網路安全的理論與實務

楊中皇 著

第二章　私密金鑰密碼系統

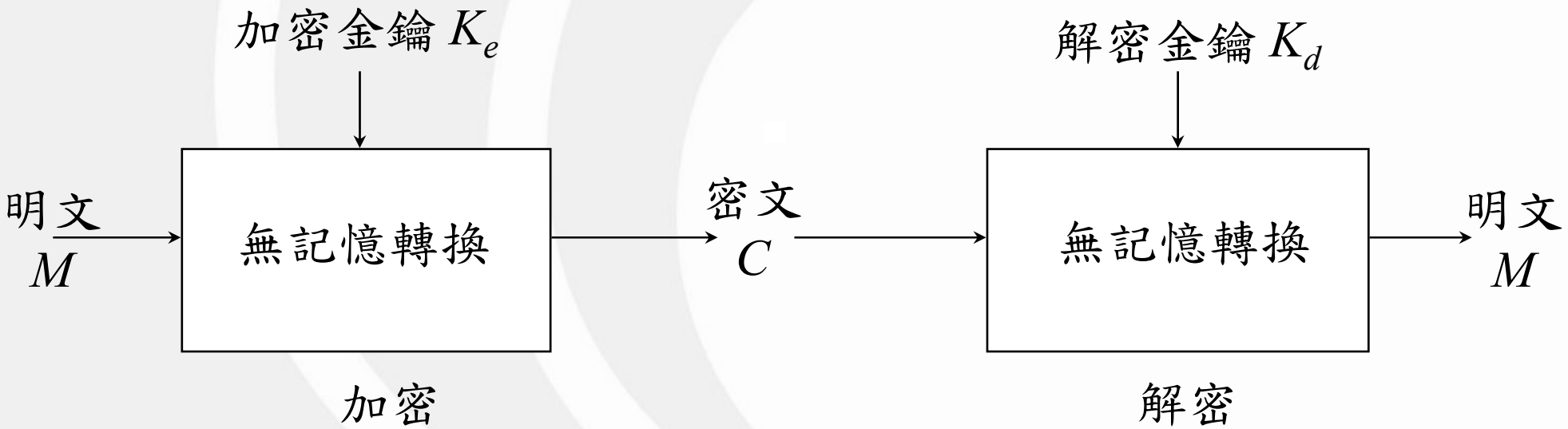
http://crypto.nknu.edu.tw/textbook/

伴 您 學 習 成 長 的 每 一 天

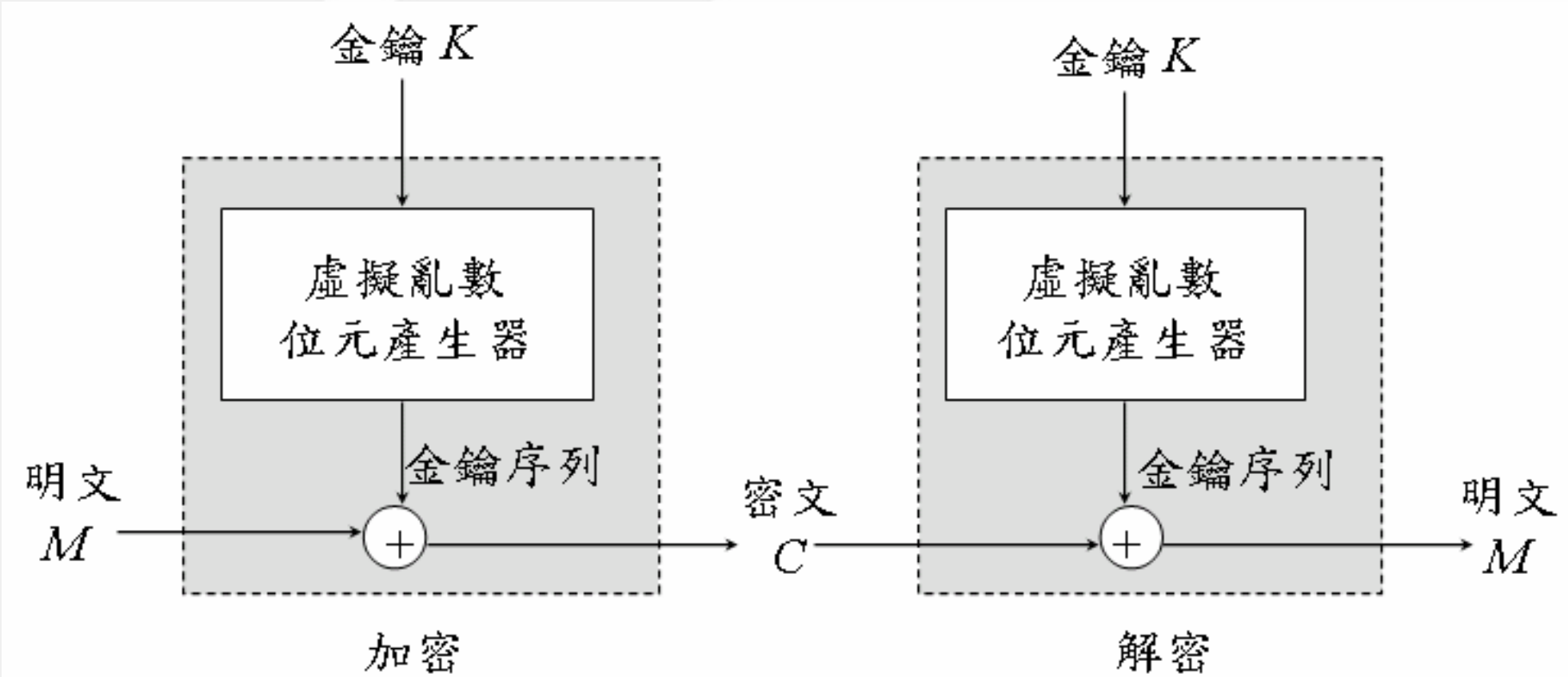
第二章 私密金鑰密碼系統

- 古典加密技術
- 資料加密標準(Data Encryption Standard，DES)
- 三重DES (Triple-DES)
- 高等加密標準(Advanced Encryption Standard，AES)
- 操作模式(Modes of Operation)
- 串流加密法(Stream Cipher)

區塊加密法(block cipher)



串流加密法(stream cipher)



古典加密技術

- 取代(substitution): 用另外的字元來取代原先的字元
- 置換(permutation)：改變原有明文字母排列的順序。

凱撒加密法(Caesar Cipher)

明文：TAKE THAT HILL (攻佔那山頭)
密文：WDNH WKDW KLOO

凱撒加密法

明文	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
密文	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

加密法則： $C = (M + 3) \mod 26$
解密法則： $M = (C - 3) \mod 26$

單字母加密法(Monoalphabetic Cipher)

- 每個字母以不重複的任意字母取代

明文	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
密文	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

- 可能金鑰數: $26\times25\times24\times\cdots\times1=26!$

- 無法用暴力攻擊法破解
- 可利用明文語言本身的特性
- 參見Claude E. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, vol.28-4, page 656-715, 1949.

英文字母出現的頻率(百分比)

字母	頻率	字母	頻率
A	8.2	N	6.7
B	1.5	O	7.5
C	2.8	P	1.9
D	4.3	Q	0.1
E	12.7	R	6.0
F	2.2	S	6.3
G	2.0	T	9.1
H	6.1	U	2.8
I	7.0	V	1.0
J	0.2	W	2.3
K	0.8	X	0.1
L	4.0	Y	2.0
M	2.4	Z	0.1

語言的冗餘性(redundancy)

- 雙字母出現的頻率依序約為 ”TH”、”ER”、”ON”、”AN”、”RE”、”HE”、”IN”、”ED”、”ND”、”HA”、”AT”、”EN”、”ES”、”OF”、”OR”、”NT”、”EA”、”TI”、”TO”、”TO” ……
- 三字母 ”THE” 、 ”AND” 、 ”THA”、 ”HER”、 ”ENT”、 ”ION” 、 ”TIO” 、 ”FOR”等是較常出現的。

希爾加密法(Hill Cipher)

- 一次將連續*d*個明文字母進行加密後得到*d*個密文字母。例如，一次加密兩個字母(*d*=2)，將明文*M* = (*m1*, *m2*)加密爲密文*C* = (*c1*, *c2*)的希爾加密公式

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \mod 26$$

- 希爾加密法雖然可抵擋僅有密文攻擊(ciphertext-only attack)，但無法抵擋已知明文攻擊(known-plaintext attack) 。若取得(m_{11}, m_{12})加密為密文(c_{11}, c_{12})與 (m_{21}, m_{22})加密為密文 (c_{21}, c_{22})，那麼

$$\begin{pmatrix} c_{11} & c_{21} \\ c_{12} & c_{22} \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} \begin{pmatrix} m_{11} & m_{21} \\ m_{12} & m_{22} \end{pmatrix} \mod 26$$

維吉尼爾加密法(Vegenere Cipher)

- 金鑰重複使用而延伸到和明文一樣的長度，然後將明文字母與對應位置的金鑰字母相加得到密文
- 使得一個明文字母可對應到多個密文字母。
- 例如，明文TAKE THAT HILL ，金鑰爲YANG

位置	1	2	3	4		5	6	7	8		9	10	11	12
明文	T	A	K	E		T	H	A	T		H	I	L	L
金鑰	Y	A	N	G		Y	A	N	G		Y	A	N	G
密文	R	A	X	K		R	H	N	Z		F	I	Y	R

金融提款卡密碼保護

- 維吉尼爾加密法保護個人識別碼(PIN, Personal Identification Number)
- 密文寫於卡片上，而非個人識別碼寫，不用擔心被別人撿到而輕易被冒用

 個人識別碼	9	4	1	0	3	1
金鑰	1	2	3	4	5	6
密文	0	6	4	4	8	7

置換加密法(Permutation cipher)

- 不會更改明文的字母，而是只移動他們的相對位置，金鑰即代表某種移動的方式。
- 例如將連續三個明文字母m_1, m_2, m_3改爲m_2, m_3, m_1；明文若爲TAKE THAT HILL，則密文爲AKTT HETH ALLI

	1	2	3	1		2	3	1	2		3	1	2	3
明文	T	A	K	E		T	H	A	T		H	I	L	L
密文	A	K	T	T		H	E	T	H		A	L	L	I

置換加密法(Permutation cipher)

- 亦可將明文寫成每四個字母爲一列，然後先選欄2的所有字母，再依序取欄4、欄3、欄1，最後輸出成密文；
- 例如明文爲TAKE THAT HILL，則密文爲AHIE TLKA LTTH

欄1	欄2	欄3	欄4
T	A	K	E
T	H	A	T
H	I	L	L

一次金鑰加密法(One-Time Pad)

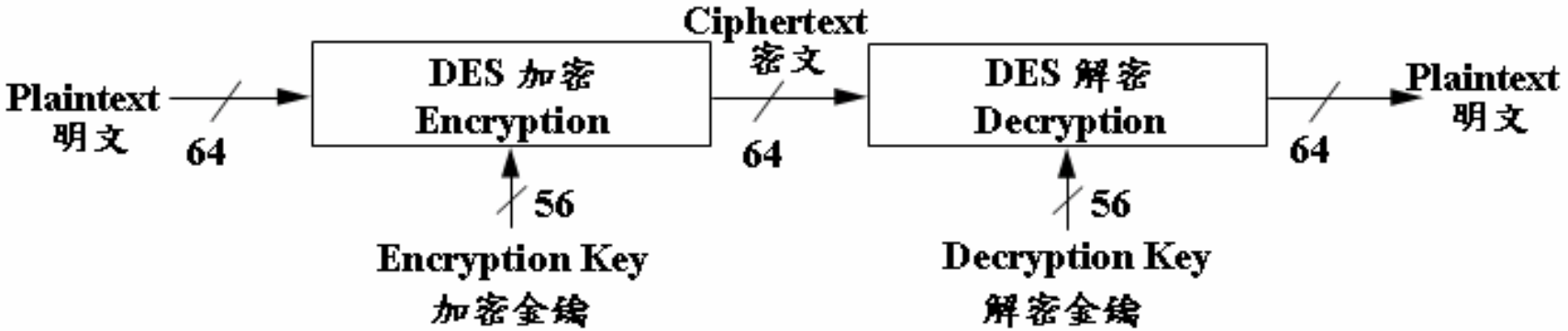
- 加(解)密金鑰只用一次，由物理亂數源產生
- 金鑰的長度等於欲傳送訊息的長度
- 加密的方法是將明文與金鑰位元串進行XOR (exclusive-OR)運算，而得到密文位元串
- 解密則是用相同私密金鑰與密文位元串做XOR運算以得到明文
- 絕對保密(perfect secrecy)

明文 M: 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 ...
金鑰 K: 1 0 0 1 1 0 0 1 0 0 0 1 0 1 1 ...

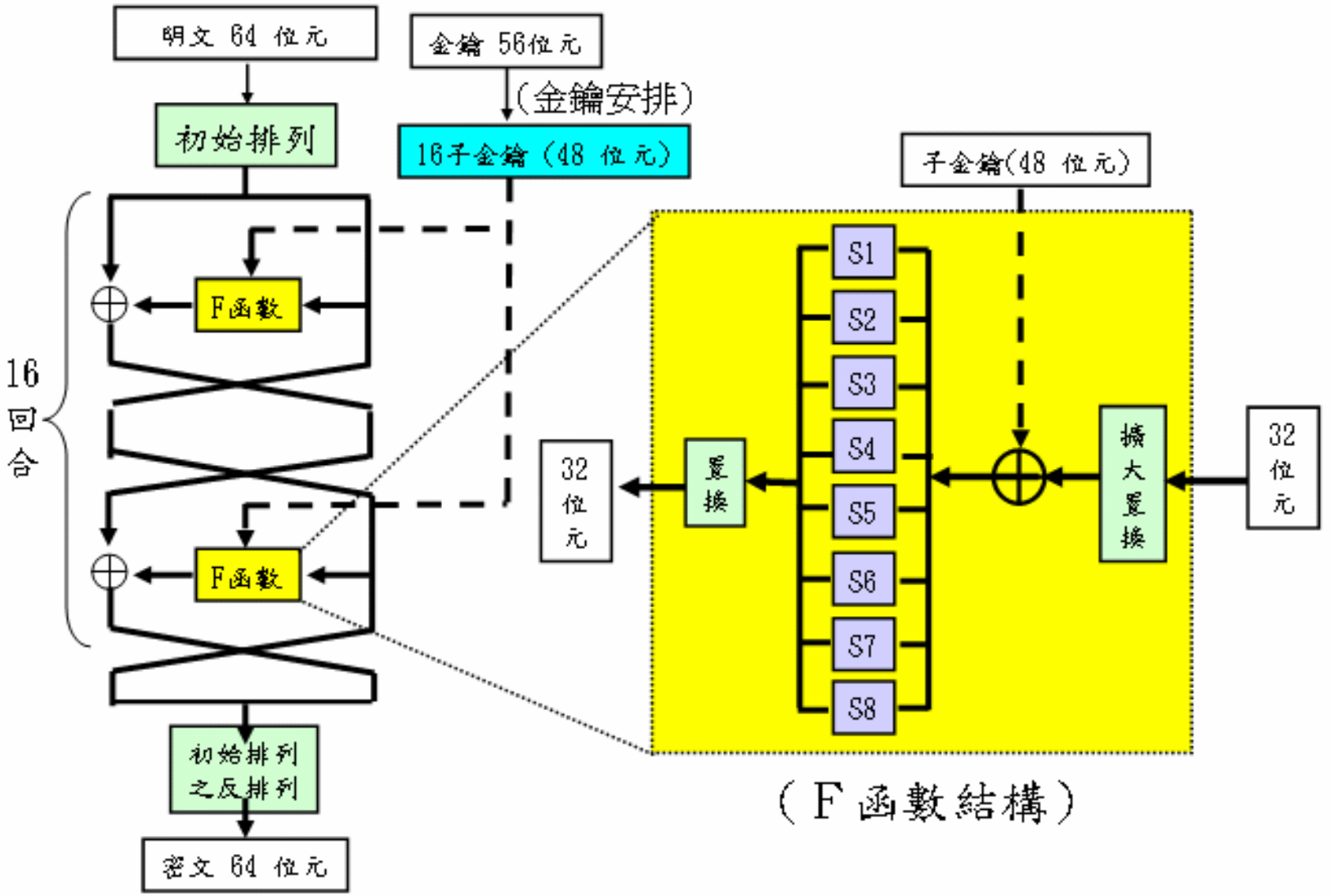
密文 C: 1 1 1 1 1 0 1 0 1 1 1 0 1 1 0 ...

