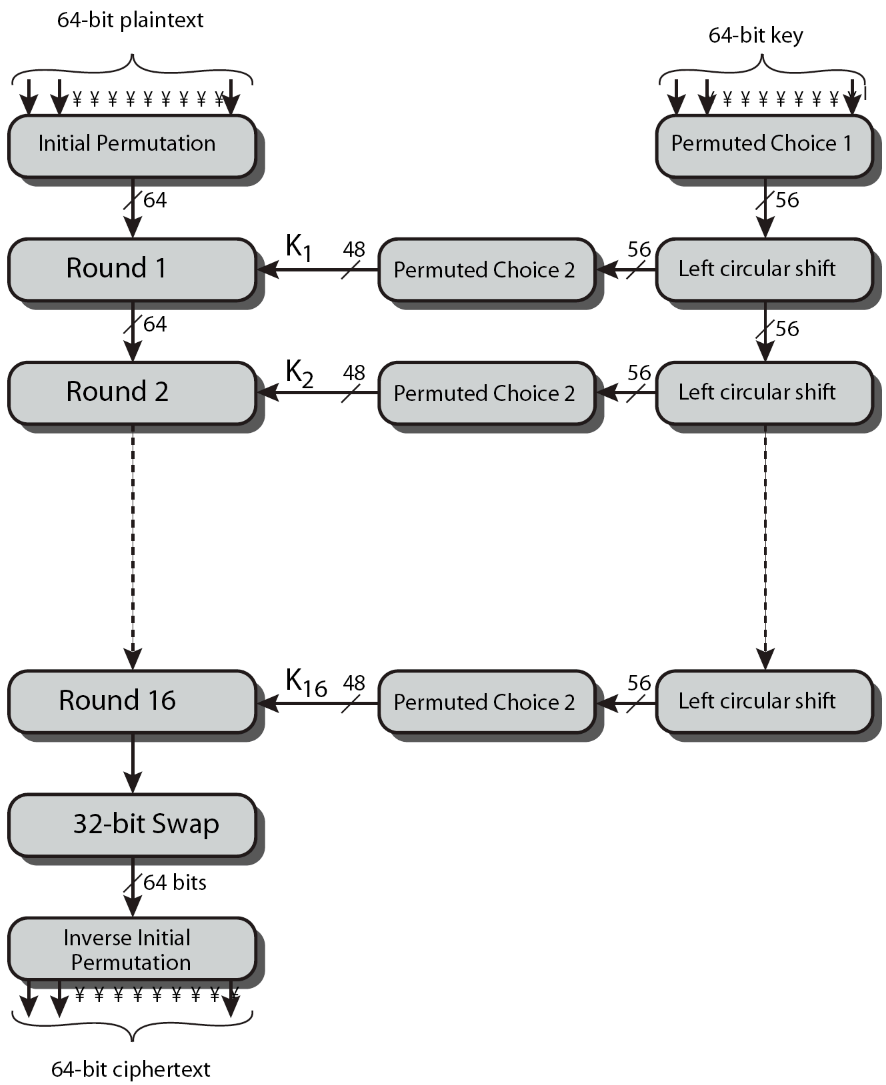
DES History

- IBM developed Lucifer cipher
  - by team led by Feistel in late 60's
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

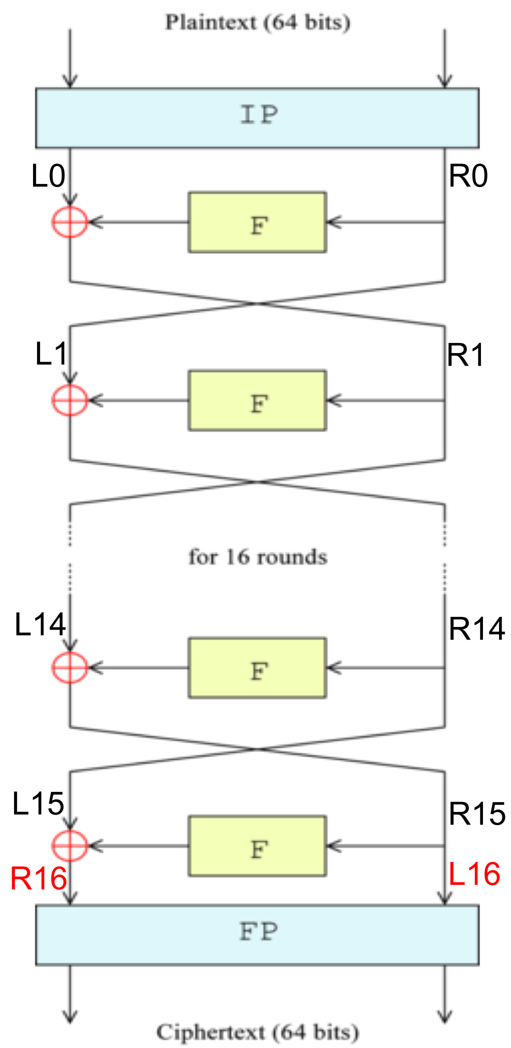
DES Design Controversy

- Although DES standard is public
- Was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- Subsequent events and public analysis show in fact design was appropriate
- Use of DES has flourished
  - especially in financial applications
  - still standardised for legacy application use

