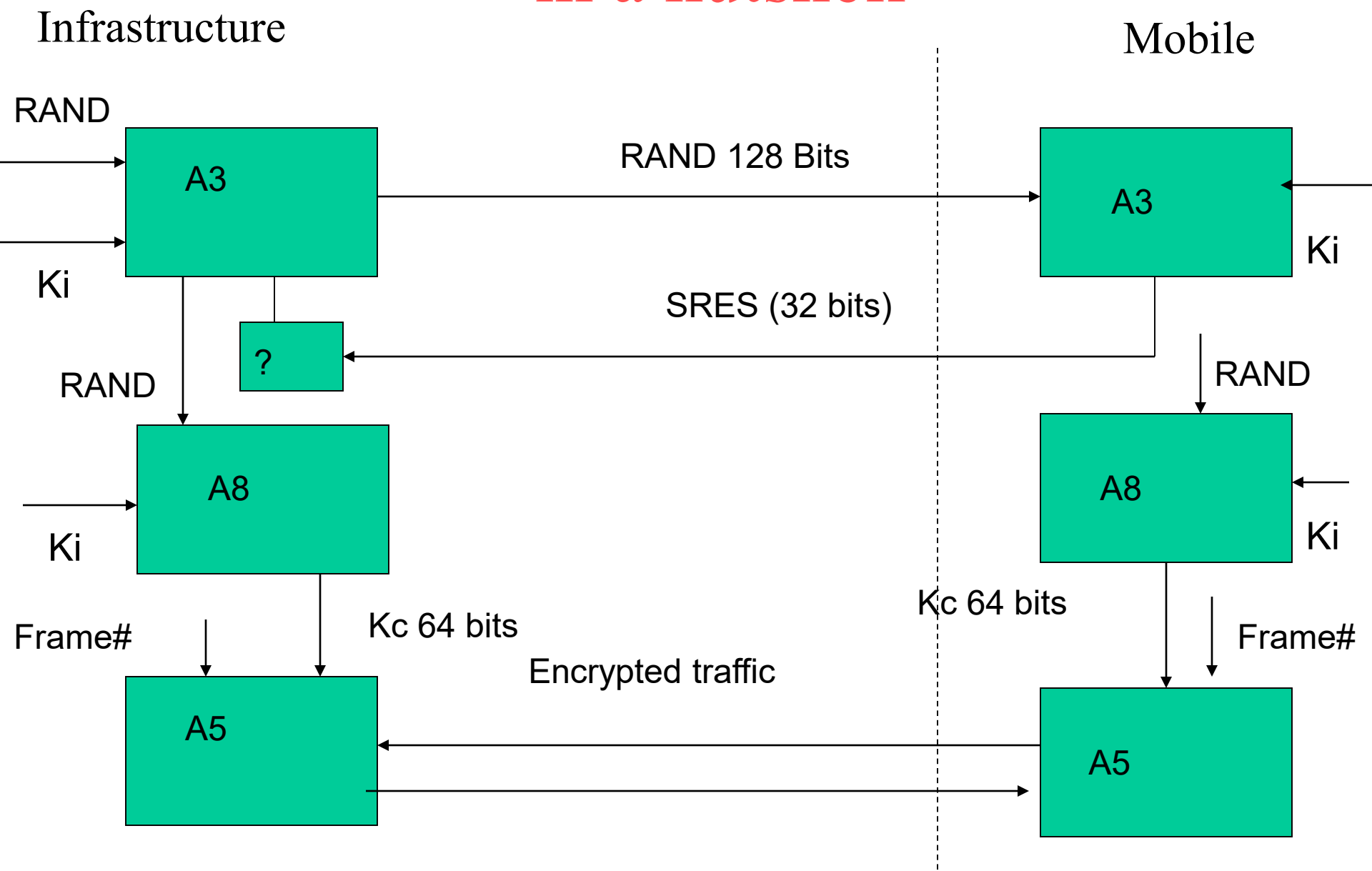
Introduction to Cryptographic Algorithms and Implementation

P.V. Ananda Mohan
Life Fellow IEEE

18-19 November 2019.