DES (Data Encryption Standard)

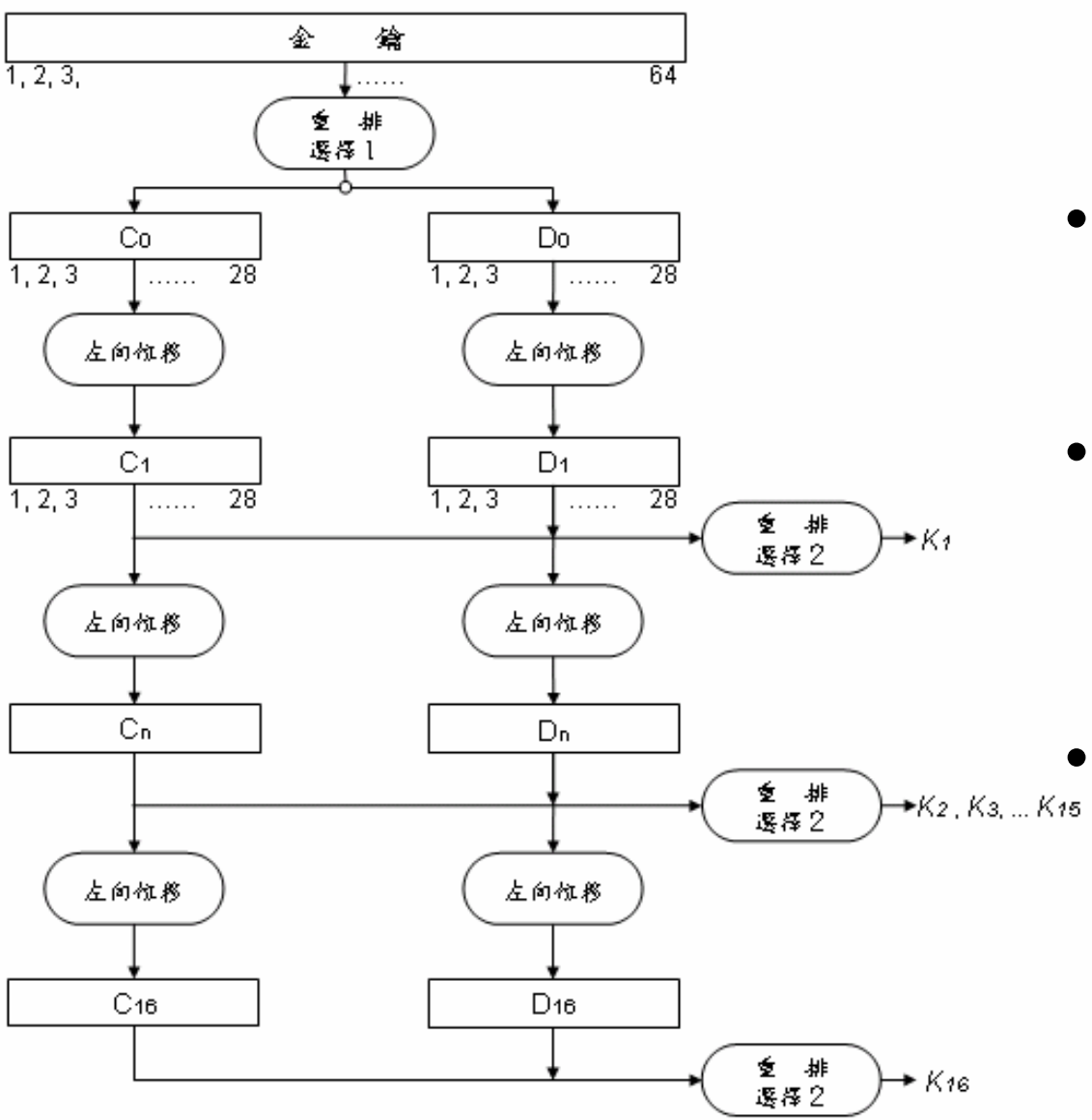
- 目前全世界使用最廣泛的密碼系統
- 明文64位元，密文64位元，金鑰56位元$2^{56} = 7\times10^{16}$
- ANSI X3.92 (1978年)，CNS X5011 "數據保密(加/解密)運算法“ (1983年)



DES加密


明文 64 位元
金鑰 56位元
(金鑰安排)
初始排列
16子金鑰 (48 位元)
子金鑰(48 位元)
16 回合
F函數
F函數
S1
S2
S3
S4
S5
S6
S7
S8
32 位元
置換
擴大置換
32 位元
初始排列之反排列
密文 64 位元
(F函數結構)

DES金鑰安排(key schedule)



- DES輸入的金鑰有56個位元，將用來產生額外的16組子金鑰(subkey)
- 金鑰原有56個位元，將它看成64個位元，並將其編號爲1到64
- 先經過重排選擇1依序去挑第57位、第49位、第41位、第33位...等，其中8的倍數的位元均不挑
- 金鑰輸入視爲64個位元，但是扣除位置爲8的倍數的位元，所以經過重排選擇1只有56個位元

DES金鑰安排 - 重排選擇1

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
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

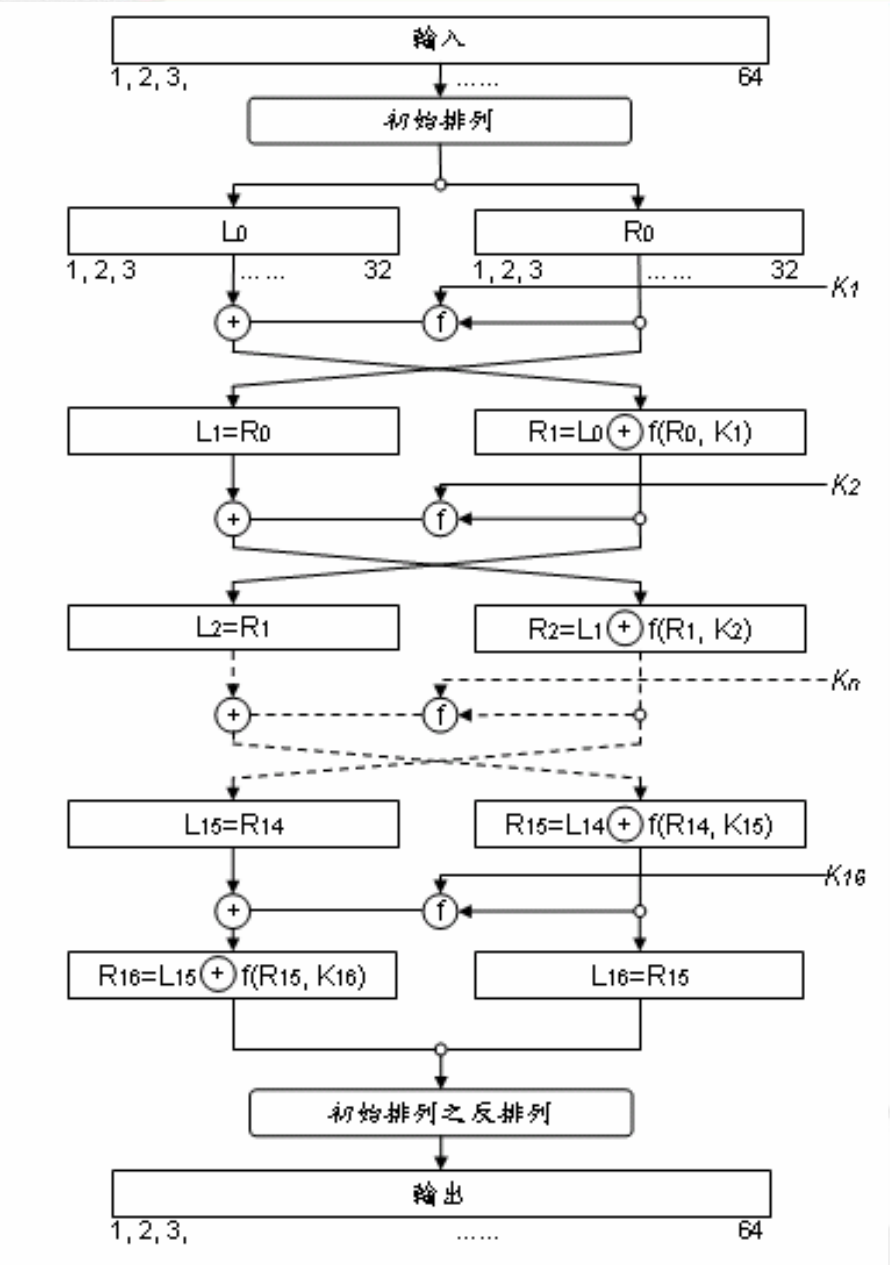
DES金鑰安排 - 向左位移表

回合數	左位移數
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金鑰安排-重排選擇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

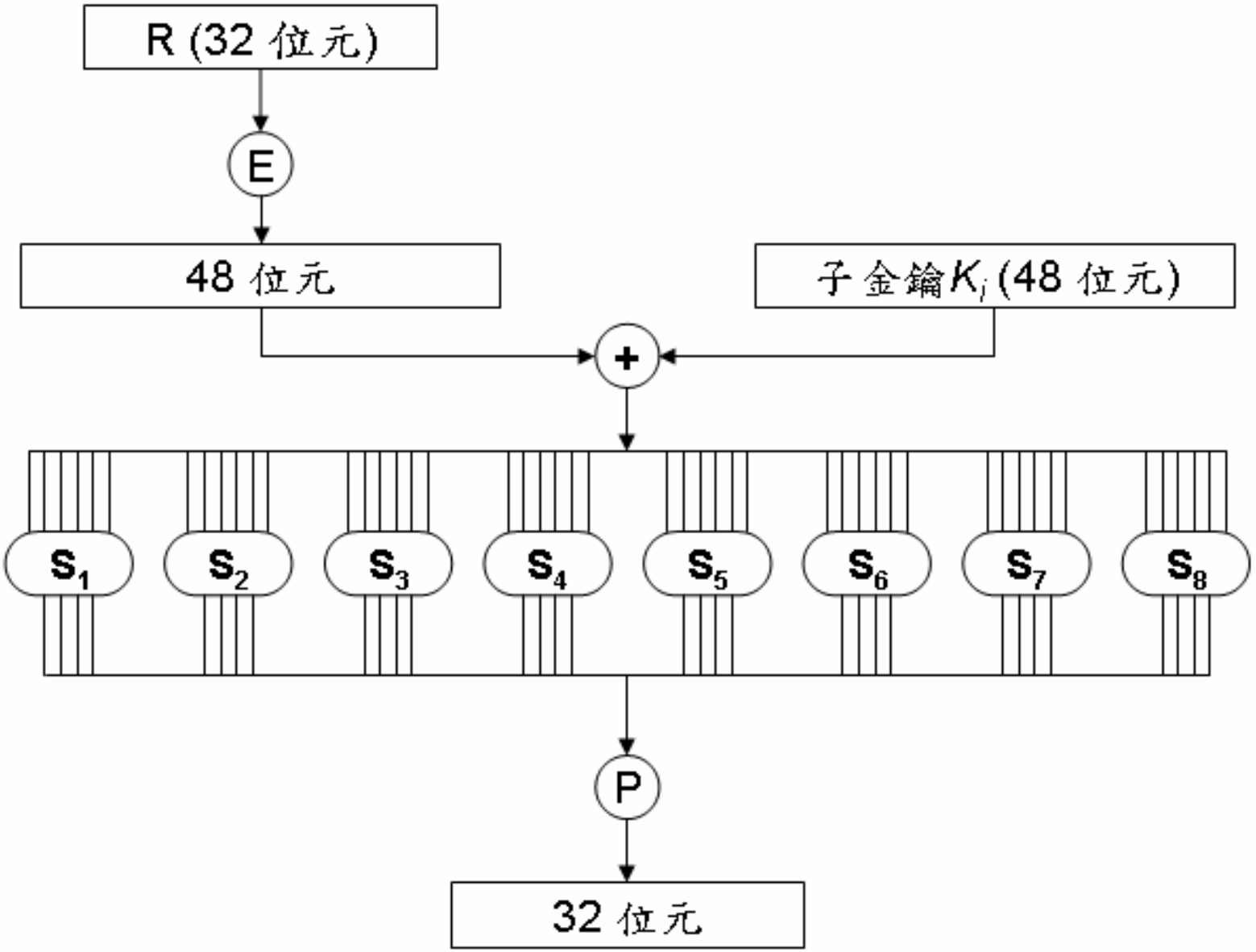
DES加密計算流程圖


輸入
1, 2, 3, ... 64
初始排列
L0
R0
1, 2, 3 32
1, 2, 3 32
K1
f
L1=R0
R1=L0 ⊕ f(R0, K1)
K2
L2=R1
R2=L1 ⊕ f(R1, K2)
Kn
L15=R14
R15=L14 ⊕ f(R14, K15)
K16
R16=L15 ⊕ f(R15, K16)
L16=R15
初始排列之反排列
輸出
1, 2, 3, ... 64

DES - 初始排列(IP)

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

DES - F函數



DES - 擴充函數(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

DES -選擇函數S_1

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

例如：S_1選擇函數的輸入爲011011 ，輸出爲0101 (5)

DES - F函數排列函數(P)

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 -初始排列之反排列(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

DES金鑰安排(子金鑰$K_1, K_2, \ldots, K_{16}$)範例

輸入金鑰	71399AED779384DA (十六進制) 0111000100111001100110101110110101110111100100111000010011011010
C0, D0	EC991BB B4588E7 11101100100110010001101110111 1011010001011000100011100111
C1, D1	D932377 68B11CF 11011001001100100011011101111 0110100010110001000111001111
子金鑰 K_1	3D8FCD373F48 001111011000111111001101001101110011111101001000
C2, D2	B2646EF D16239E 101100100110010001101110111 1101000101100010001110011110
子金鑰 K_2	AB3D984E1647 101010110011110110011000010011100001011001000111

DES加密範例

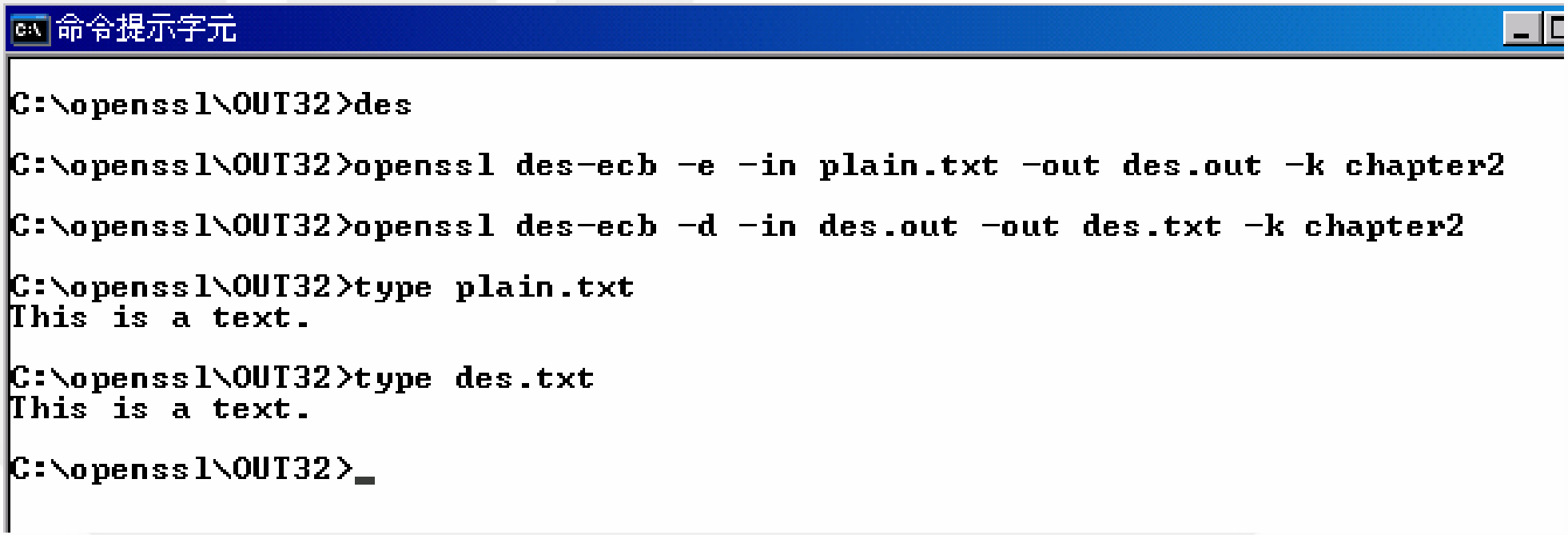
輸入 明文	7465874626F6F6B 0111010001100101011110000111010001100010011011110110111101101011
L0, R0	FF0D6BE2 00FFE4F0 11111111000011010110101111100010 00000000111111111110010011110000
L1, R1	00FFE4F0 21084679 00000000111111111110010011110000 00100001000010000100011001111001
L2, R2	21084679 7B4689FC 00100001000010000100011001111001 01111011010001101000100111111100
L3, R3	7B4689FC 1042EFA3 01111011010001101000100111111100 00010000010000101110111110100011

差異破解法(Differential Cryptanalysis)

- 以色列研究人員Biham和Shamir於1990年提出
- Eli Biham, Adi Shamir, **Differential cryptanalysis of DES-like cryptosystems**, Technical report CS90-16, Weizmann Institute of Science
- 提出一套如果DES只做8個回合或12個回合時的破解方式

OpenSSL進行DES加解密

- 加密 openssl des-ecb -e -in plain.txt -out des.out -k chapter2
- 解密 openssl des-ecb -d -in des.out -out des.txt -k chapter2