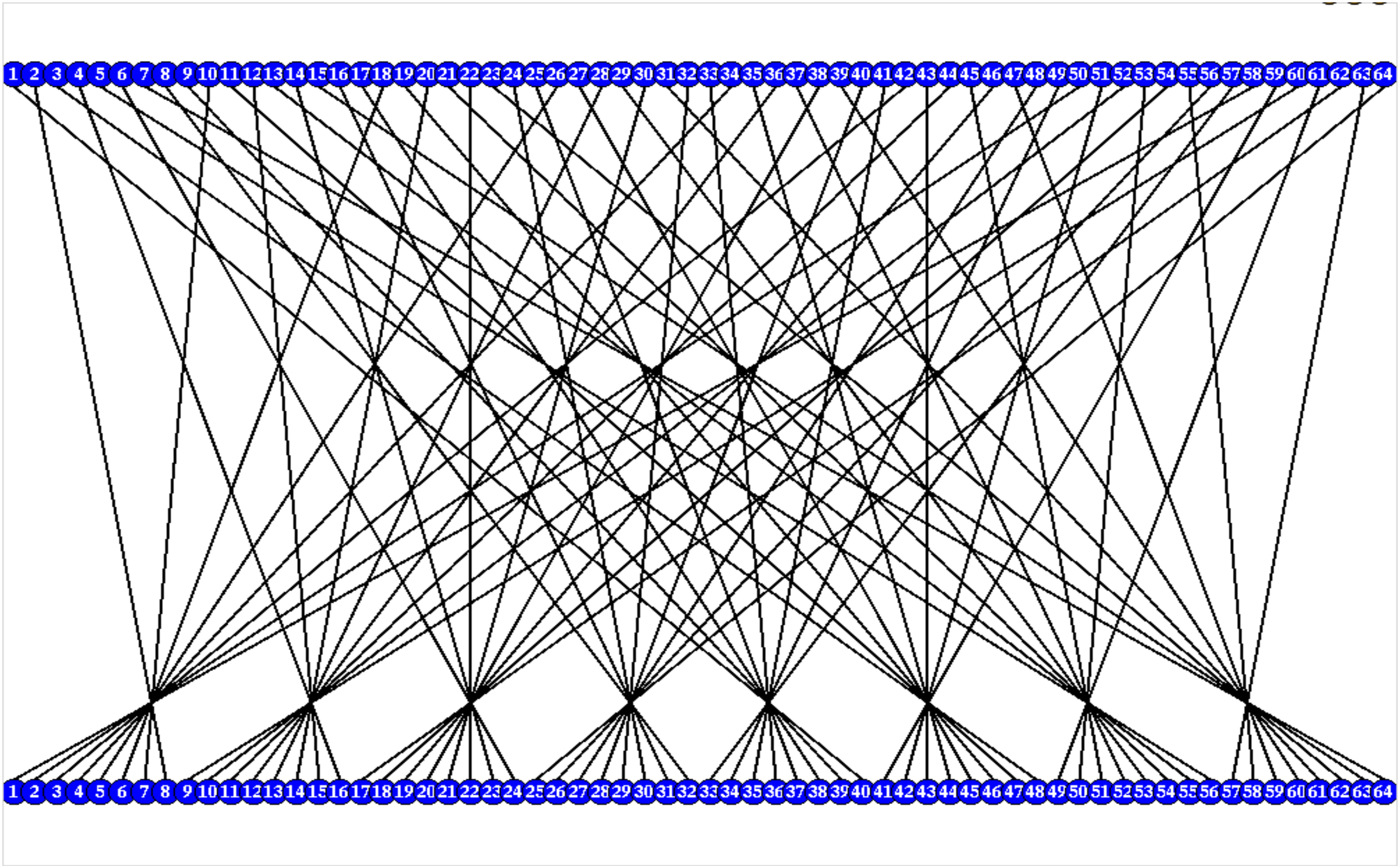
DES Encryption Overview



DES Encryption



Initial Permutation - IP



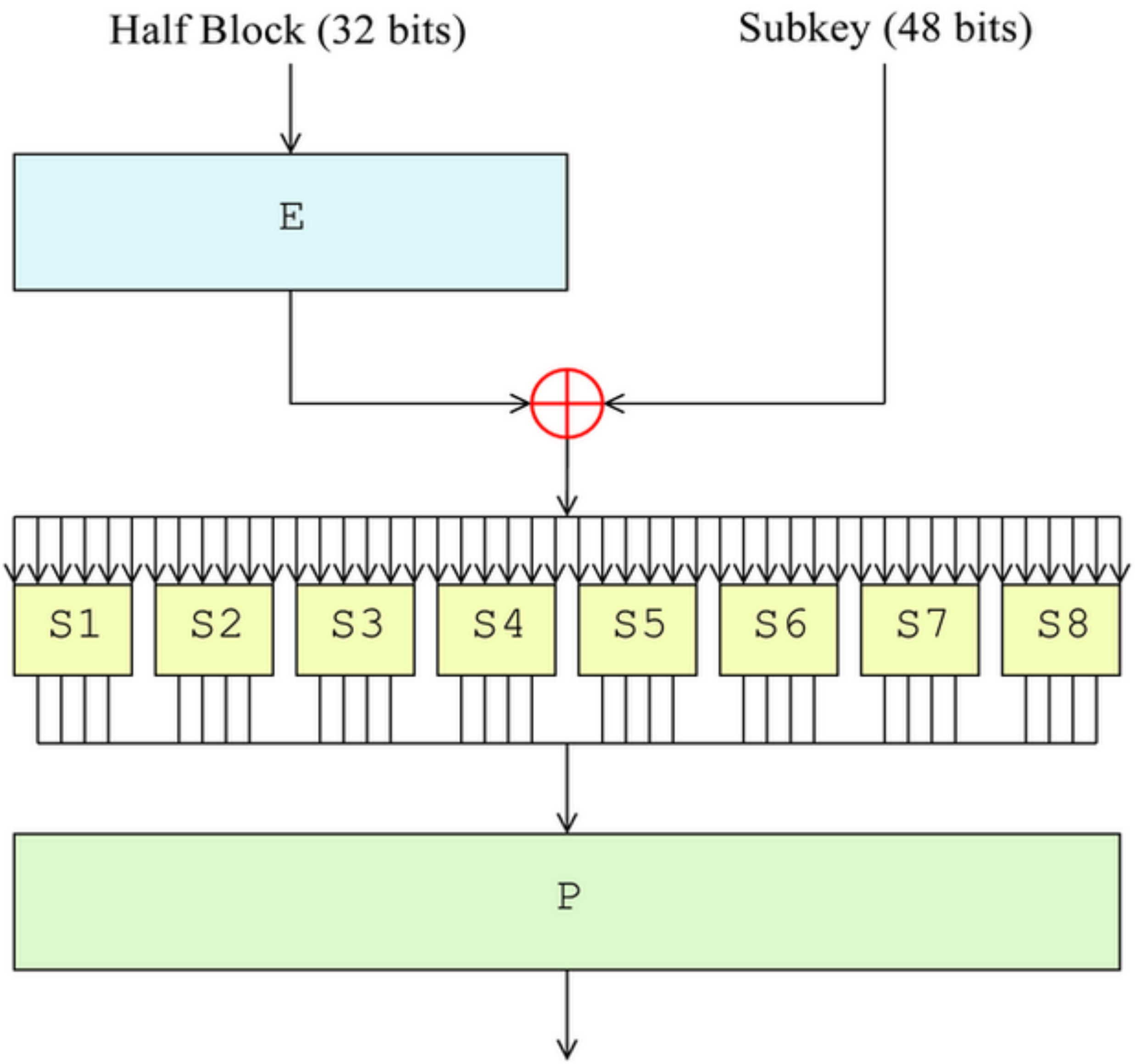
IP Table

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Final Permutation - IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Feistel function F in DES



Expansion Permutation - E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Substitution Boxes - S

- have eight S-boxes which map 6 to 4 bits
- each S-box is a 4-by-16 table
  - outer bits 1 & 6 (**row** bits) select one row from 4
  - inner bits 2-5 (**col** bits) select one col from 16
  - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
- Show the S-boxes from DES-tables

S-Boxes - S1

S1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

- Example:
  - Input= 011001
  - Row = 01=1
  - Column=1100=12
  - Output=9=1001

S2 - S4

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5 - S7

S5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES Round Structure

- Uses two 32-bit L & R halves
- As for any Feistel cipher can describe as:

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \oplus \mathrm{F}(R_{i-1}, K_i)$
- F takes 32-bit R half and 48-bit subkey:
  - expands R to 48-bits using perm E
  - adds to subkey using XOR
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes using 32-bit perm P