OpenSSL測試DES加解密效率

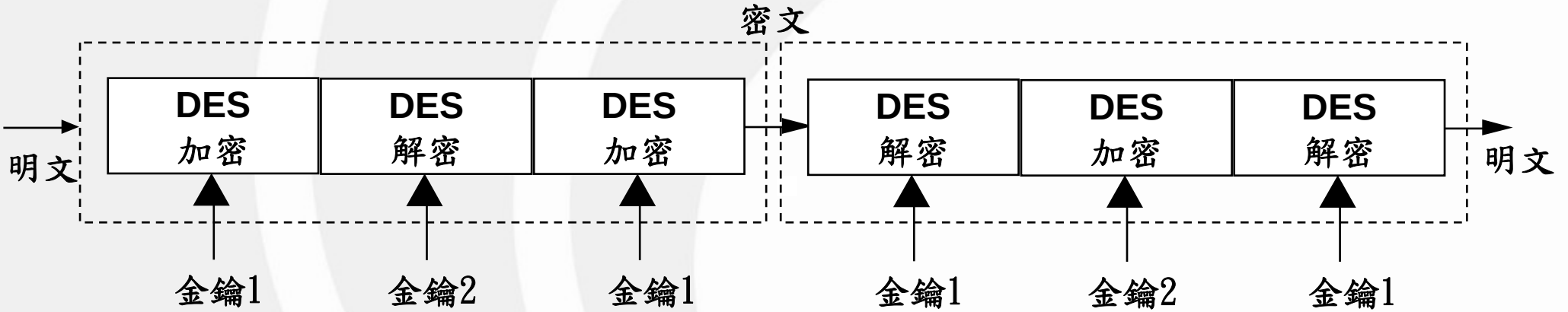
- openssl speed des

```
C:\openssl\OUT32>desspeed

C:\openssl\OUT32>openssl speed des
To get the most accurate results, try to run this
program when this computer is idle.
First we calculate the approximate speed ...
Doing des cbc 10485760 times on 16 size blocks: 10485760 des cbc's in 11.35s
Doing des cbc 2621440 times on 64 size blocks: 2621440 des cbc's in 11.04s
Doing des cbc 655360 times on 256 size blocks: 655360 des cbc's in 11.02s
Doing des cbc 163840 times on 1024 size blocks: 163840 des cbc's in 11.10s
Doing des cbc 20480 times on 8192 size blocks: 20480 des cbc's in 11.49s
Doing des ede3 3495253 times on 16 size blocks: 3495253 des ede3's in 9.50s
Doing des ede3 873813 times on 64 size blocks: 873813 des ede3's in 9.28s
Doing des ede3 218453 times on 256 size blocks: 218453 des ede3's in 9.02s
Doing des ede3 54613 times on 1024 size blocks: 54613 des ede3's in 9.51s
Doing des ede3 6826 times on 8192 size blocks: 6826 des ede3's in 9.19s
OpenSSL 0.9.8 05 Jul 2005
built on: Mon Sep 19 08:36:38 2005
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,cisc,4,long) aes(partial) idea(int) blow
compiler: bcc32 -DWIN32_LEAN_AND_MEAN -q -w-aus -w-par -w-inl  -c -tWC -tWM -DOPENSSL_SY
32 -D_stricmp=stricmp -O2 -ff -fp -DOPENSSL_NO_RC5 -DOPENSSL_NO_MDC2 -DOPENSSL_NO_KRB5
available timing options: TIMEB HZ=1000
timing function used: ftime
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
des cbc          14786.90k    15202.26k    15229.86k    15120.06k    14606.67k
des ede3          5884.26k     6024.35k     6197.94k     5878.04k     6082.74k
```

三重DES (Triple-DES)

- 提昇DES加解密演算法的安全性



- 金鑰爲56+56=112個位元
- 金鑰1等於金鑰2，則三重DES就等於原先的DES
- 安全性2^{112}

OpenSSL進行三重DES加解密

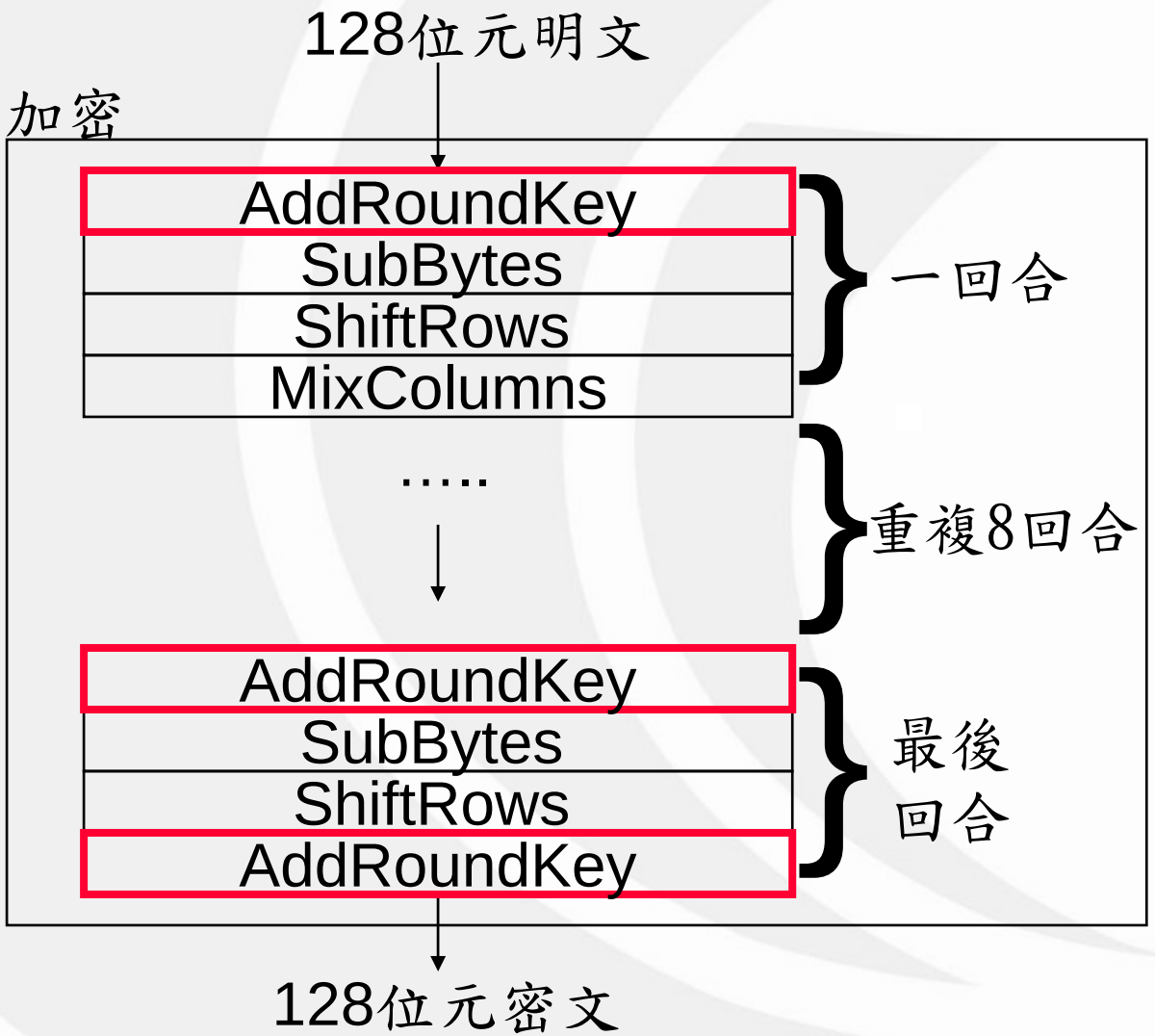
- 加密 openssl des-ede3 -e -in plain.txt -out des3.out -k chapter2
- 解密openssl des-ede3 -d -in des3.out -out des3.txt -k chapter2
- 測試效率 openssl speed des

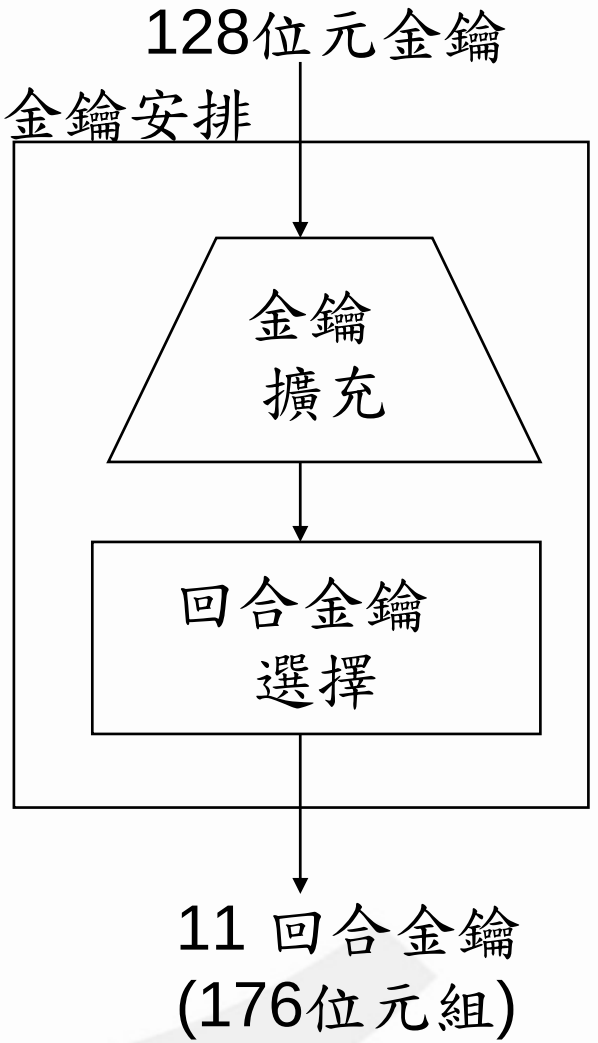
```
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
des cbc          14786.90k    15202.26k    15229.86k    15120.06k    14606.67k
des ede3          5884.26k     6024.35k     6197.94k     5878.04k     6082.74k
```

高等加密標準(Advanced Encryption Standard，AES)

- 美國國家標準科技局(National Institute of Standard and Technology，NIST) 2001年公佈
- 全部設計原理都是公開
- 明文與密文都是128位元(16位元組) ，金鑰長度可128、192及256位元
- AES-128(金鑰長度爲128位元的AES)軟韌體實現速度遠比DES快，同時由於AES-128有3.4×10^{38}種可能的金鑰選擇，遠較56位元DES的金鑰選擇爲大，且截至目前爲止並無有效的攻擊方式；AES-128除了速度較快外也遠較DES/三重DES安全

128位元金鑰之AES加密





AES-128加解密

- AES-128金鑰安排把輸入的128位元金鑰擴展成11組各128位元的回合金鑰(round key)
- AES加解密時有四種基本運算(稱作層，layer)分別是AddRoundKey、SubBytes、ShiftRows、及MixColumns等四層
- 每次進行AddRoundKey都需用到回合金鑰，其餘運算層與回合金鑰無關。
- 明文先用回合金鑰依序進行AddRoundKey、SubBytes、ShiftRows，然後再做MixColumns，這四個運算稱爲一個回合
- 總共需重複做九個回合
- 最後再進行 AddRoundKey、SubBytes、ShiftRows、及AddRoundKey即得到密文

AES-128加密過程之虛擬程式碼

```
AES-128 (byte in[16], byte out[16], word w[44])
begin
    byte    state[4,4]
    state = in

    for round = 0 step 1 to 8
        AddRoundKey (state, w[round*4, round*4 +3])
        SubBytes (state)
        ShiftRows (state)
        MixColumns (state)
    end for

    AddRoundKey (state, w[36, 39])
    SubBytes (state)
    ShiftRows (state)
    AddRoundKey (state, w[40, 43]

    out = state
end
```

- **in**代表輸入的明文長度爲**16**個位元組
- **out**代表輸出的密文長度也是**16**個位元組

- **state (狀態)** 變數是四乘四矩陣，共爲**16**個位元組
- **w**代表**44**個字組**(word)**的回合金鑰，每個字組包含**4**個位元組，而**176**個位元組的回合金鑰**(round key)**以**w[0]**至**w[43]**共**44**個字組代表

state 16位元組以行優先

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

(a) state 的位元組採行優先

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

(b) state 的矩陣內容

AES運算層-1: AddRoundKey

- AddRoundKey (state, w[round*4, round*4 +3]) 是將16位元組的state與16位元組(4字組)的回合金鑰中的每個對應位元進行XOR運算，並將運算結果重新放於state
- 回合金鑰僅用於這個運算，與其他三個運算無關，且僅需做XOR位元組運算，屬於快速計算

AES運算層-2: SubBytes

- SubBytes (state) 會將state(16位元組)的每一個位元組進行取代(substitution)，運算結果重新放於state
- 取代的方式見表，每個位元組用兩個十六進制數字xy表示，依據x與y值查表可得取代新值。例如xy=00取代爲63，亦即S-Box({00})=63，另外，S-Box({f0})= 8c，S-Box({ff})=16。

AES 位元組取代(S-Box)

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

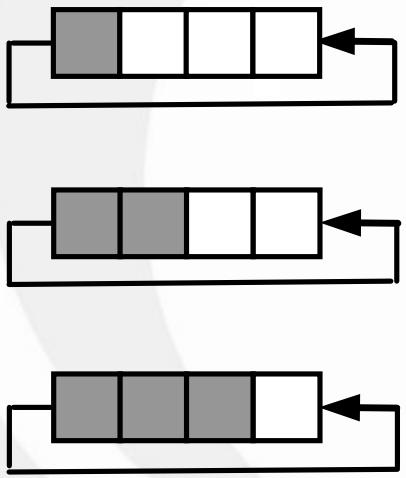
例如 **S-Box({00})=63，S-Box({f0})= 8c，S-Box({ff})=16**

AES運算層-3: ShiftRows

- ShiftRows (state)，是將state(四乘四矩陣)的第一列不予以更動，第二列向左移一個位元組的位置，而第三列向左移兩個位元組，最後一列則向左移三個位元組，此運算結果重新放於state，
- 第二列4位元組原先爲($s_{1,0}$, $s_{1,1}$, $s_{1,2}$, $s_{1,3}$)經過移位後成爲($s_{1,1}$, $s_{1,2}$, $s_{1,3}$, $s_{1,0}$)
- 第三列原先爲($s_{2,0}$, $s_{2,1}$, $s_{2,2}$, $s_{2,3}$)經過移位後成爲($s_{2,2}$, $s_{2,3}$, $s_{2,0}$, $s_{2,1}$)
- 第四列原先爲($s_{3,0}$, $s_{3,1}$, $s_{3,2}$, $s_{3,3}$)經過移位後成爲($s_{3,3}$, $s_{3,0}$, $s_{3,1}$, $s_{3,2}$)

AES之ShiftRows

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

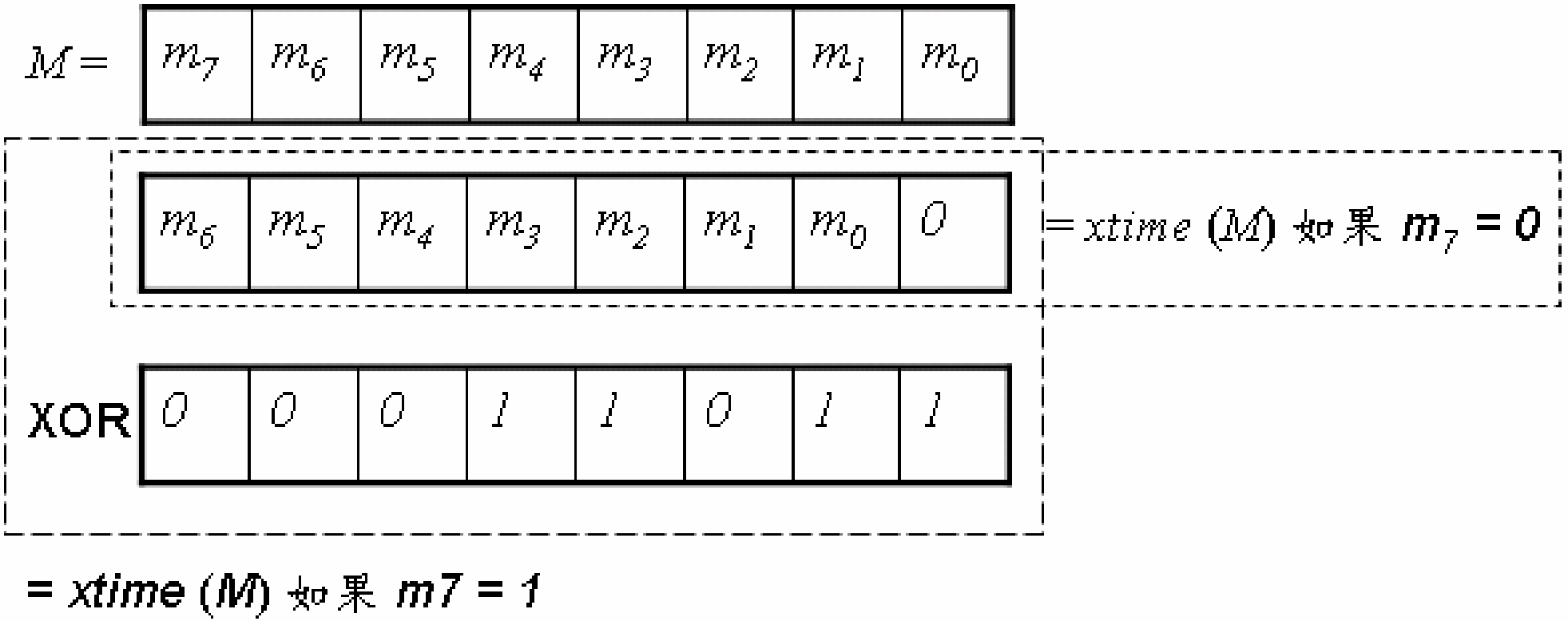


$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

AES運算層-4: MixColumns

- MixColumns (state)則將每一行四位元組經過特殊矩陣運算，得到新的四位元組
- *xtime*函數: 此函數輸入與輸出皆爲八位元(一位元組)，計算方法是將輸入的八位元值向左移動一位元，最右邊無效位元(least significant bit, lsb)的位置補0，如果原輸入值的最左邊有效位元(most significant bit, msb)爲1，則將左移結果與十六進制{1b}進行XOR運算再輸出。假設*xtime*函數八位元輸入$M = (m_7, m_6, m_5, m_4, m_3, m_2, m_1, m_0)$，則左移一位元將成爲$(m_6, m_5, m_4, m_3, m_2, m_1, m_0, 0)$；如果$m_7 = 0$，則$xtime\ (M) = (m_6, m_5, m_4, m_3, m_2, m_1, m_0, 0)$ ；如果$m_7 = 1$，則$xtime\ (M) = (m_6, m_5, m_4, m_3, m_2, m_1, m_0, 0)$ XOR $(0, 0, 0, 1, 1, 0, 1, 1)$。
- xtime({57}) = {ae}, xtime({ae})={47}, xtime({d4}) = {b3}

AES *xtime*函數


M =
m7 m6 m5 m4 m3 m2 m1 m0
m6 m5 m4 m3 m2 m1 m0 0
= xtime (M) 如果 m7 = 0
XOR
0 0 0 1 1 0 1 1
= xtime (M) 如果 m7 = 1

AES之MixColumns

$$\begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \qquad 0 \le c < 4$$

⊕代表 XOR

$$s_{0,0}\,(\text{新}) = (\{02\} \bullet s_{0,0}) \oplus (\{03\} \bullet s_{1,0}) \oplus s_{2,0} \oplus s_{3,0}$$
$$s_{1,0}\,(\text{新}) = (\{02\} \bullet s_{1,0}) \oplus (\{03\} \bullet s_{2,0}) \oplus s_{3,0} \oplus s_{0,0}$$
$$s_{2,0}\,(\text{新}) = (\{02\} \bullet s_{2,0}) \oplus (\{03\} \bullet s_{3,0}) \oplus s_{0,0} \oplus s_{1,0}$$
$$s_{3,0}\,(\text{新}) = (\{02\} \bullet s_{3,0}) \oplus (\{03\} \bullet s_{0,0}) \oplus s_{1,0} \oplus s_{2,0}$$

$$\{02\} \bullet s_{i,j} = s_{i,j} \bullet \{02\} = xtime(s_{i,j})$$
$$\{03\} \bullet s_{i,j} = s_{i,j} \bullet \{03\} = s_{i,j} \bullet (\{01\} \oplus \{02\}) = s_{i,j} \oplus xtime(s_{i,j})$$

MixColumns範例

如果$(s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0})$ = ({d4}, {bf}, {5d}, {30})，

那麼 $s_{0,0}$ (新) = ({02}•{d4}) ⊕ ({03}•{bf}) ⊕ {5d} ⊕ {30} = {b3}⊕{da}⊕{5d}⊕{30} = {04}，

經過 MixColumns 的運算，四位元組$(s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0})$的新值為({04}, {66}, {81}, {e5})

AES-128金鑰擴充之虛擬程式碼

```
KeyExpansion (byte key[16], word w[44])
begin
i=0
    while (i < 4)
    w[i] = word[key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]]
    i = i + 1
    end while
    i = 4
    while (i < 44)
    word temp = w[i - 1]
    if (i mod 4 = 0)
    temp = SubWord(RotWord(temp)) XOR Rcon[i / 4]
    end if
    w[i] = w[i - 4] XOR temp
i = i + 1
end while
end
```

- **16位元組的AES-128金鑰(key [16])擴展成為44字組的回合金鑰(w [44])**
- **先將16位元組的輸入金鑰放至前4字組即w[0]至w[3]，而後依序算出w[4]、w[5]、…、 w[43]，**
- **其中當w陣列指標為4的倍數(w[4]、w[8]、w[12]、…、w[40])時，須作額外處理**

AES SubWord(RotWord(temp))

- RotWord是將一個字組內的四個位元組向前移動一位元組，例如原先字組爲 [b_0, b_1, b_2, b_3] 那麼經RotWord移動後即成爲 [b_1, b_2, b_3, b_0]
- SubWord則是將四個位元組各自進行S-Box，例如原先字組爲 [b_0, b_1, b_2, b_3] 那麼經SubWord取代後成爲 [S-Box(b_0), S-Box(b_1), S-Box(b_2), S-Box(b_3)]
- Rcon則爲字組常數，第一位元組非零，Rcon [1] ={01},{00},{00},{00})，…，Rcon [10] = ({36},{00},{00},{00})

i	1	2	3	4	5	6	7	8	9	10
Rcon [*i*] 第一個位元組(十六進制)	01	02	04	08	10	20	40	80	1b	36

AES-128第2組回合金鑰的計算方式

Key [16] = Key [0] ⊕ S-Box(Key [13]) ⊕ {01}
Key [17] = Key [1] ⊕ S-Box(Key [14])
Key [18] = Key [2] ⊕ S-Box(Key [15])
Key [19] = Key [3] ⊕ S-Box(Key [12])
Key [20] = Key [4] ⊕ Key [16]
Key [21] = Key [5] ⊕ Key [17]
Key [22] = Key [6] ⊕ Key [18]
Key [23] = Key [7] ⊕ Key [19]
Key [24] = Key [8] ⊕ Key [20]
Key [25] = Key [9] ⊕ Key [21]
Key [26] = Key [10] ⊕ Key [22]
Key [27] = Key [11] ⊕ Key [23]
Key [28] = Key [12] ⊕ Key [24]
Key [29] = Key [13] ⊕ Key [25]
Key [30] = Key [14] ⊕ Key [26]
Key [31] = Key [15] ⊕ Key [27]

AES-128加密範例

- 明文爲(十六進制) 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34，16位元組金鑰爲2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c，而計算得到的16位元組密文爲39 25 84 1d 02 dc 09 fb dc 11 85 97 19 6a 0b 32。

回合數	回合開始 `state` 值	經過 `SubBytes`	經過 `ShiftRows`	經過 `MixColumns`		回合金鑰值	
	32 88 31 e0					2b 28 ab 09	
輸入	43 5a 31 37				⊕	7e ae f7 cf	=
明文	f6 30 98 07					15 d2 15 4f	
	a8 8d a2 34					16 a6 88 3c	

	39 02 dc 19
輸出	25 dc 11 6a
密文	84 09 85 0b
	1d fb 97 32

AES解密

- AES的解密需將四個運算層進行逆計算，AddRoundKey的逆計算仍爲AddRoundKey，而ShiftRows的逆計算則爲向右移動零到三位元組，且SubBytes的逆計算爲另一反取代，在MixColumns的逆計算係數由{02}{03}{01}{01}改爲{0e}{0b}{0d}{09}，其中除MixColumns的逆計算係數更改後計算變慢外，其餘三個運算層則速度不變
- 由於AES解密比加密速度慢，所以通常在進行檔案加解密時，會利用下一節即將介紹的操作模式來避開進行AES解密動作

OpenSSL進行AES加解密

- 假設明文存放於檔案”plain.txt”，並希望能將加密過後的密文檔存放在”aes.out”，加密金鑰則用通行碼”chapter2”來保護，那麼可用下列指令將明文檔加密
  - openssl aes-128-ecb -e -in plain.txt -out aes.out -k chapter2
- 而當要解密的密文檔爲”aes.out”，並希望解密出來的明文能存放於檔案” aes.txt”，且利用通行碼”chapter2”來解密時，則可利用下列指令解密
  - openssl aes-128-ecb -d -in aes.out -out aes.txt -k chapter2

OpenSSL測試AES加解密效率

- openssl speed aes

命令提示字元

```
C:\openssl\OUT32>openssl speed aes
To get the most accurate results, try to run this
program when this computer is idle.
First we calculate the approximate speed ...
Doing aes-128 cbc 10485760 times on 16 size blocks: 10485760 aes-128 cbc's in 5.52s
Doing aes-128 cbc 2621440 times on 64 size blocks: 2621440 aes-128 cbc's in 4.96s
Doing aes-128 cbc 655360 times on 256 size blocks: 655360 aes-128 cbc's in 5.17s
Doing aes-128 cbc 163840 times on 1024 size blocks: 163840 aes-128 cbc's in 5.20s
Doing aes-128 cbc 20480 times on 8192 size blocks: 20480 aes-128 cbc's in 5.07s
Doing aes-192 cbc 10485760 times on 16 size blocks: 10485760 aes-192 cbc's in 6.00s
Doing aes-192 cbc 2621440 times on 64 size blocks: 2621440 aes-192 cbc's in 5.54s
Doing aes-192 cbc 655360 times on 256 size blocks: 655360 aes-192 cbc's in 5.60s
Doing aes-192 cbc 163840 times on 1024 size blocks: 163840 aes-192 cbc's in 5.96s
Doing aes-192 cbc 20480 times on 8192 size blocks: 20480 aes-192 cbc's in 5.63s
Doing aes-256 cbc 10485760 times on 16 size blocks: 10485760 aes-256 cbc's in 6.76s
Doing aes-256 cbc 2621440 times on 64 size blocks: 2621440 aes-256 cbc's in 6.13s
Doing aes-256 cbc 655360 times on 256 size blocks: 655360 aes-256 cbc's in 6.40s
Doing aes-256 cbc 163840 times on 1024 size blocks: 163840 aes-256 cbc's in 6.61s
Doing aes-256 cbc 20480 times on 8192 size blocks: 20480 aes-256 cbc's in 6.09s
OpenSSL 0.9.8 05 Jul 2005
built on: Mon Sep 19 08:36:38 2005
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,cisc,4,long) aes(partial) idea(int) blowfish(idx)
compiler: bcc32 -DWIN32_LEAN_AND_MEAN -q -w-aus -w-par -w-inl  -c -tWC -tWM -DOPENSSL_SYSNAME_WIN32 -DL_ENDIAN -DDSO_WIN
32 -D_stricmp=stricmp -O2 -ff -fp -DOPENSSL_NO_RC5 -DOPENSSL_NO_MDC2 -DOPENSSL_NO_KRB5
available timing options: TIMEB HZ=1000
timing function used: ftime
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
aes-128 cbc      30404.52k    33845.50k    32463.65k    32282.50k    33104.21k
aes-192 cbc      27971.35k    30294.72k    29970.02k    28154.42k    29810.26k
aes-256 cbc      24818.37k    27373.50k    26218.50k    25385.41k    27553.32k
```

操作模式

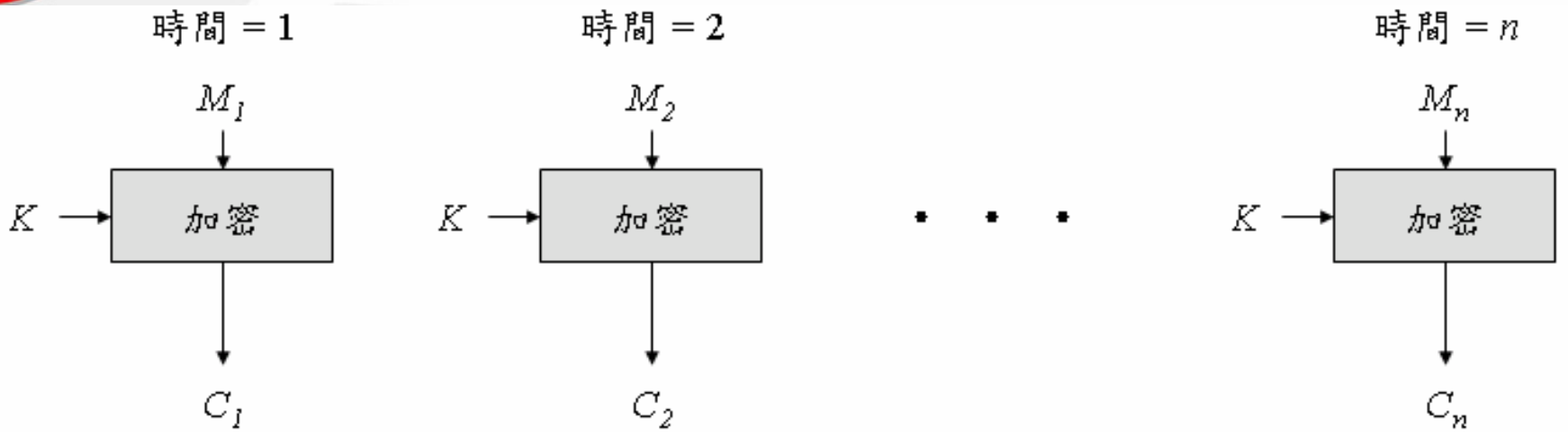
- 區塊加密器有五種操作模式用來提供保密性，包括：電子密碼本 (Electronic Codebook，ECB) 模式、計數器(Counter，CTR)模式、密文區塊鏈結(Cipher Block Chaining，CBC) 模式、密文反饋(Cipher Feedback，CFB)模式、輸出反饋(Output Feedback，OFB)模式
- http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

電子密碼本(Electronic Codebook，ECB)

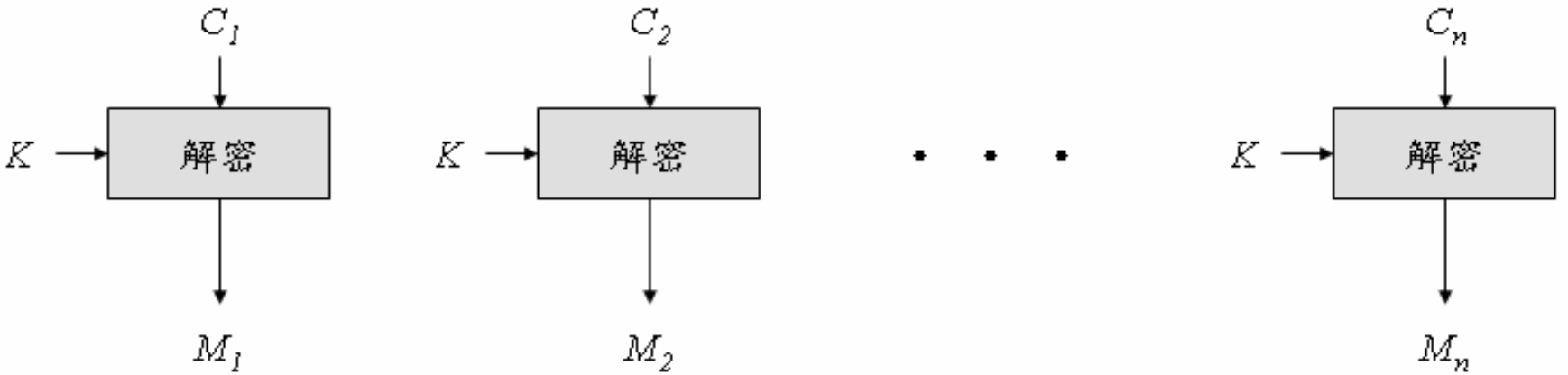
將欲加密的訊息 M 依區塊加密器明文的大小切割成 n 個相同長度的區塊，即 M_1, M_2, M_3, …, M_n，然後用同樣的金鑰一一將 n 個明文區塊加密，即得到 n 個與明文相對應的密文區塊 C_1, C_2, C_3, …, C_n

- 相同的明文區塊，經相同金鑰加密後會得到相同的密文區塊，就像是查表一樣
- 如果訊息M長度並非剛好區塊長度的倍數，則須將明文補齊(padding)並添加額外的位元，使得新長度為區塊長度的倍數
- 通常補齊的方式為先固定補一個“1”位元，然後再補零或多個“0”位元，使新長度為區塊長度的倍數

ECB加解密



(a) ECB模式加密

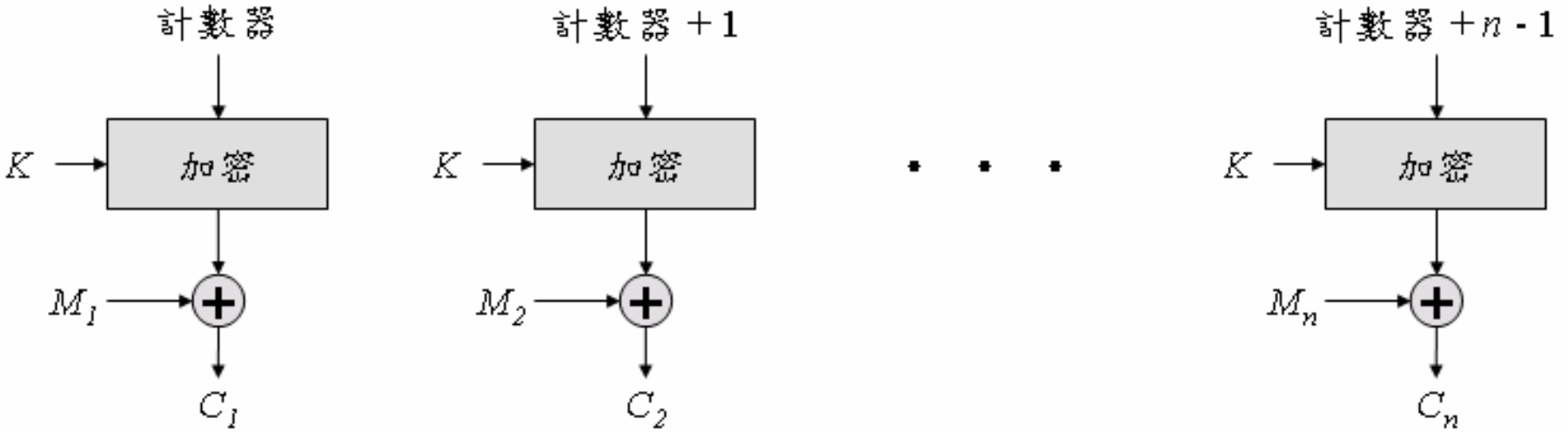


(b)ECB模式解密

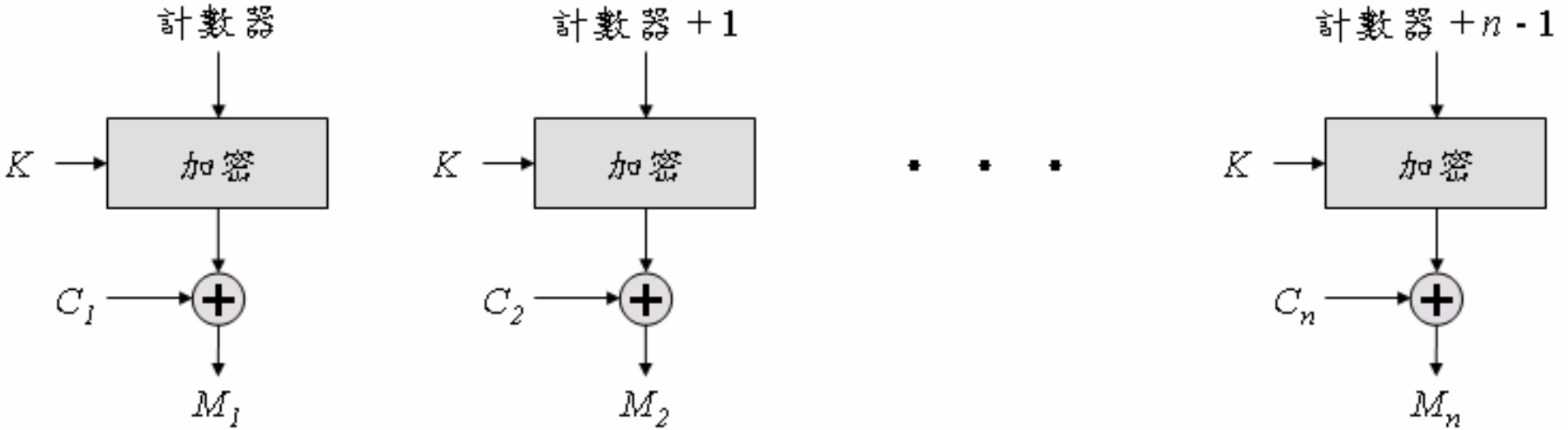
計數器(Counter，CTR)模式

- 計數器(Counter，CTR)模式，是2001年被提出的
- 此模式只用到區塊加密器的加密程序而不需要解密程序
- 在使用時須先設定一個計數器，邏輯上計數器的長度與明文長度相同，但實務上計數器僅須每次有不同的數值便可，並將其起始值當作明文，用金鑰進行區塊加密法加密，加密後的結果再與第一個區塊明文(M_1)做XOR (exclusive-OR)，便是第一個區塊密文(C_1)；第二次再將計數器值加一並用相同金鑰加密，加密後的結果再與第二個區塊明文(M_2)去做XOR，得到的結果就是第二個區塊密文(C_2)，其餘區塊依此類推。計數器可遞增、遞減、或做其他不重複的變化
- 計數器的優點：不需要區塊加密法的解密程序，並允許事先及平行處理來加速運算，原因是以計數器來作區塊加密法的加密程序，跟明文密文並無關，因此可事先進行並予以儲存，也可單獨平行進行，待取得明文時馬上進行XOR便可得到密文，所以效率會較高；另外，同一個明文區塊加密後，會產生不同密文區塊，且明文區塊不需按順序進行加密，可以單獨針對某區塊進行加解密