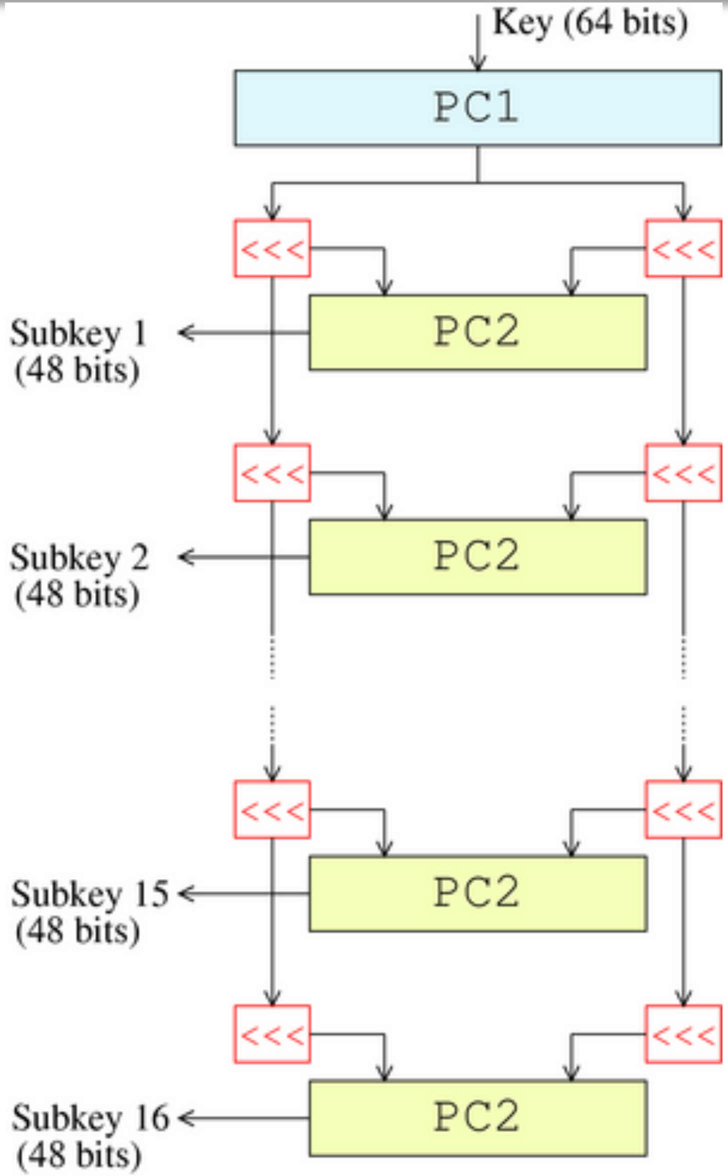
Permutation – P (in F- function, 32 bits)

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

DES Key Schedule

- Forms subkeys used in each round
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of:
    - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K
    - selecting 24-bits from each half & permuting them by PC2 for use in round function F
- Note practical use issues in h/w vs s/w

DES Key Schedule



Permutation Choice 1 - PC-1

Left						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
Right						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Rotations in Key Schedule

Round number	Number of left rotations
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

DES Decryption

- Decrypt must unwind steps of data computation
- With Feistel design, do encryption steps again Using subkeys in reverse order (SK16 … SK1)
  – IP undoes final FP step of encryption
  – 1st round with SK16 undoes 16th encrypt round
  – ….
  – 16th round with SK1 undoes 1st encrypt round
  – then final FP undoes initial encryption IP
  – thus recovering original data value

Avalanche Effect

- Key desirable property of encryption alg
- Where a change of **one** input or key bit results in changing approx **half** output bits
- Making attempts to “home-in” by guessing keys impossible
- DES exhibits strong avalanche

Strength of DES - Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- Brute force search is hard, but (more and more) feasible
- 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

DES Design Criteria

- As reported by Coppersmith in [COPP94]
- 7 criteria for S-boxes provide for
  - non-linearity
  - resistance to differential cryptanalysis
  - good confusion
- 3 criteria for permutation P provide for
  - increased diffusion

Triple DES

- Clearly a replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- Prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

Why Triple-DES?

- Why not Double-DES?
  - NOT same as some other single-DES use, but have
- Meet-in-the-middle attack
  - works whenever use a cipher twice
  - since $X = E_{K1}[P] = D_{K2}[C]$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes $O(2^{56})$ steps

Triple-DES with Two-Keys

- Hence must use 3 encryptions
  - would seem to need 3 distinct keys
- But can use 2 keys with E-D-E sequence
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - nb encrypt & decrypt equivalent in security
  - if K1=K2 then can work with single DES
- Standardized in ANSI X9.17 & ISO8732
- No current known practical attacks

Triple-DES with Three-Keys

- Although are no practical attacks on two-key Triple-DES have some indications
- Can use Triple-DES with Three-Keys to avoid even these
  – $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
- Has been adopted by some Internet applications, eg PGP, S/MIME

Block Cipher Characteristics

- Features seen in modern block ciphers are:
  - variable key length / block size / no rounds
  - mixed operators, data/key dependent rotation
  - key dependent S-boxes
  - more complex key scheduling
  - operation of full data in each round
  - varying non-linear functions
- Contemporary block ciphers:
  - DES,IDEA,Blowfish,RC5, RC6

Stream Ciphers

Stream Cipher Basics

- Process the message bit by bit (as a stream)
- Typically have a (pseudo) random **stream key**
- XOR with plaintext bit by bit (Vernam Cipher!)
- Randomness of **stream key** completely destroys any statistically properties in the message
  - $C_i = M_i \text{ XOR } StreamKey_i$
- Never reuse stream key
  - otherwise can remove effect and recover messages

Stream Cipher Properties

- Design considerations:
  - long period with no repetitions
  - statistically random
  - depends on large key
  - large linear complexity
  - correlation immunity
  - confusion
  - diffusion
  - use of highly non-linear boolean functions

Case Study - RC4

RC4

- Ron’s Code #4 (RC2, RC5, RC6)
- A proprietary cipher owned by RSA DSI
- Simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input info processed a byte at a time

RC4 Initialization

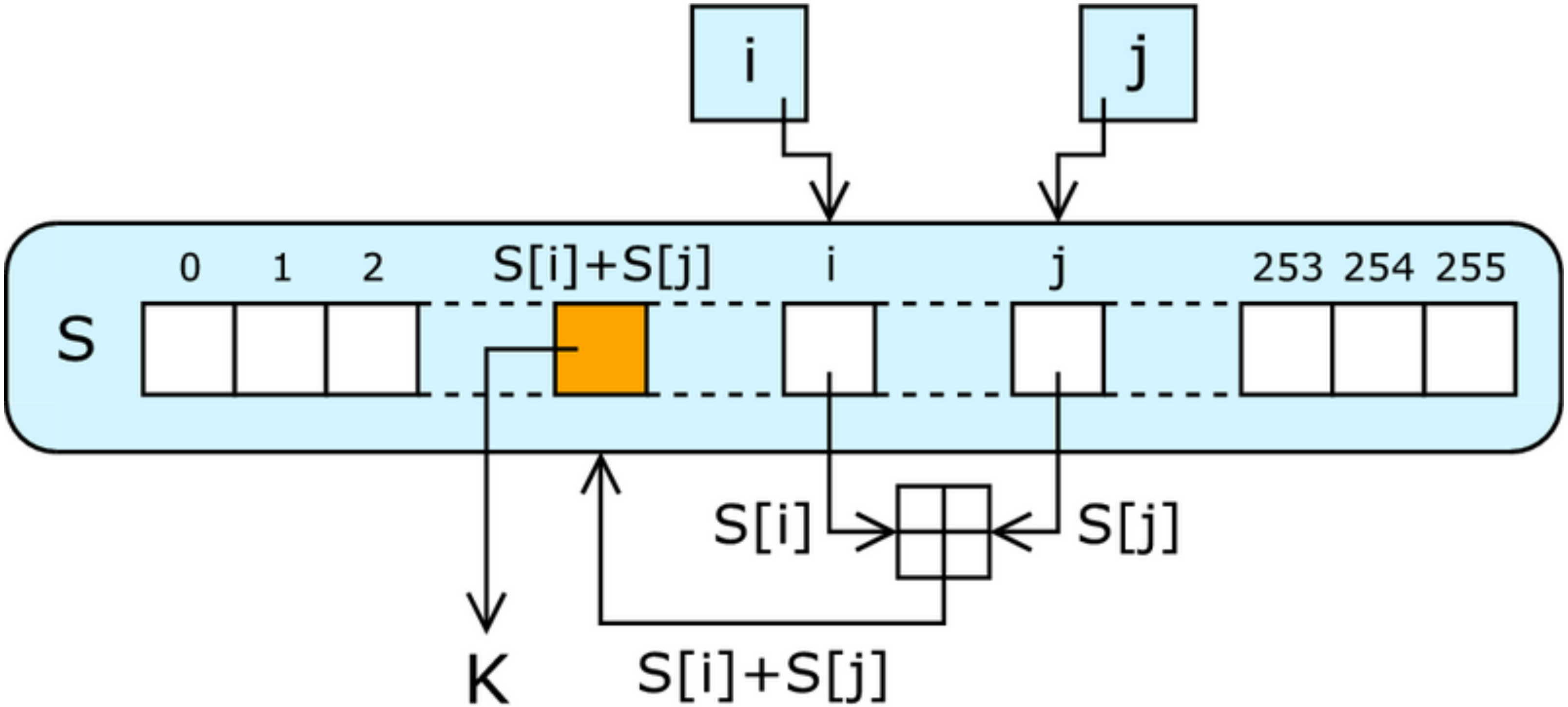
- Initialize two arrays S[256] and T[256]
- S[0]=0,S[1]=1,…,S[i]=i,…,S[255]=255
- Secret key K[0],K[1],…,K[keylen-1], each of which is 1-byte
- T[0]=K[0],…,T[keylen-1]=K[keylen-1]
- T[keylen]=K[0],…,T[i]=K[I mod keylen]
- Normally, 5<keylen<16

RC4 Key Schedule

- Use key to well and truly shuffle
- S forms **internal state** of the cipher

```
for i = 0 to 255 do
    S[i] = i
j = 0
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256
    swap (S[i], S[j])
```

RC4 Key Schedule



$j = (j + S[i] + T[i]) \bmod 256$

swap (S[i], S[j])

RC4 Encryption

- encryption continues shuffling array values
- sum of shuffled pair selects "stream key" value
- XOR with next byte of message to en/decrypt

```
i = j = 0
for each message byte M_i
while{
    i = (i + 1) mod 256
    j = (j + S[i]) mod 256
    swap(S[i], S[j])
    t = (S[i] + S[j]) mod 256
    C_i = M_i XOR S[t]
}
```

A Short Example

- Size of array = 4 (instead of 256)
- keylen = 2
- S={0,1,2,3}
- K={2,5}
- T={2,5,2,5}

Initializing

- i=j=0, S={0,1,2,3}, T={2,5,2,5}
- j=j+S[i]+T[i] mod 4
- =0+S[0]+T[0] mod 4 = 2
- swap S[0], S[2]
- S={2,1,0,3}
- now i=i+1=1
- j=2+S[1]+T[1] mod 4 = 2+1+5 mod 4= 0
- swap S[1], S[0]

Encryption: PRN generation

- Finally S={1,2,3,0}
- now i=j=0, for each 8-bit word
- i=i+1 mod 4 =1
- j=j+S[i] mod 4 = 0+2 mod 4=2
- swap S[1],S[2] and S={1,3,2,0}
- t=S[1]+S[2] mod 4 =2+3 mod 4 =1
- Generate a 8-bit PRN S[1]=3=0000 0011

RC4 Security

- claimed secure against known attacks
  - have some analyses, none practical
- result is very non-linear
- since RC4 is a stream cipher, must **never reuse a key**
- have a concern with WEP, but due to key handling rather than RC4 itself

Block Cipher
Modes of Operation

Modes of Operation

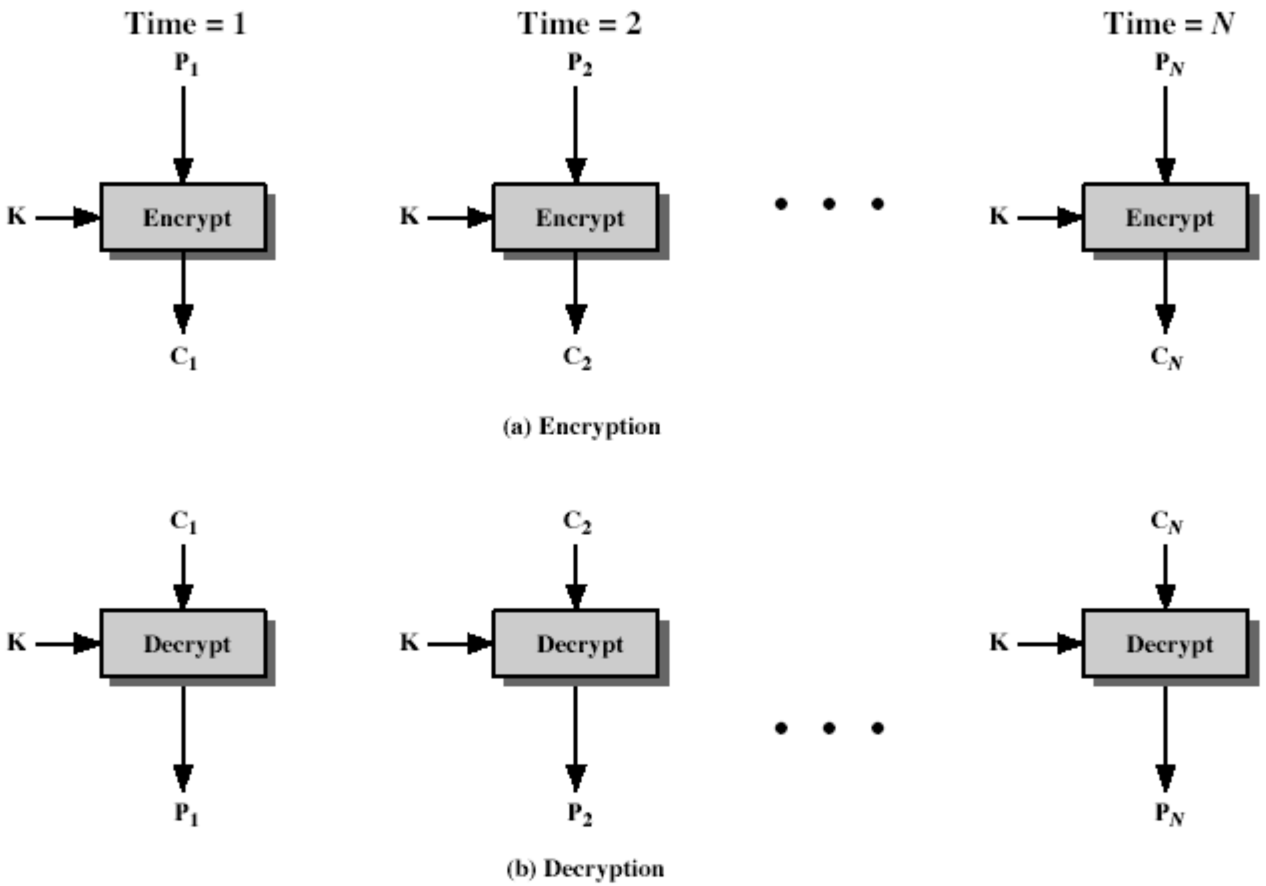
- block ciphers encrypt fixed size blocks
- eg. DES encrypts 64-bit blocks, with 56-bit key
- need way to use in practise, given usually have arbitrary amount of information to encrypt
- four were defined for DES in ANSI standard **ANSI X3.106-1983 Modes of Use**
- subsequently now have 5 for DES and AES
- have **block** and **stream** modes