CTR加解密



(a)計數器模式加密

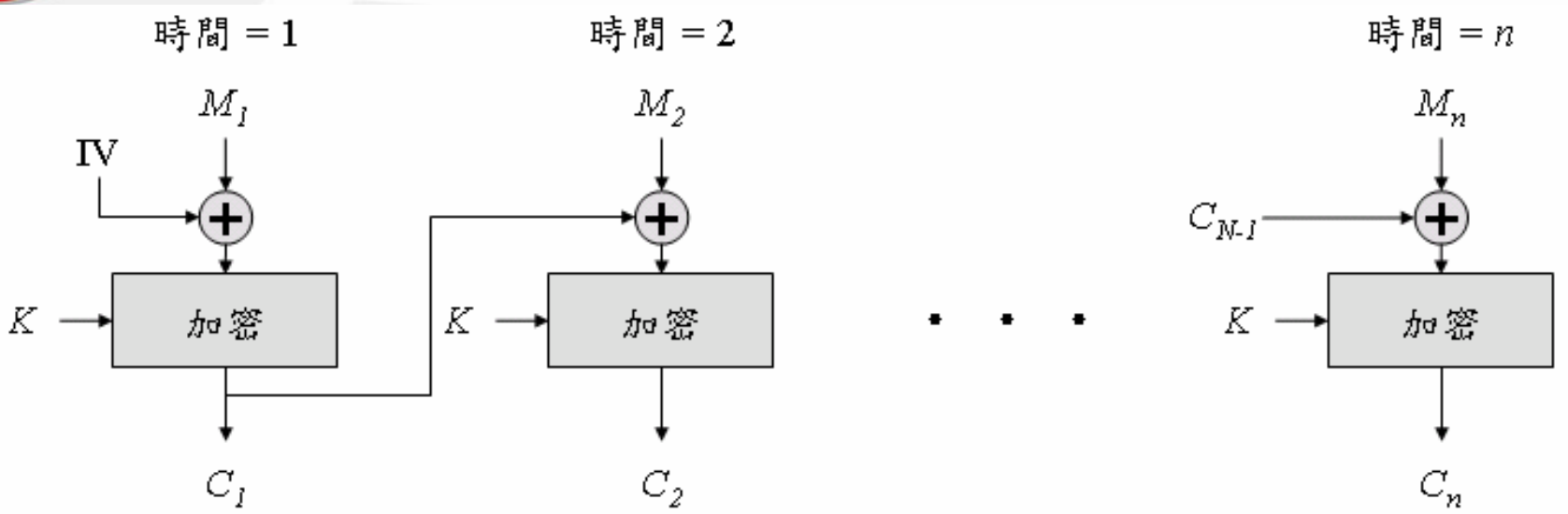


(b)計數器模式解密

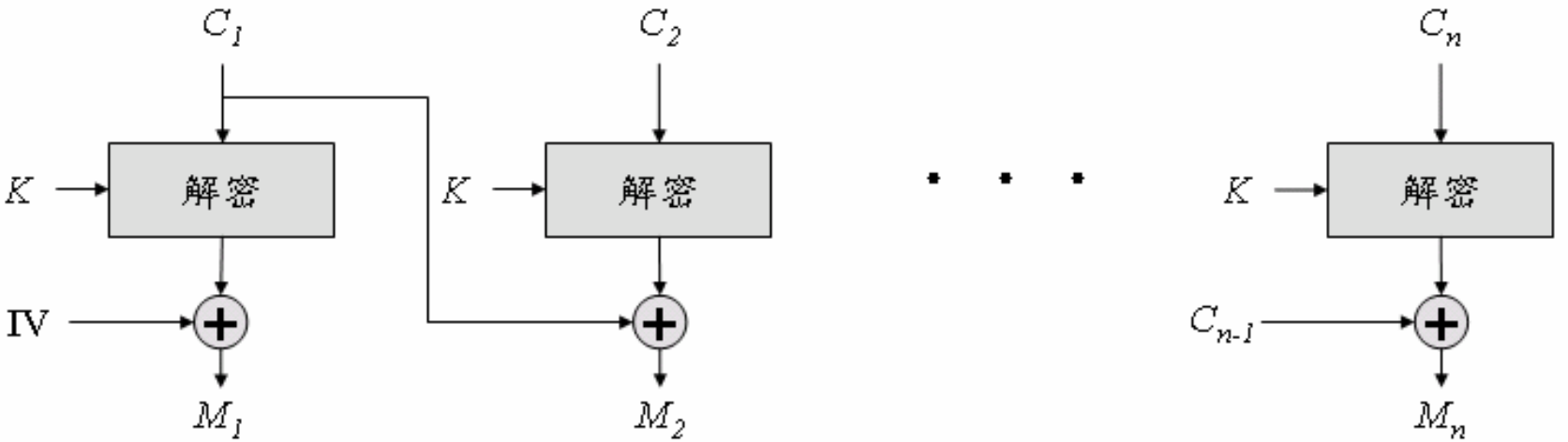
密文區塊鏈結(Cipher Block Chaining，CBC)

- 與區塊同長度的初始向量(initialization vector，IV)並非秘密值，例如可設定是加密的日期與時間或序號。
- 第一個區塊明文(M_1)中的每位元先與初始向量值做XOR運算後，再利用金鑰進行區塊加密程序，即得到第一個區塊密文(C_1)；而第二個區塊明文(M_2)需先與第一個區塊密文(C_1)作XOR運算後再做區塊加密，即可得到第二個區塊的密文(C_2)。亦即每次區塊在加密之前，皆要與上一個區塊的密文做XOR運算，然後再進行加密，依此類推，最後再將一個個區塊串鏈在一起，解密時亦同。
- 密文區塊鏈結模式的優點爲：可打亂明文與密文間一對一的關係，使得同樣的明文區塊加密後得到不同的密文區塊。

CBC加解密



(a)加密

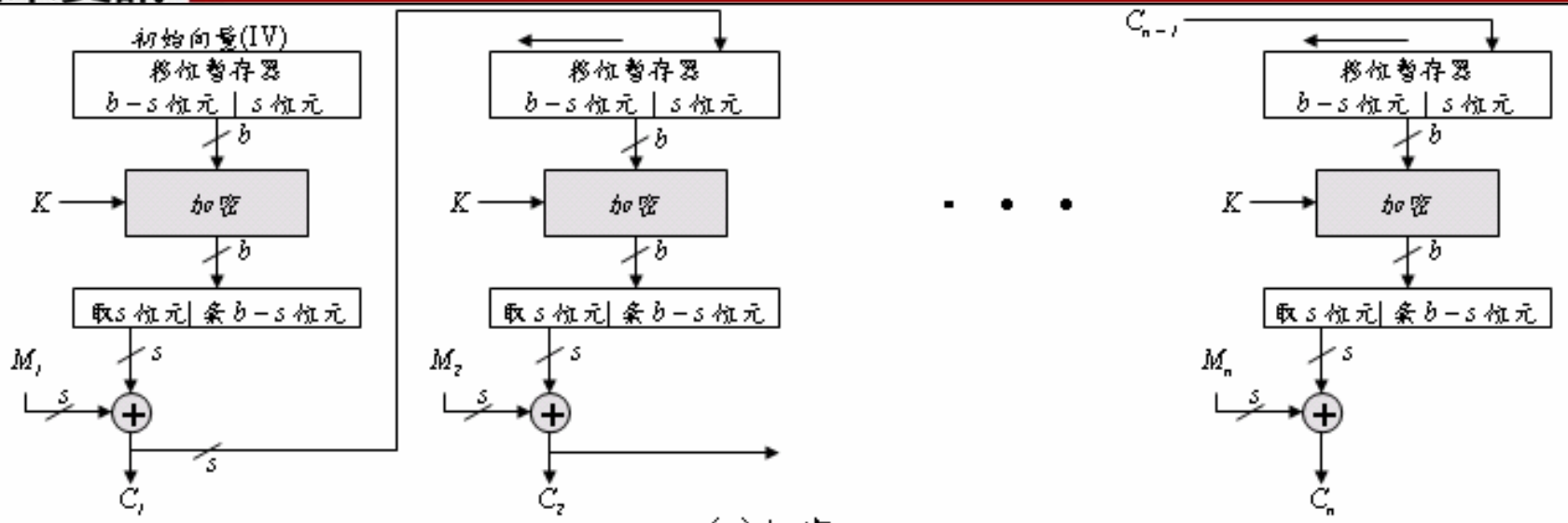


(b)解密

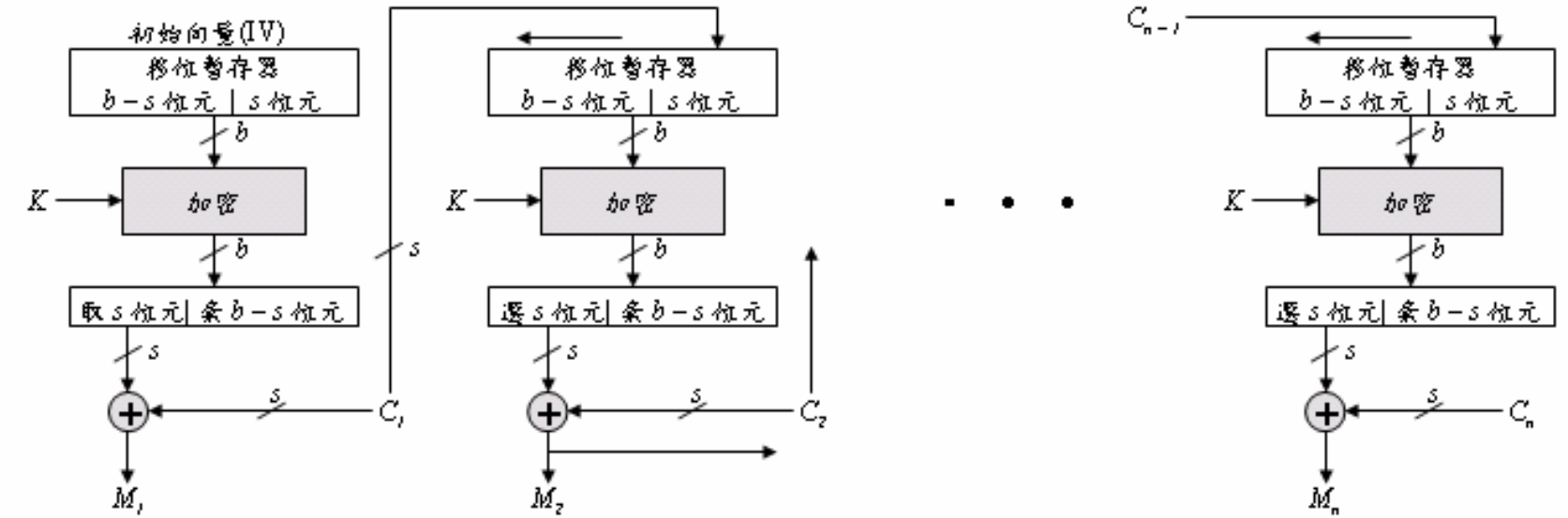
密文反饋(Cipher Feedback，CFB)模式

- 此模式是將區塊加密法轉換成串流加密法
- 首先設定一個與區塊同樣長度的移位暫存器(shift register)，此暫存器可以向左移動位元，且移位暫存器需預先設定一個初始向量。例如區塊加密法的區塊長度爲*b*位元，而*b* = 128 (AES加密法)
- 在加密時，首先將初始向量值用金鑰進行區塊加密法的加密程序，再從加密法輸出的*b*位元中，固定挑選出其中的*s*位元並與明文*s*位元(M_1)進行XOR運算，即得到密文*s*位元(C_1)，通常*s*爲1或8，代表明文一次加密一位元或一位元組。接著將移位暫存器向左移動*s*位元而右邊空出的*s*位元則放C_1，再將移位暫存器中的新內容用金鑰進行區塊加密法的加密程序，再從加密法輸出的*b*位元中，同樣挑選出其中的*s*位元並與明文*s*位元(M_2)進行XOR運算，以得到密文*s*位元(C_2)，依此類推，每次加密*s*位元。此模式可用在數據機之類的通信加解密
- CFB模式在解密時，同樣先將移位暫存器設定與上述加密時相同的初始向量值，再用金鑰進行區塊加密法的加密程序，再從輸出的*b*位元中，固定挑選出其中*s*位元並與密文*s*位元(C_1)進行XOR運算，即得到明文*s*位元(M_1)。而後將移位暫存器向左移動*s*位元而右邊空出的*s*位元則放C_1，且將移位暫存器中的新內容用金鑰進行區塊加密法的加密程序，再從加密法輸出的*b*位元中，同樣挑選出其中*s*位元並與密文*s*位元(C_2)進行XOR運算，得到明文*s*位元(M_2)，依此類推，每次解密*s*位元

CFB加解密



(a)加密

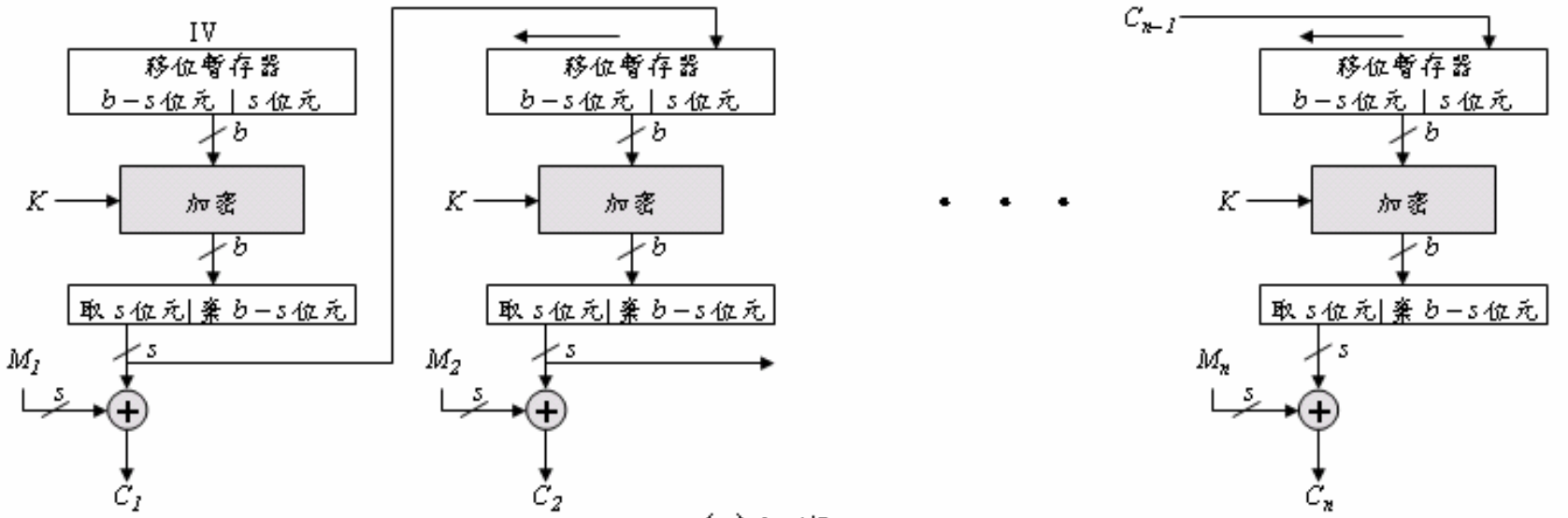


(b)解密

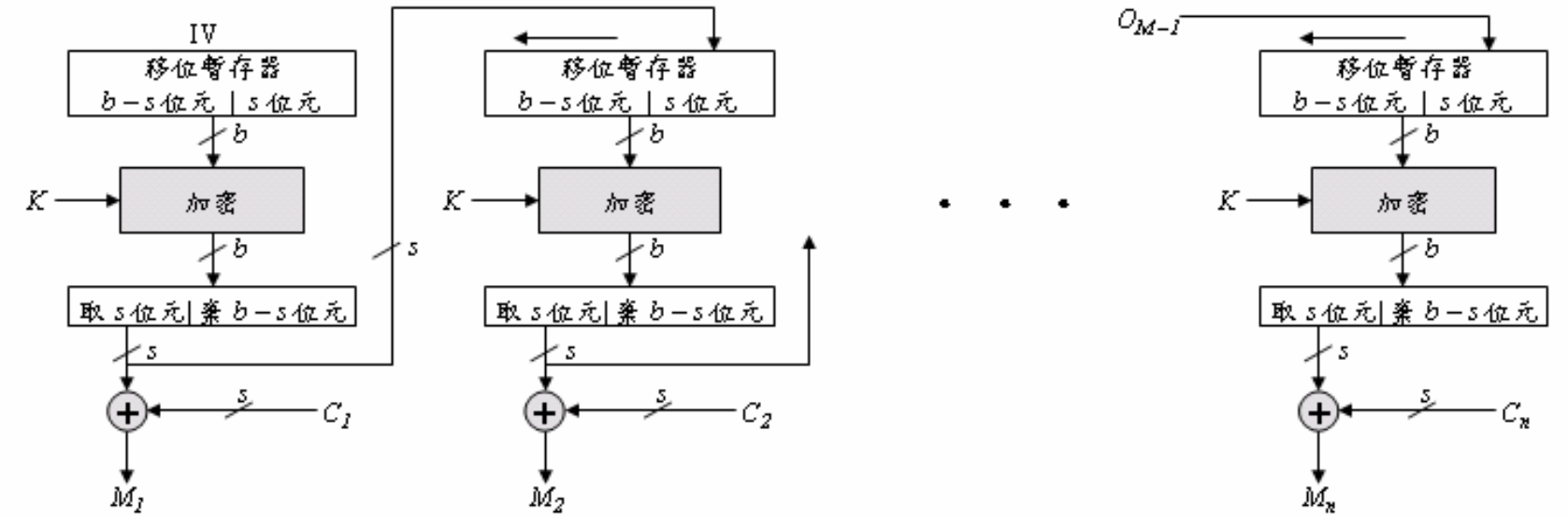
輸出反饋(Output Feedback，OFB)模式

- 同樣將區塊加密法轉換成串流加密法
- 與CFB模式的差別僅在於CFB模式是將區塊加密法所輸出結果先與明文做XOR得到密文後，再將密文放入移位暫存器，而OFB模式則將從區塊加密法所輸出的結果直接放入移位暫存器

OFB加解密



(a)加密



(b)解密

串流加密法(Stream Ciphers)

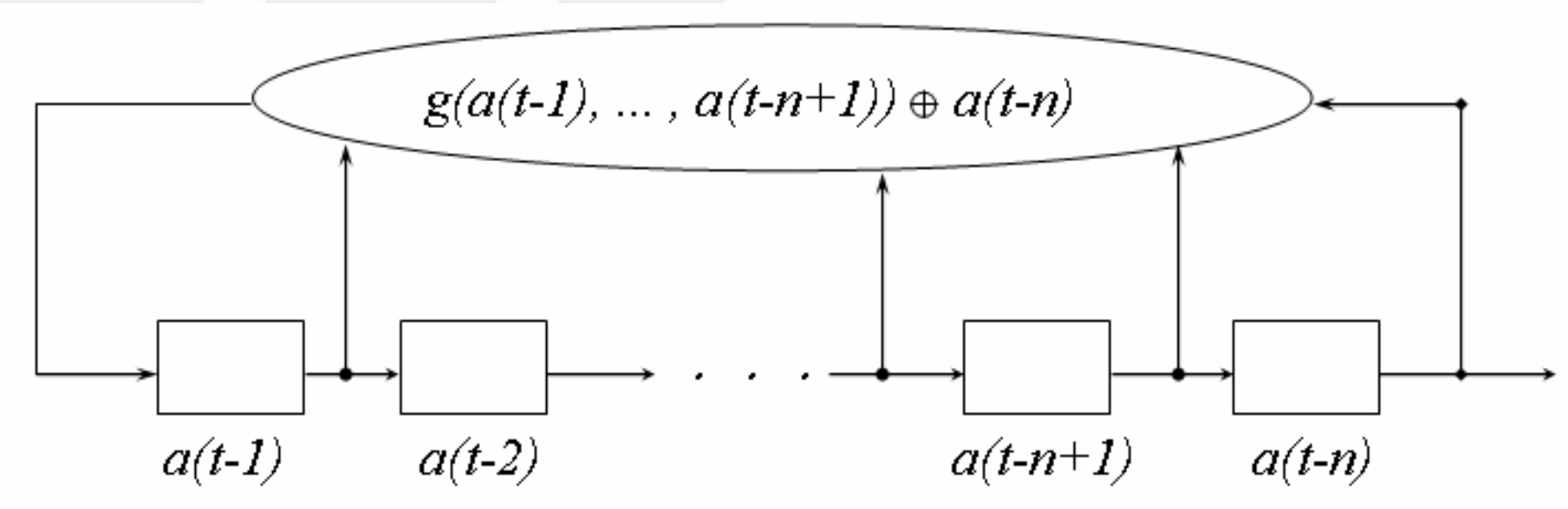
- 串流加密基本上就是要從一個短的金鑰，進而產生一組次序夠亂且長度夠長的二元訊號，「次序夠亂又難猜測」是串流密碼學所追求的目標
- 藉由虛擬亂數位元產生器(pseudo-random bit generator，PRBG)來產生金鑰序列(keystream)在目前是非常廣泛的，而所謂亂數就是無法預測，就一個亂數序列來說，要有很大序列週期，不同長度樣式要平均分散在序列上，一般來說，金鑰序列產生器有三項安全需求：
  - 1.金鑰序列週期要夠大，超過訊息長度
  - 2.金鑰序列要容易計算產生
  - 3.金鑰序列難被預測
- 假設一個虛擬亂數位元產生器最初有*n*位元輸出，*a(0)*，*a(1)*,…., *a(n-1)*，在計算上它應該是無法由前面的*n*位元推算出*n+1*位元，也就是*a(n)*是0或1的機率該各佔百分之50。
- 到目前為止，沒有一套世界標準與檢測方法可用來確定虛擬亂數位元產生器是否難以被預測，除了用各種已知攻擊方式去嘗試外，還沒有破解的理論存在

線性同餘產生器(Linear congruence generator，LCG)

- 遞迴形式：　$X_{i+1} = a\,X_i + b \bmod m$
  - (a，b，m)代表亂數產生器的參數也可當作私密金鑰，X_0作爲亂數產生器的初始值
  - 如果參數經過適當挑選，X_i值就會介於[0，m-1]區間，而區間所有整數都一一出現後才會重複
- 例如一個數列$X_i = 5X_{i-1} + 3 \bmod 16$且$X_0$=1產生的序列是{1，8，11，10，5，12，15，14，9，0，3，2，13，4，7，6，1，8，……}
- 然而LCG產生的序列在密碼學上是不夠安全的，因爲只要給一小段充足的序列長度，就可以推算出參數m，a，b。截短的LCG (僅使用前面幾個位元) 也已經被證實是不安全

反饋移位暫存器(Feedback shift register，FSR)

- 由*n*個正反器(flip-flop)以及反饋函數(feedback function)所組成
- $a(t) = g(a(t-1)，a(t-2)，\ldots，a(t-n+1)) \oplus a(t-n)$



FSR

- 根據*g*可以決定FSR是線性反饋移位暫存器(Linear Feedback Shift Register，LFSR)，或是非線性反饋移位暫存器(Non-Linear Feedback Shift Register，NLFSR)
- *g*若是線型函數則稱爲LFSR，反之則稱爲NLFSR
- 在FSR中每一個單獨儲存元件稱作階(stage)
- 二元訊號*a(0)*，*a(1)*，*a(2)*，…..*a(n-1)*儲存在暫存器做爲初始值，暫存器的內容稱爲狀態
- FSR序列的週期是由階數與反饋連接方式所決定
- *n*階FSR序列所能產生的最大週期是2^n

LFSR反饋函數

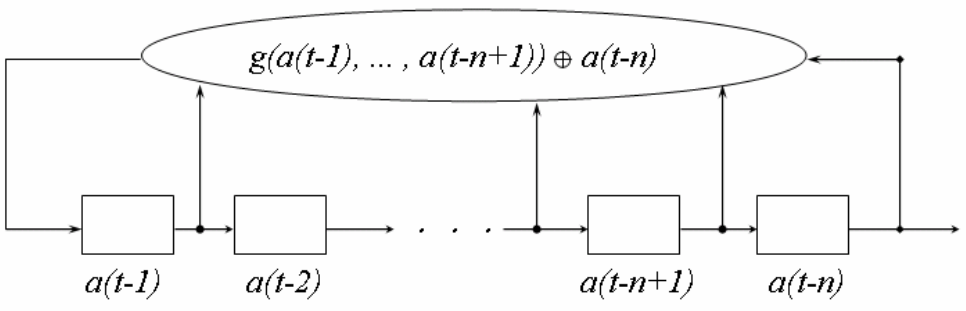
- 線性反饋移位暫存器(LFSR)的反饋函數形式爲:

$$a(t)=c_1a(t-1) \oplus c_2a(t-2) \oplus \ldots \oplus c_{n-1}a(t-n+1) \oplus a(t-n)$$

- 係數c_i爲0或1，而⊕代表XOR
- LFSR的反饋可以用反饋多項式(feedback polynomial) $f(x)$表示

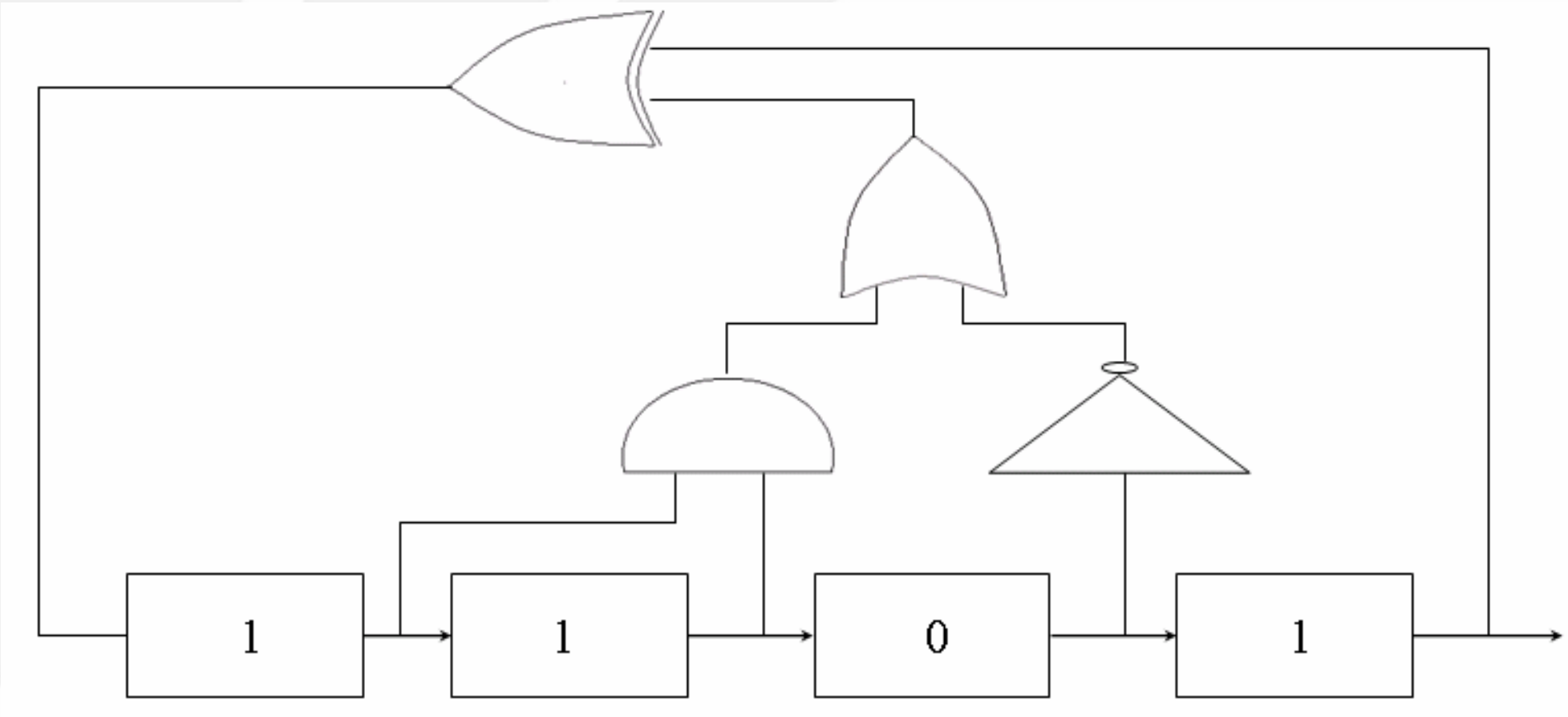
$$f(x)=1+c_1x+c_2x^{n-2}+\ldots+c_{n-1}x^{n-1}+x^n$$

- 反饋多項式決定輸出序列的週期與統計性質，而LFSR的初始值不可全部是零。



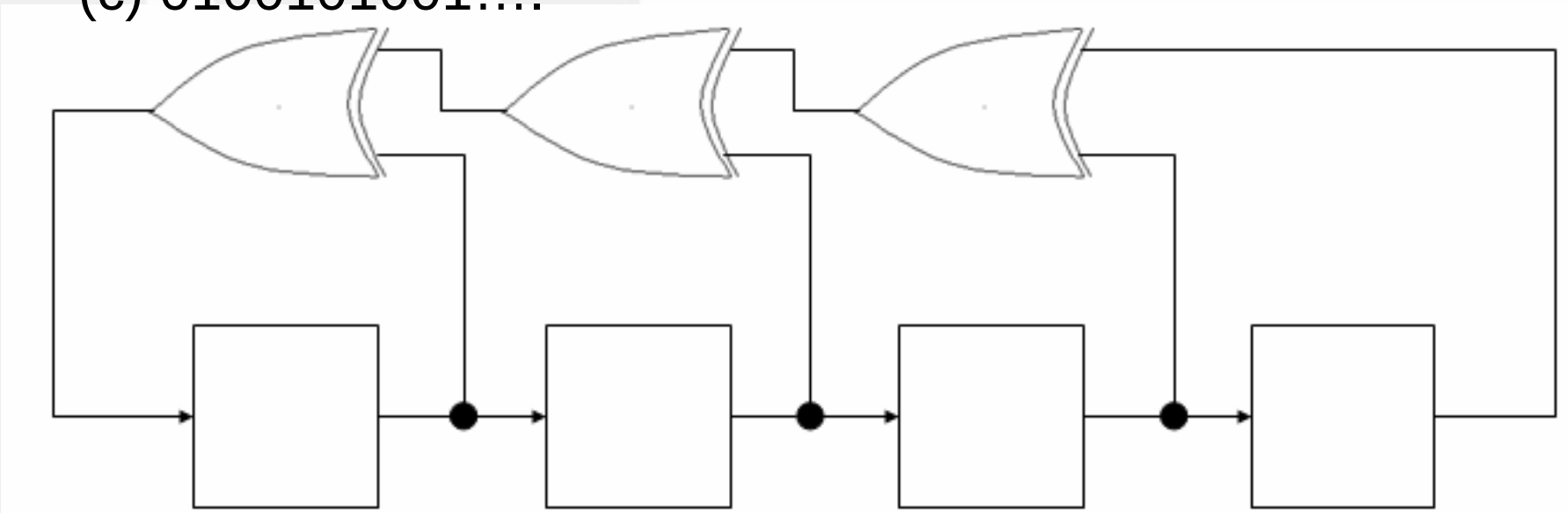
4階非線性反饋移位暫存器(NLFSR)

- 4階NLFSR產生週期爲2^4 =16的De Bruijn序列

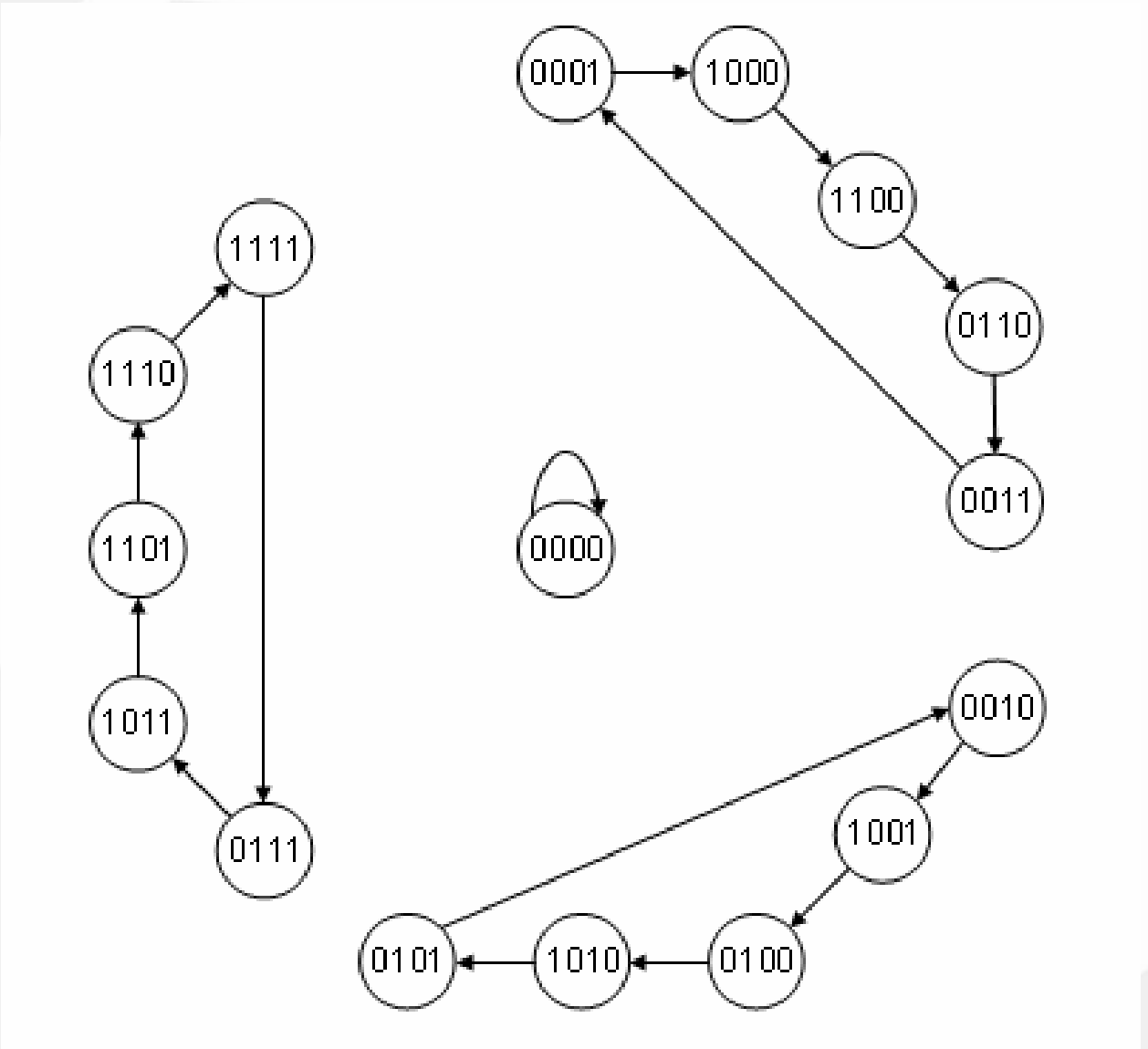


4階線性反饋移位暫存器 (LFSR)-1

- 反饋多項式爲 *f(x)=1+x+x2+x3+x4*時，我們載入不同LFSR初始值，將產生以下三組統計性質不佳而週期爲5的序列。
  - (a) 1111011110…..
  - (b) 1000110001…..
  - (c) 0100101001….