Electronic Codebook (ECB)

- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

  $C_i = DES_{K1} (P_i)$
- uses: secure transmission of single values

ECB Mode


Time = 1
Time = 2
Time = N
P1
P2
PN
K
Encrypt
C1
C2
CN
(a) Encryption
Decrypt
(b) Decryption

Advantages and Limitations of ECB

- repetitions in message may show in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness due to encrypted message blocks being independent
- main use is sending a few blocks of data

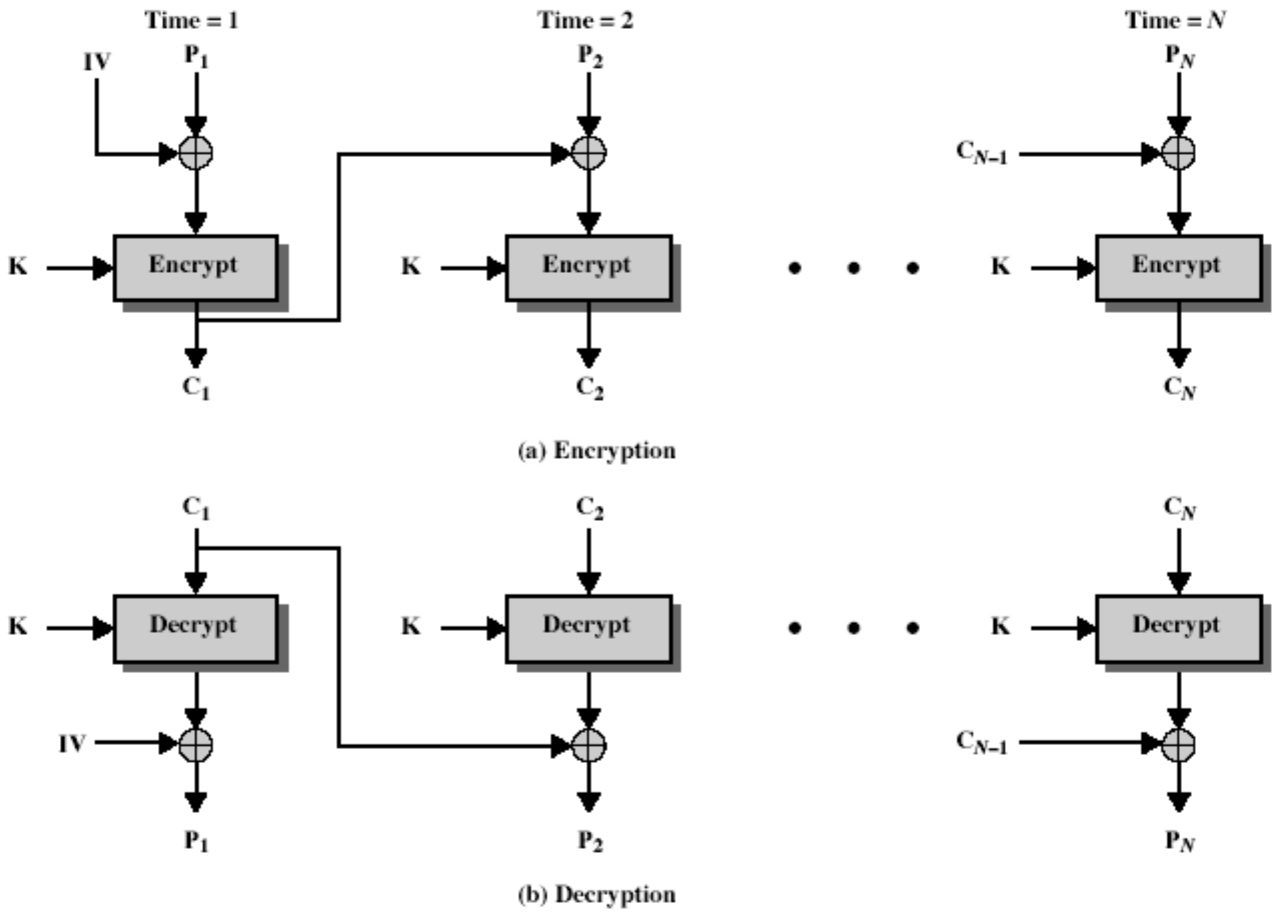
Cipher Block Chaining (CBC)

- message is broken into blocks
- but these are linked together in the encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

  $C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$

  $C_{-1} = IV$
- uses: bulk data encryption, authentication

CBC Mode



Time = 1
Time = 2
Time = N
IV
P1
P2
PN
CN–1
K
Encrypt
C1
C2
CN
(a) Encryption
C1
C2
CN
K
Decrypt
IV
CN–1
P1
P2
PN
(b) Decryption

Advantages and Limitations of CBC

- each ciphertext block depends on **all** message blocks
- thus a change in the message affects all ciphertext blocks after the change as well as the original block
- need **Initial Value** (IV) known to sender & receiver
  – however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
  – hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
- at end of message, handle possible last short block
  – by padding either with known non-data value (eg nulls)
  – or pad last block with count of pad size
    - eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count