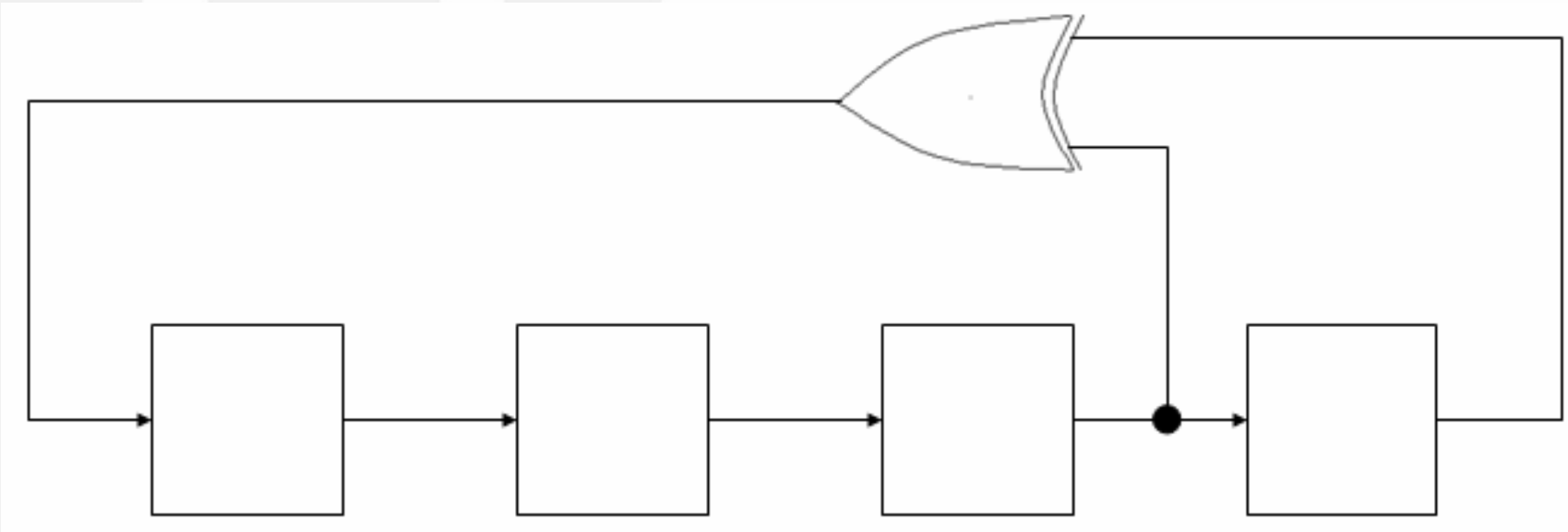


4階線性反饋移位暫存器 (LFSR)-1狀態圖

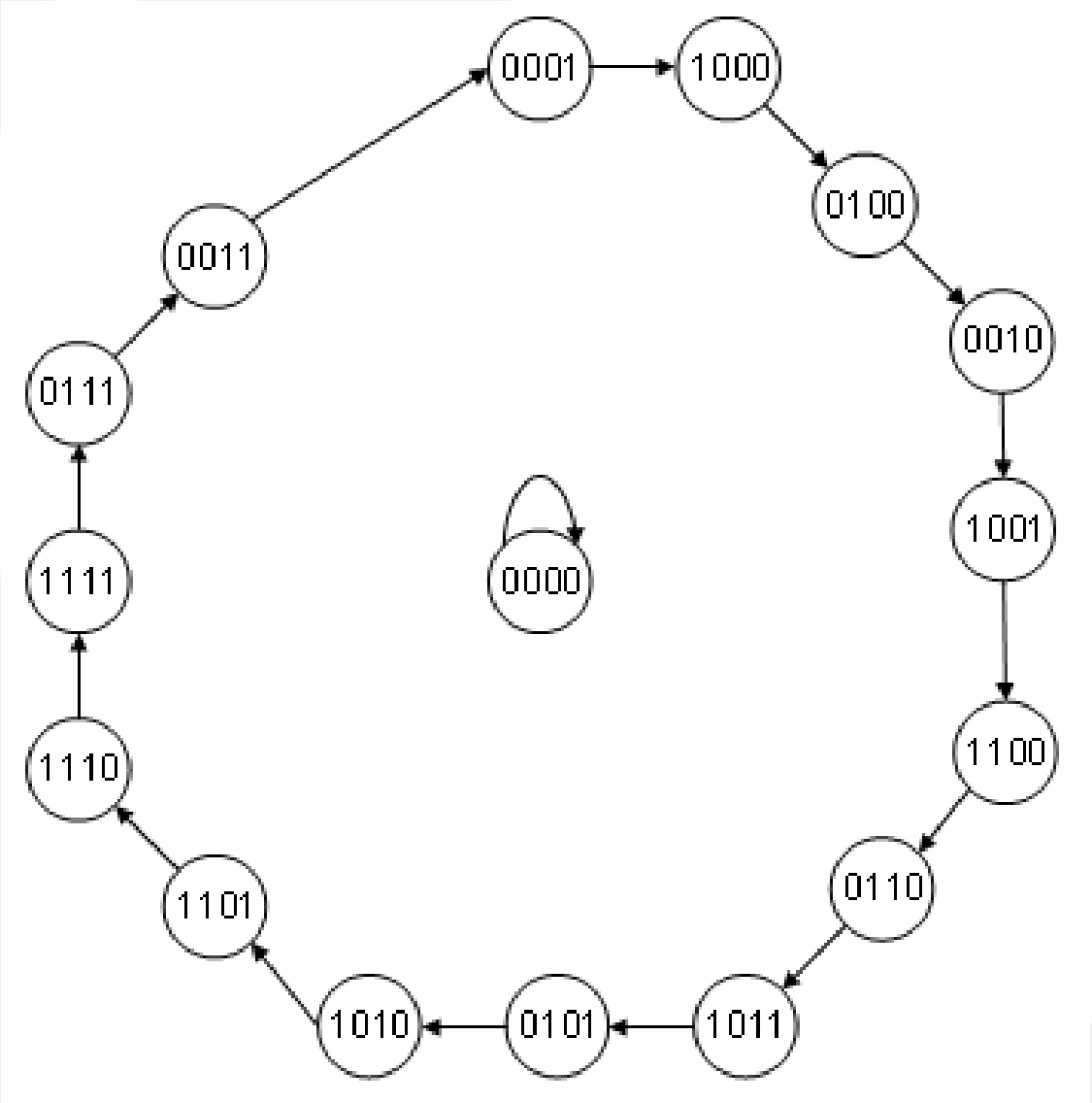


4階線性反饋移位暫存器 (LFSR)-2

- 如果4階LFSR反饋多項式爲$f(x)=1+x+x^4$，我們載入初始值將產生週期爲15且擁有最佳統計性質的序列: 101100100011110 。



4階線性反饋移位暫存器 (LFSR)-2狀態圖



m-序列

- 一個*n*階LFSR最多有2^n狀態，但若初始狀態全爲零，則正反器輸入時鐘脈衝後狀態恆爲零，所以*n*級LFSR狀態週期小於等於2^n -1。
- 通常爲確保輸出序列有最大可能週期2^n-1，LFSR的反饋多項式應該是本原(primitive)，這表示X^T-1被*f(x)*整除的最小正整數T = 2^n-1。*n*級本原多項式的數量等於 $\varphi((2^n-1)/n)$，其中φ就是歐拉函數(Euler function)。
- 一個由擁有本原反饋多項式的LFSR產生的序列被稱叫做最大長度LFSR序列(maximal-length LFSR sequence)，或是簡單稱爲*m*-序列。*m*-序列有長週期與好的統計性質。

線性複雜度 (Linear complexity)

- *n*階LFSR的秘密金鑰(初始值與反饋連接)可以從連續*2n*位元的輸出序列得出。
- 假設我們已獲得部份*2n*位元*a(t), a(t+1),...,a(t+2n-1)*，那麼我們可以將*a(t+n)*，……，*a(t+2n-1)*等*n* 位元各由前面*n*位元來表示:

$$a(t+n) = a(t)C_0 \oplus a(t+1)C_1 \oplus \ldots \oplus a(t+n-1)C_{n-1}$$
$$a(t+n+1) = a(t+1)C_0 \oplus a(t+2)C_1 \oplus \ldots \oplus a(t+n)C_{n-1}$$
$$\ldots$$
$$a(t+2n-1) = a(t+n-1)C_0 \oplus a(t+n)C_1 \oplus \ldots \oplus a(t+2n-2)C_{n-1}$$

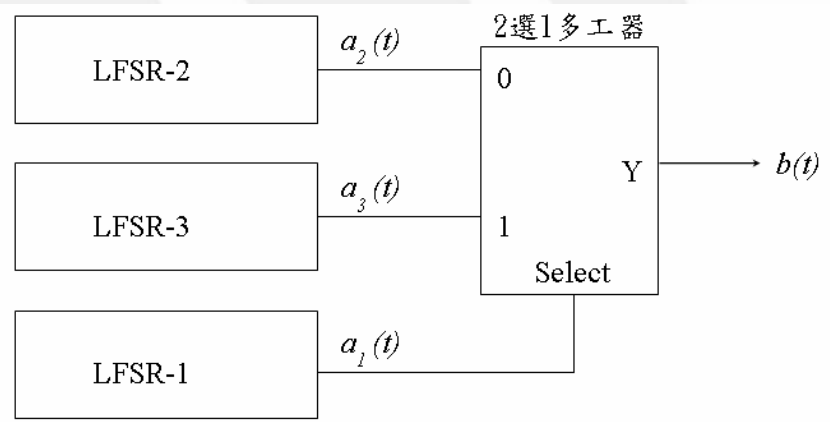
- 只要解開上述線性代數方程式中未知的$c_1, c_2, \ldots, c_{n-1}$就可以得到反饋常數，進而算出序列的任何*位元*
- 每一個週期序列**a**可以由某種階數的LFSR產生，而我們有技術找出可以產生序列**a**的最短LFSR的反饋多項式，像這樣一個LFSR階數被稱爲序列**a**的線性複雜度(linear complexity)，以LC(**a**)表示。
- 密碼學上的虛擬亂數產生器必須擁有與金鑰無關而又夠大的線性複雜度。

虛擬亂數產生器設計技巧

- 在公開文獻已有許多技術可保證金鑰序列的不可預期性
- 兩種簡單的金鑰序列設計技術：
  - 非線性前饋轉換(nonlinear feed-forward transformation)
  - 步進控制(step control)

Geffe產生器

- Geffe產生器是組合三個LFSR的簡單方法，用LFSR-1的輸出送到2選1多工器(multiplexer)去挑選LFSR-2的輸出或LFSR-3的輸出
- 如果LFSR-1，LFSR-2，LFSR-3的本原反饋多項式的級數分別是*n1*，*n2*，*n3*，那麼Geffe產生器的線性複雜度LC = $n3 \times n1+(n1+1)n2$，序列週期T=最小公倍數($2^{n1}-1$，$2^{n2}-1$，$2^{n3}-1$)



攻擊Geffe產生器

- Geffe產生器的安全弱點來自

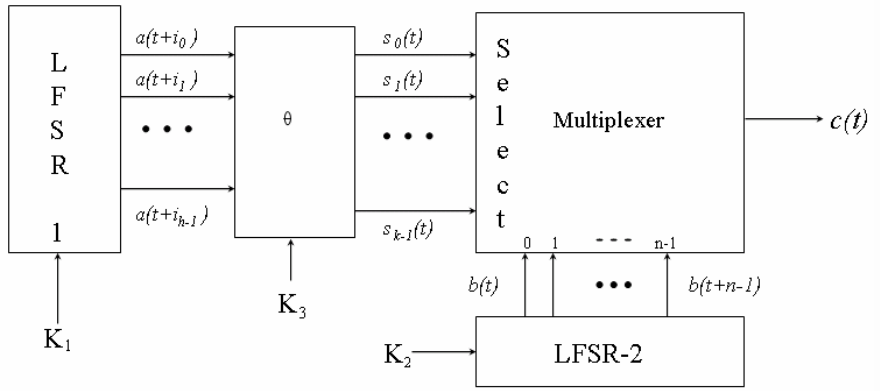
$$Prob(b(t)=a_2(t))=Prob(a_1(t)=0)+Prob(a_1(t)=1)\times Prob(a_3(t)=a_2(t))$$

$$=\frac{1}{2}+\frac{1}{4}=\frac{3}{4}$$

- $b(t)$與$a_3(t)$相等之機率同樣可以被估計。
- 如果LFSR的反饋多項式已知，我們可利用這75%的機率輕易破解Geffe產生器
- 見Kencheng Zeng, Chung-Huang Yang, Dah-Yea Wei and T.R.N. Rao, "Pseudorandom Bit Generators in Stream-Cipher Cryptography," *IEEE Computer*, Vol. 24, No. 2, February 1991, pp. 8-17

Jennings產生器

- 使用多工器組合兩個階數分別爲*l*與*n*的LFSR



我們首先固定一個正整數$h \le \min(l, [\log_2 n])$，且$0 \le i_0 < i_1 < \ldots < i_{n-1} \le l-1$。

對每一個時間 $t \ge 0$，LFSR-1 形成的數字為 $U(t)=a(t+i_0)+a(t+i_1)2+\ldots+a(t+i_{n-1})2^{n-1}$

可將他轉換為 $\theta(u(t))=s_0(t)+s_1(t)2+\ldots+s_{k-1}(t)2^{k-1}$

$k=\lceil \log_2 n \rceil$且 θ 為將$\{0,1,\ldots,2^n-1\}$內射到$\{0,1,\ldots,n-1\}$。假設我們已知 LFSR-1，LFSR-2 的本原反饋多項式，那麼 Jennings 產生器的秘密金鑰由 θ 函數跟兩個 LFSR 的初始狀態 K_1，K_2 組成。輸出訊號 $c(t)=b[t+\theta(u(t))]$

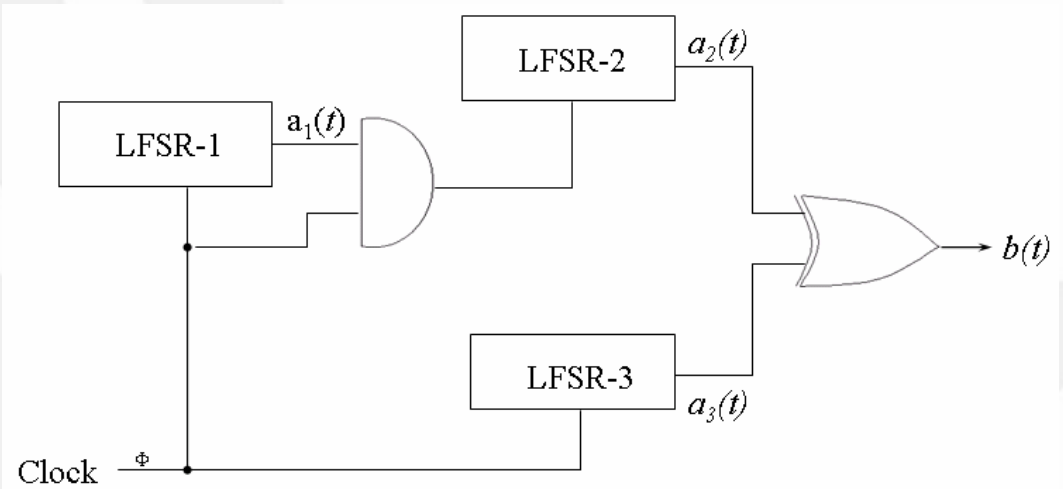
如果最大公因數(l，n)=1，那麼輸出序列週期為$(2^l-1)(2^n-1)$，線性複雜度為

$$LC \le n(1+\sum_{i=1}^{h} C_i^l) \quad 。$$

假如所有 LFSR 的反饋多項式為已知，研究發現 Jennings 產生器可經 2^{l+h} 工作量針對 K_1 攻擊而找出所有祕密金鑰。

步進控制

- 所謂步進控制即是將LFSR的時鐘脈衝控制前進或停止，步進控制有多種不同實作方法。
- Beth-Piper停止與前進(stop-and-go)產生器， LFSR-2時脈輸入被LFSR-1輸出控制，因此，LFSR-2僅有在$a_1(t-1)=1$，才可以改變狀態。
- 在三個組成的LFSR 級數分別是$n1$，$n2$，$n3$的適當狀況下，此產生器的輸出序列的線性複雜度LC=$(2^{n1}-1)n_2+n_3$，週期T=$(2^{n1}-1)(2^{n2}-1)(2^{n3}-1)$。