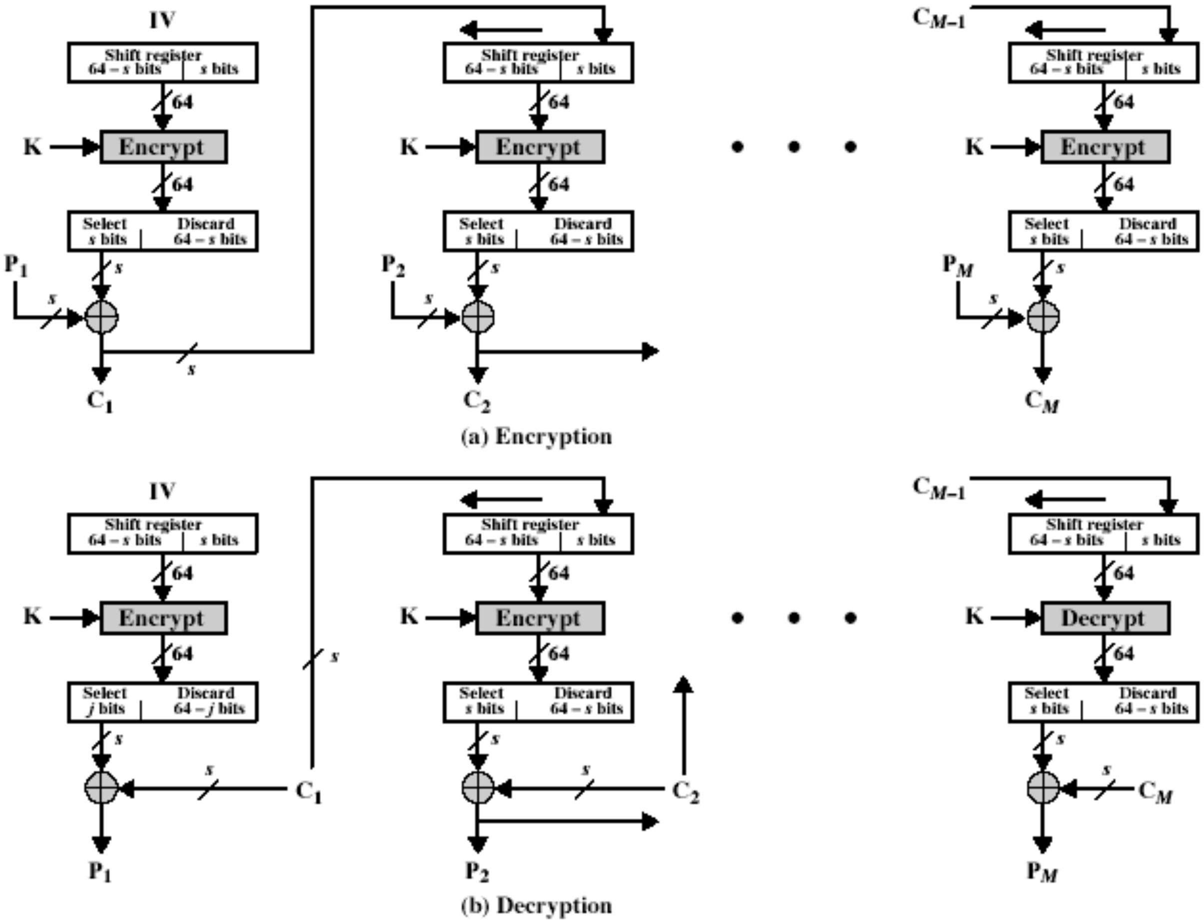
Cipher Feedback (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8 or 64 or whatever) to be feed back
  - denoted CFB-1, CFB-8, CFB-64 etc
- is most efficient to use all 64 bits (CFB-64)

  $C_i = P_i \text{ XOR } DES_{K1}(C_{i-1})$

  $C_{-1} = IV$
- uses: stream data encryption, authentication

CFB Mode


IV
Shift register
64 – s bits
s bits
K
Encrypt
64
Select
s bits
Discard
64 – s bits
P_1
s
C_1
P_2
C_2
C_{M-1}
P_M
C_M
(a) Encryption
Select
j bits
Discard
64 – j bits
Decrypt
(b) Decryption

Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in **encryption** mode at **both** ends
- errors propogate for several blocks after the error

Output Feedback (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance

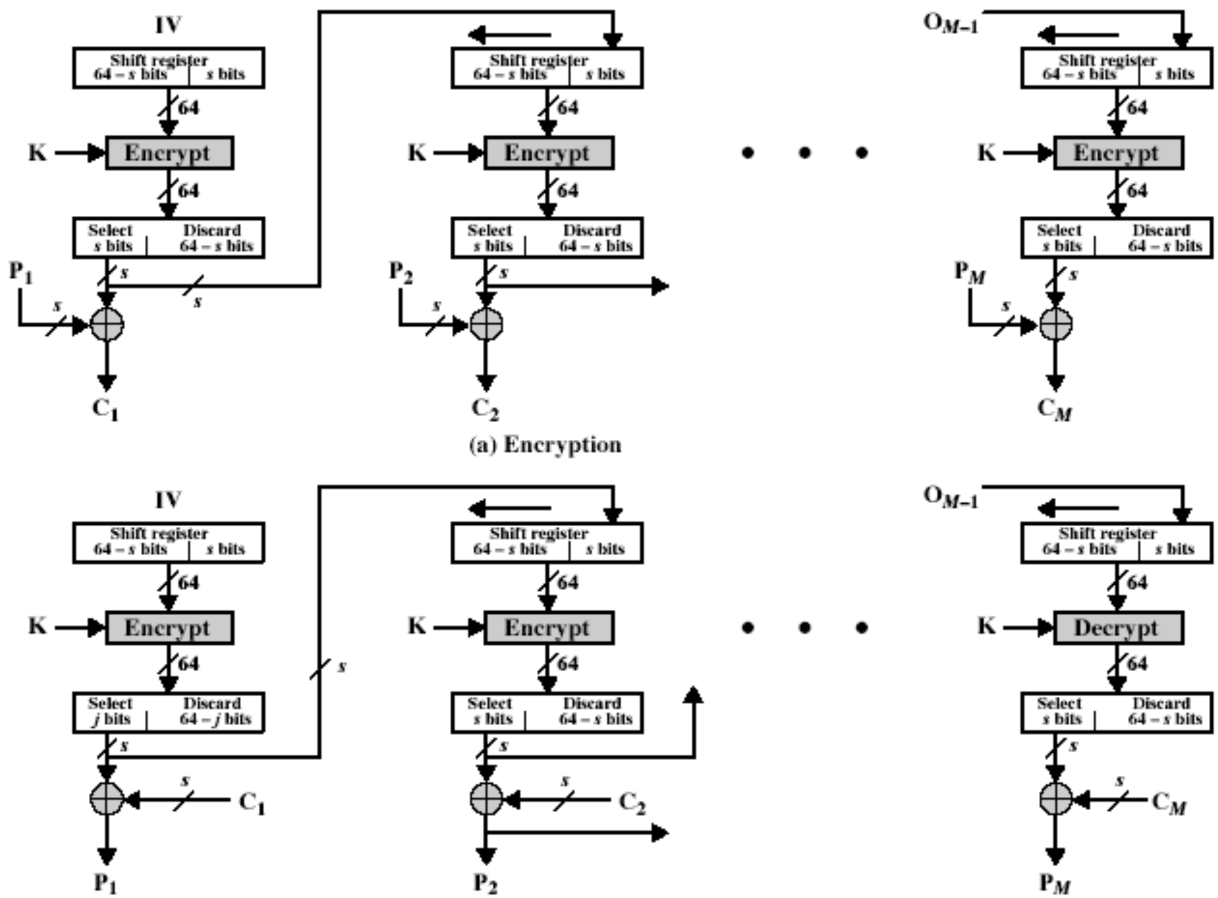
  $C_i = P_i \text{ XOR } O_i$

  $O_i = DES_{K1}(O_{i-1})$

  $O_{-1} = IV$

- uses: stream encryption over noisy channels

OFB Mode



(a) Encryption

(b) Decryption

Advantages and Limitations of OFB

- used when error feedback a problem or where need to encryptions before message is available
- superficially similar to CFB
- but feedback is from the output of cipher and is independent of message
- a variation of a Vernam cipher
  - hence must **never** reuse the same sequence (key+IV)
- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- originally specified with m-bit feedback in the standards
- subsequent research has shown that only **OFB-64** should ever be used

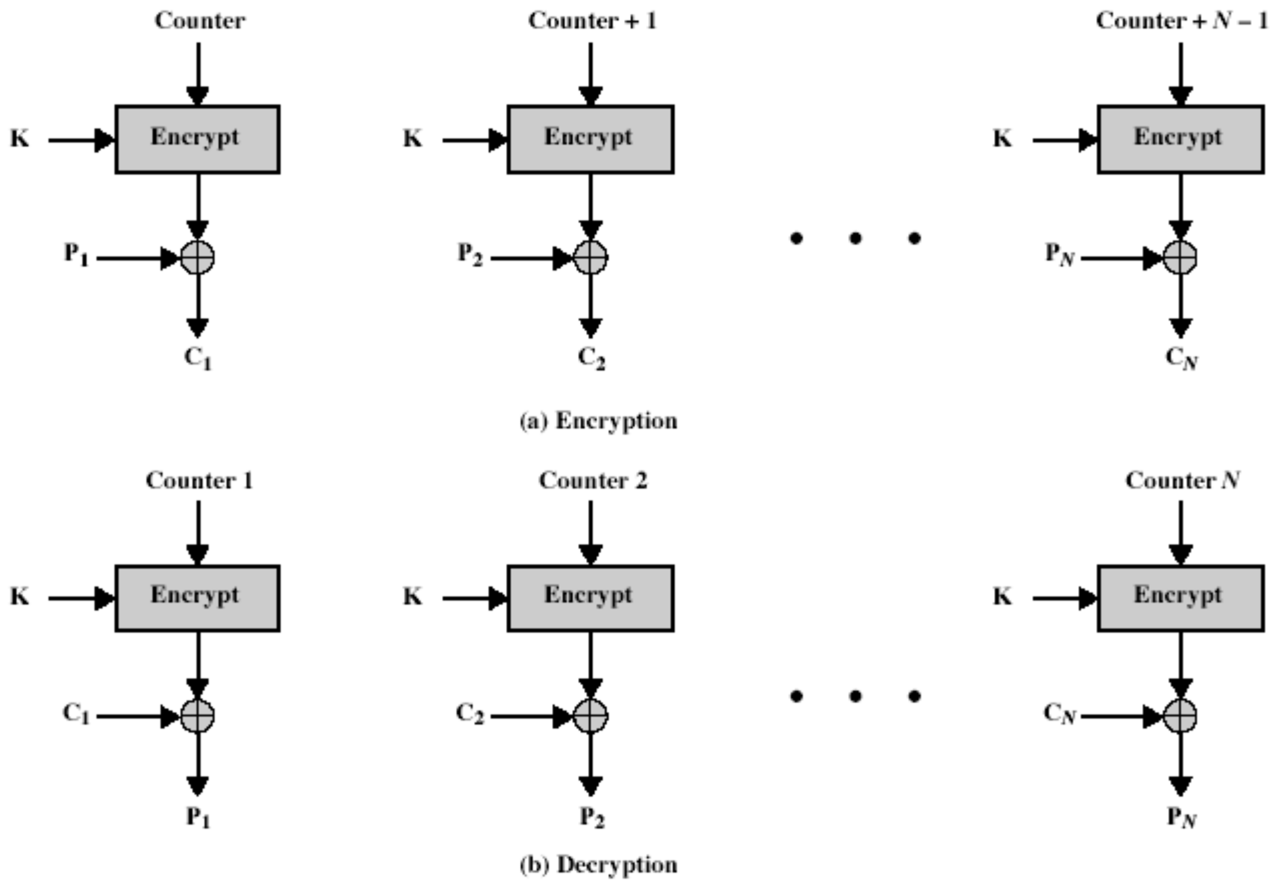
Counter (CTR)

- a "new" mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)

  $C_i = P_i \text{ XOR } O_i$

  $O_i = DES_{K1}(i)$
- uses: high-speed network encryptions

CTR Mode


Counter
Counter + 1
Counter + N – 1
K
Encrypt
P_1
P_2
P_N
C_1
C_2
C_N
(a) Encryption
Counter 1
Counter 2
Counter N
(b) Decryption

Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions
  - in advance of need
  - good for bursty high speed links
- random access to encrypted data blocks
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break (cf OFB)

XTS Mode

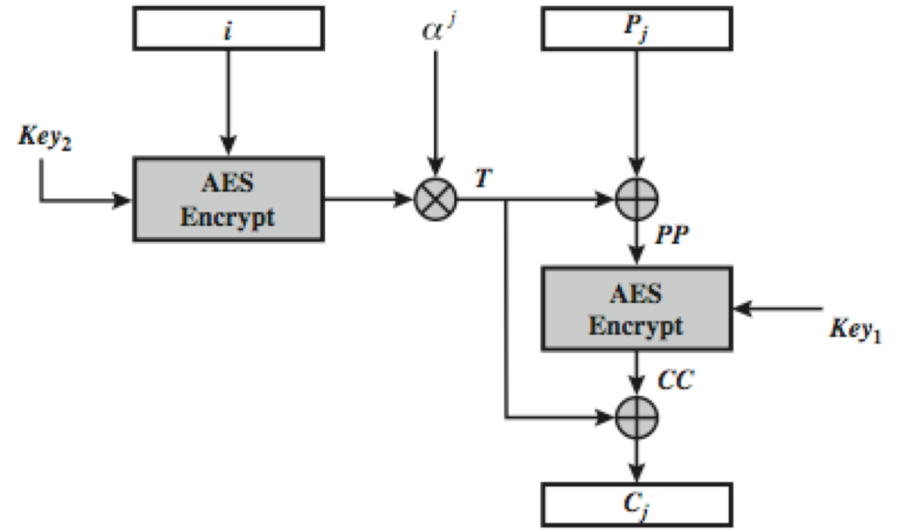
- new mode, for block oriented storage use
  - in IEEE Std 1619-2007
- concept of tweakable block cipher
- different requirements to transmitted data
- uses AES twice for each block

  $T_j = E_{K2}(i) \text{ XOR } \alpha^j$

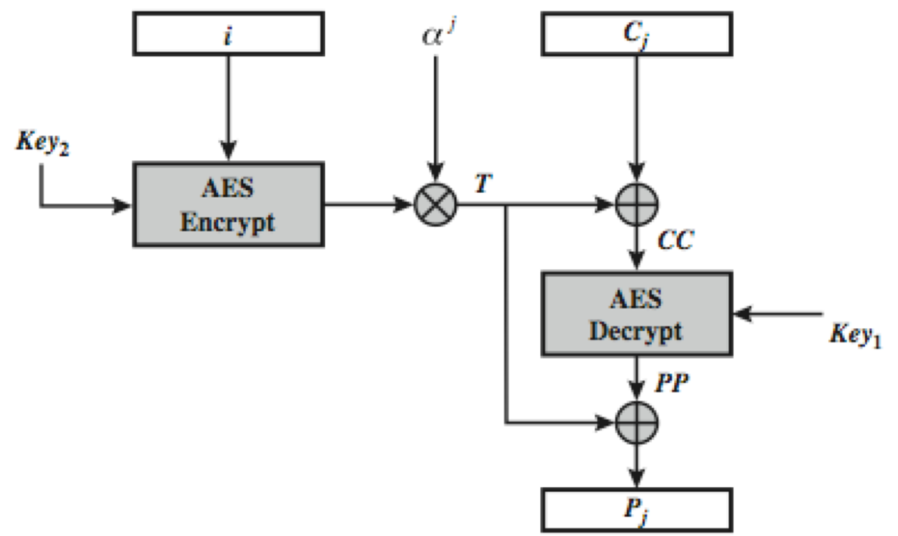
  $C_j = E_{K1}(P_j \text{ XOR } T_j) \text{ XOR } T_j$