攻擊Beth-Piper停止與前進產生器

- Beth-Piper停止與前進產生器輸出序列已經有夠大線型複雜度，但安全性仍不夠。
- 若*a'(t)*表示不受步進控制的LFSR-2在時間*t*的輸出訊號，那麼我們有機率

$$Prob\left(b(t)\oplus b(t+1)=a_3(t)\oplus a_3(t+1)\right)$$
$$=Prob\left(a_1(t)=0\right)+Prob\left(a_1(t)=1\right)Prob\left(a_2(t)=a_2(t+1)\right)$$
$$=\frac{1}{2}+\frac{1}{4}=\frac{3}{4}$$

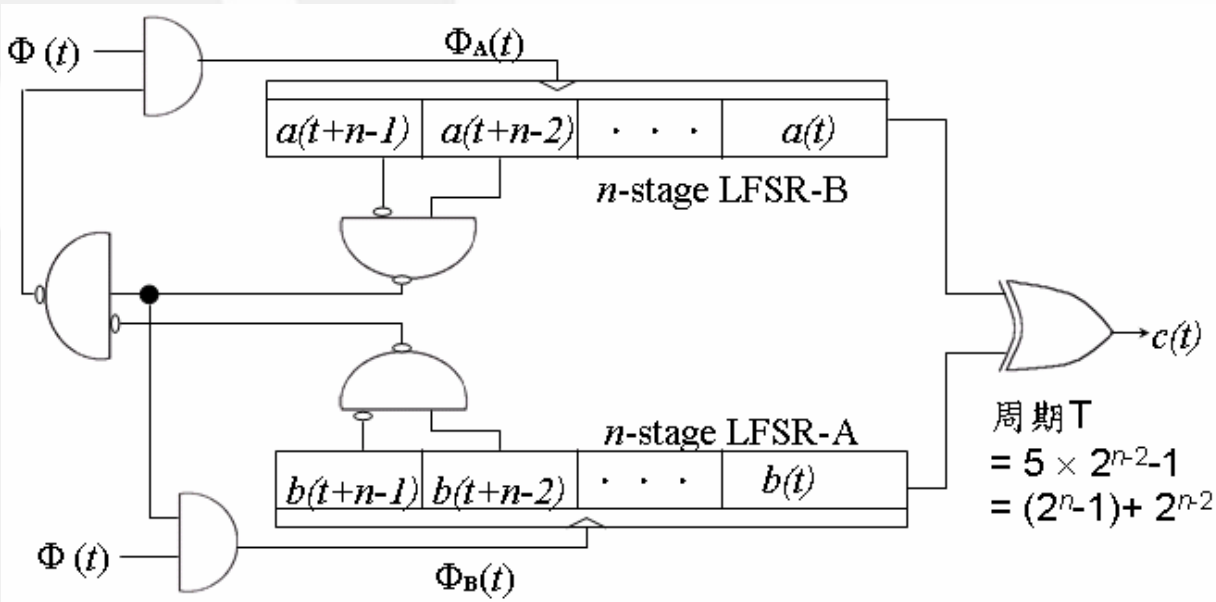
- 假設LFSR-1與LFSR-3的反饋多項式爲已知，研究人員發現可以從序列**b**求得序列$\mathbf{a_3}$，然後再從$\mathbf{a_2}$求得$\mathbf{a_1}$，且無須假設LFSR-2的反饋多項式是否爲已知

周期爲質數的序列

- 如果一個二元序列**b**的週期T為質數，那麼**b**的線性複雜度一定滿足 $LC(\mathbf{b}) \geq Ord_T(2)$
- $Ord_T(2)$代表2模T的階(order)，也就是$2^x \bmod T = 1$成立的最小正整數x
- 一個質數週期的序列可以再進行任何的轉換運算，只要運算後的序列不是常數序列就不會影響到已建立最小線性複雜度
- 如果週期$T=2^n-1$成為梅森質數，那麼$Ord_T(2)=n$非常小，所以週期T應該選擇異於2^n-1形式的質數
- 序列週期形式為$q\times 2^m-1$，奇數$q \geq 3$，可從一對階數為n的LFSR建構

兩邊停止與前進(bilateral stop-and-go)產生器

- 序列週期為$5\times2^{n-2}-1$，兩LFSR彼此互相控制的方式如下：
  - 如果$(a(t+n-1)，a(t+n-2))=(0，1)$，那麼LFSR-B的時鐘脈衝會被停止。
  - 如果$(b(t+n-1)，b(t+n-2))=(0，1)$，但是$(a(t+n-1)，a(t+n-2))\neq(0，1)$，那麼LFSR-A的時鐘脈衝會被鎖住

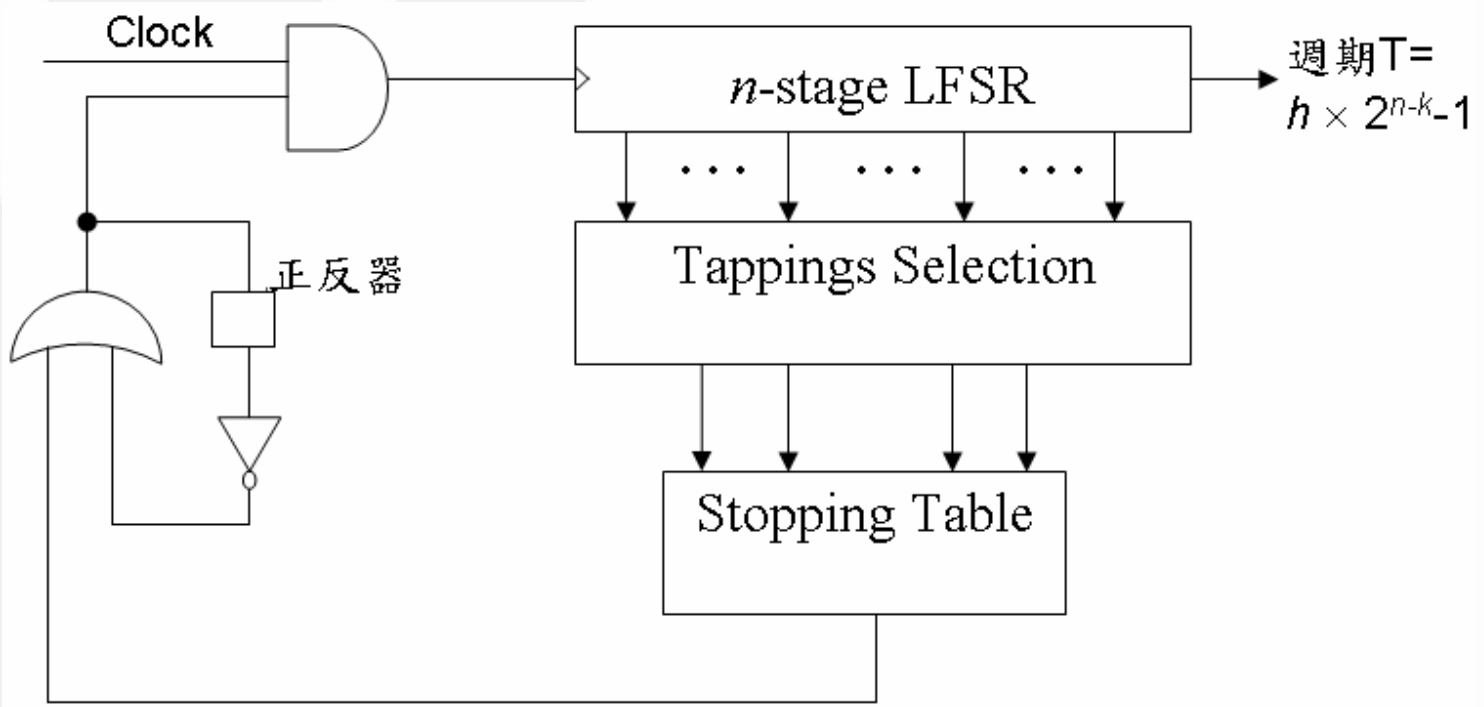


兩邊停止與前進產生器的週期與最小線性複雜度

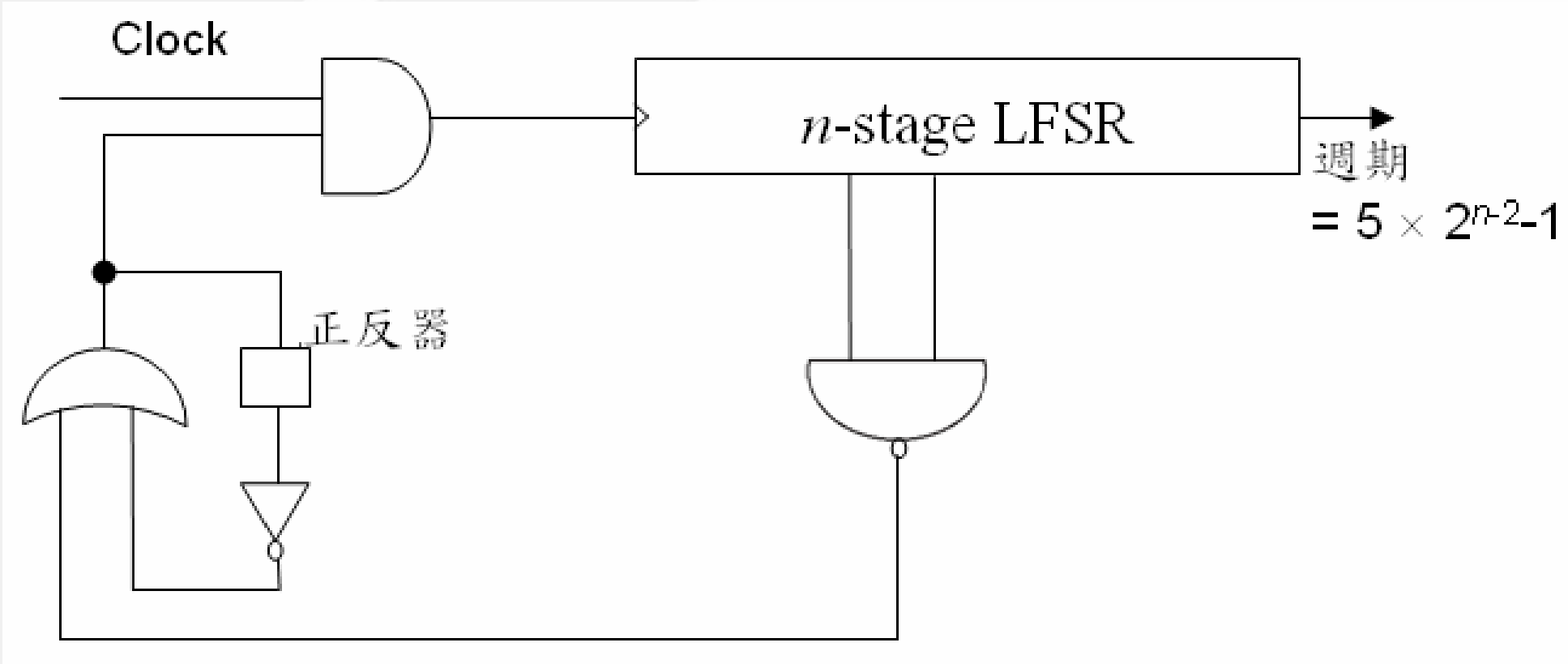
n	周期 T	最小線性複雜度 $Ord_T(2)$
4	19	18
6	79	39
10	1279	639
12	5119	2559
14	20479	10239
16	81919	40959
20	1310719	218453
34	2.147×10^{10}	1.074×10^{10}
50	1.407×10^{15}	7.037×10^{14}

自我停止與前進(self stop-and-go)產生器

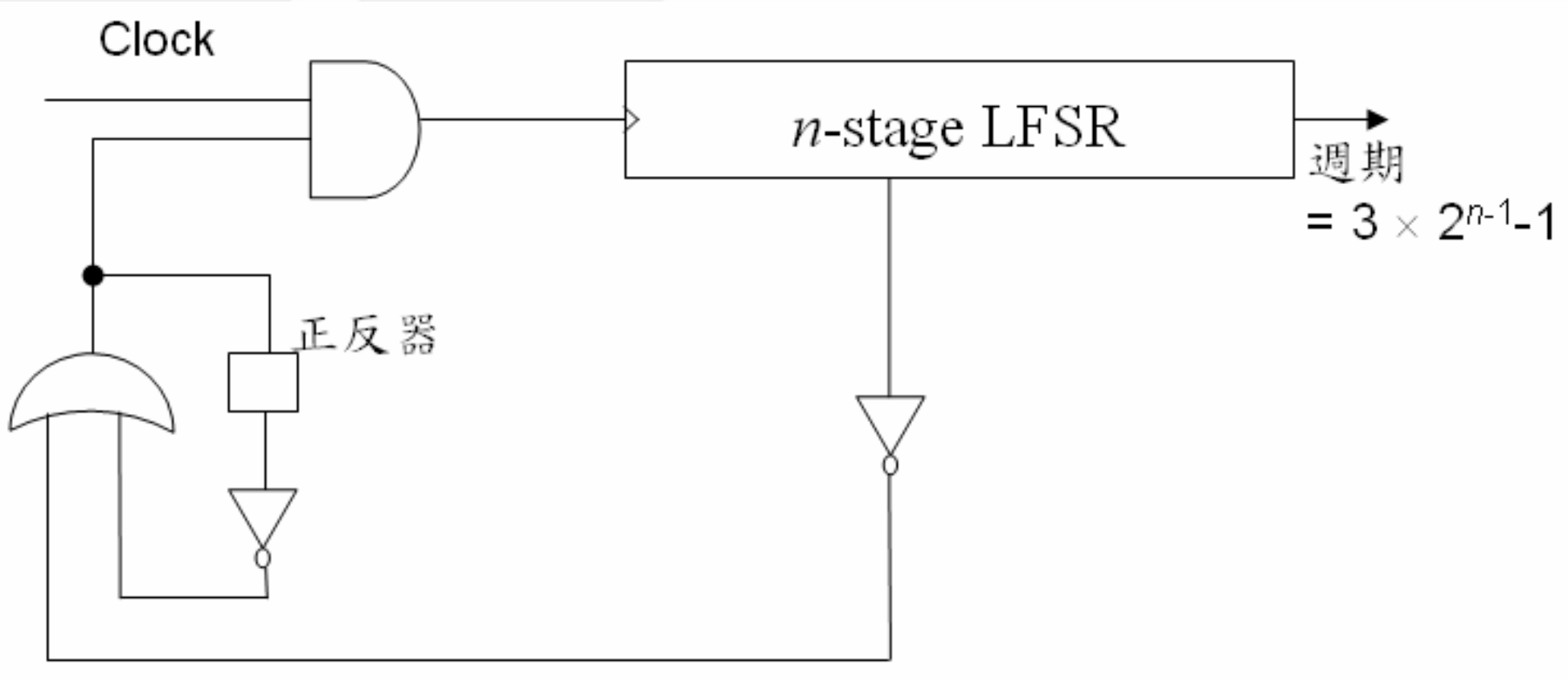
- 序列週期形式為$q \times 2^m$-1，奇數q3，也可由單一階數為n的LFSR建構
- LFSR自我停止與前進(stop-and-go)。當停止表(stopping table)的輸出為零時，LFSR會在下一個時鐘脈衝時停止，但LFSR不會連續停兩時鐘脈衝



自我停止與前進(self stop-and-go) $5\times2^{n-2}-1$週期產生器



自我停止與前進(self stop-and-go) $3\times2^{n-1}-1$週期產生器



參考資料

- 中華民國國家標準 CNS X5011，數據保密(加/解密)運算法，1983年公佈。
- 賴溪松、韓亮、張真誠，近代密碼學及其應用，旗標出版股份有限公司，2003年。
- 王旭正、柯宏叡，密碼學與網路安全，博碩文化股份有限公司，2004年。
- 丁存生、肖國鎮，流密碼學及其應用，國防工業出版社，1994年。
- D.E. Knuth, *The Art of Computer Programming – Seminumerical Algorithms*, Vol. 2, 3rd edition, Addison-Wesley, 1998.
- A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press Series on Discrete Mathematics and Its Applications, 1996.
- National Institute of Standards and Technology, *Advanced Encryption Standard (AES)*, Federal Information Processing Standard (FIPS) 197, November 26, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- National Institute of Standards and Technology, Federal Information Processing Standard (FIPS) 46-3, *Data Encryption Standard (DES)*, October 1999. 2005年5月撤案。 http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
- National Institute of Standards and Technology, Special Publications SP 800-38A, *Recommendation for Block Cipher Modes of Operation - Methods and Techniques*, December 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/ sp800-38a.pdf
- N. Smart, *Cryptography: An Introduction*, McGraw-Hill, 2003.
- Kencheng Zeng, Chung-Huang Yang, Dah-Yea Wei and T.R.N. Rao, "Pseudorandom Bit Generators in Stream-Cipher Cryptography," *IEEE Computer*, Vol. 24, No. 2, February 1991, pp. 8-17.