  where i is tweak & j is sector no
- each sector may have multiple blocks

XTS Mode per block

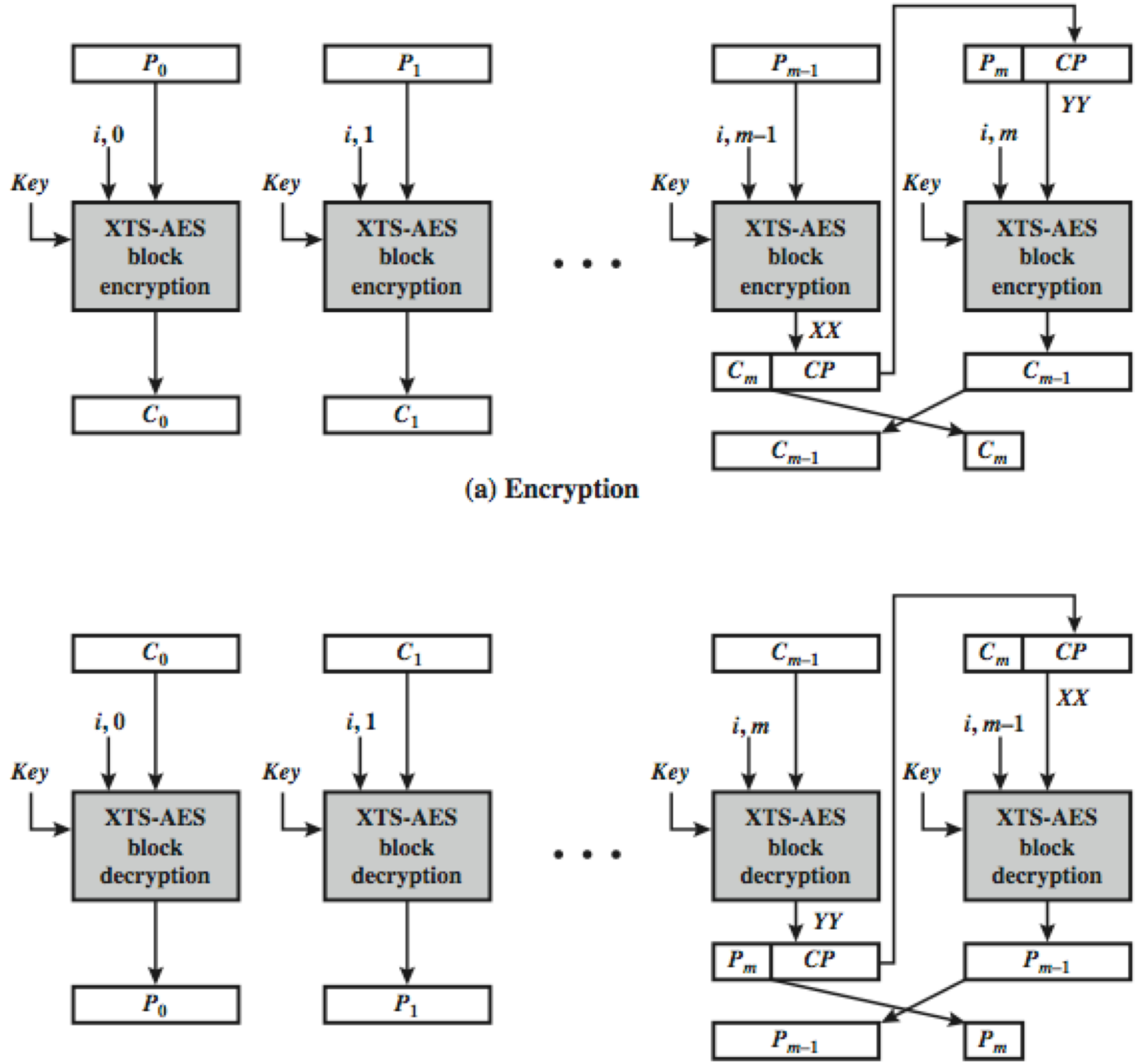


(a) Encryption



(b) Decryption

XTS Mode Overview



(a) Encryption

(b) Decryption

Advantages and Limitations of XTS

- efficiency
  - can do parallel encryptions in h/w or s/w
  - random access to encrypted data blocks
- has both nonce & counter
- addresses security concerned related to stored data

Cryptanalysis

Differential Cryptanalysis

- one of the most significant recent (public) advances in cryptanalysis
- known by NSA in 70's cf DES design
- Murphy, Biham & Shamir published in 90’s
- powerful method to analyse block ciphers
- used to analyse most current block ciphers with varying degrees of success
- DES reasonably resistant to it, cf Lucifer

Differential Cryptanalysis

- a statistical attack against Feistel ciphers
- uses cipher structure not previously used
- design of S-P networks has output of function *f* influenced by both input & key
- hence cannot trace values back through cipher without knowing value of the key
- differential cryptanalysis compares two related pairs of encryptions

Differential Cryptanalysis

- It's a **chosen plaintext attack**
- Initial plaintext: LH: m_0, RH: m_1
- At each round we produce a new half m_i
- After 16 rounds: $m_0, m_1, \ldots, m_{17}$

$$
\begin{aligned}
\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\
&= [m_{i-1} \oplus \mathrm{f}(m_i, K_i)] \oplus [m'_{i-1} \oplus \mathrm{f}(m'_i, K_i)] \\
&= \Delta m_{i-1} \oplus [\mathrm{f}(m_i, K_i) \oplus \mathrm{f}(m'_i, K_i)]
\end{aligned}
$$

Differential Cryptanalysis

- There are some **input difference** giving some **output difference** with high probability p
- if we know Δm_{i-1} and Δm_i with high probability then we can make a reasonable guess for Δm_{i+1}.
- From the equation we can infer subkey that was used in round i.
- then must iterate process over many rounds (with decreasing probabilities)

Differential Cryptanalysis

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR
- for large numbers of rounds, probability is so low that more pairs are required than brute-force
- Biham and Shamir have shown how a 13-round iterated characteristic can break the full 16-round DES
- DES with 15 rounds is easier to break than brute force

Linear Cryptanalysis

- another recent development
- also a statistical method
- must be iterated over rounds, with decreasing probabilities
- developed by Matsui et al in early 90's
- based on finding linear approximations
- can attack DES with 2^{43} known plaintexts, easier but still in practise infeasible

Summary

We have covered:

- Block cipher concepts
  - DES (details,strength)
- Stream cipher concepts – RC4
- Modes of Operation
  - ECB, CBC, CFB, OFB, CTR
- Differential & Linear Cryptanalysis