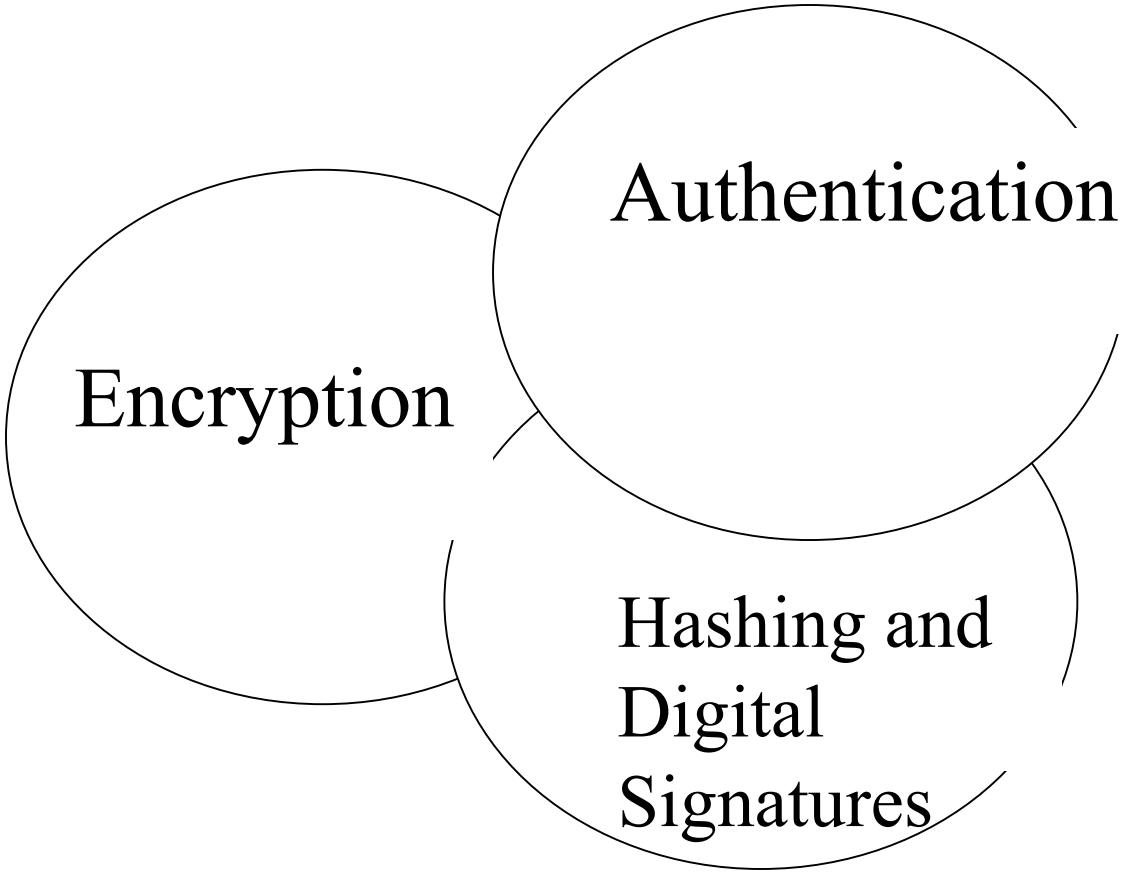
GSM Authentication and encryption in a nutshell



GSM Authentication and Encryption

- Challenge response protocol
- Network sends RAND (128 bits)
- K_i is secret key
- K_i, RAND used with Algorithm A3 to produce SRES (32 bits)
- K_i cannot be accessed by the user.
- Ues Hashing algorithm COMP128
- Uses Stream cipher A5/1
- Other versions A5/2, A5/3 (now),GMR for

Three Related domains



Encryption techniques

- ANALOG ENCRYPTION for Speech signals, video
- DIGITAL ENCRYPTION for Coded speech, coded Picture, Coded Video, Multimedia, Data files, E-commerce, Mobile communication, Wireless systems including GSM, CDMA phones, hot spots (802.11 Wireless networks) and in everything in future!!!

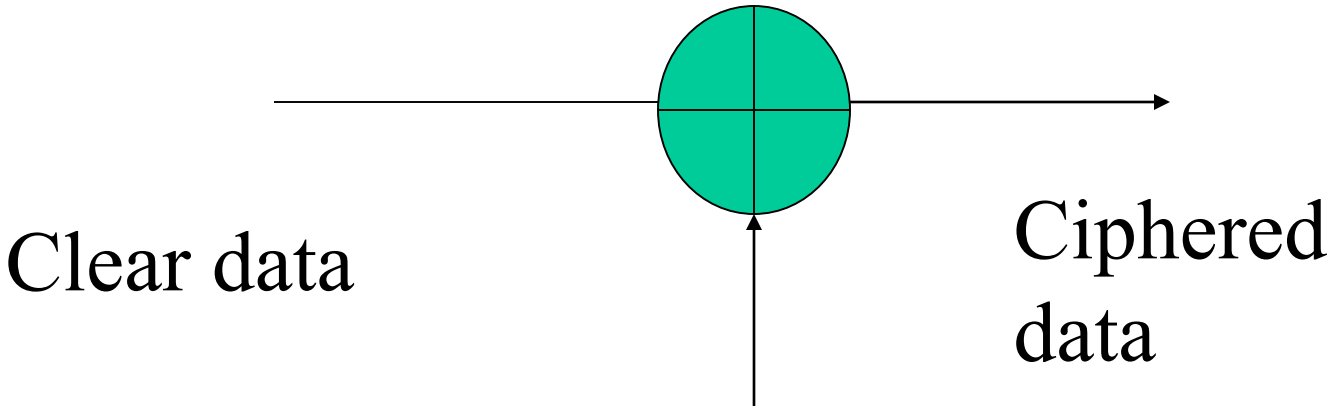
Digital encryption

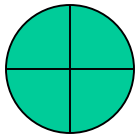
- Advantages:
- Efficient utilization of key space
- Error correction on medium to yield robust performance
- Variety of techniques possible
- Can take advantage of the characteristics of the signal e.g. Speech, fax,video, English text etc

Digital encryption

- Two techniques:
- Stream ciphering
- Block ciphering

Stream ciphering



 = Masking = modulo 2

SEQUENCE GENERATOR

Stream ciphers

Stream ciphers

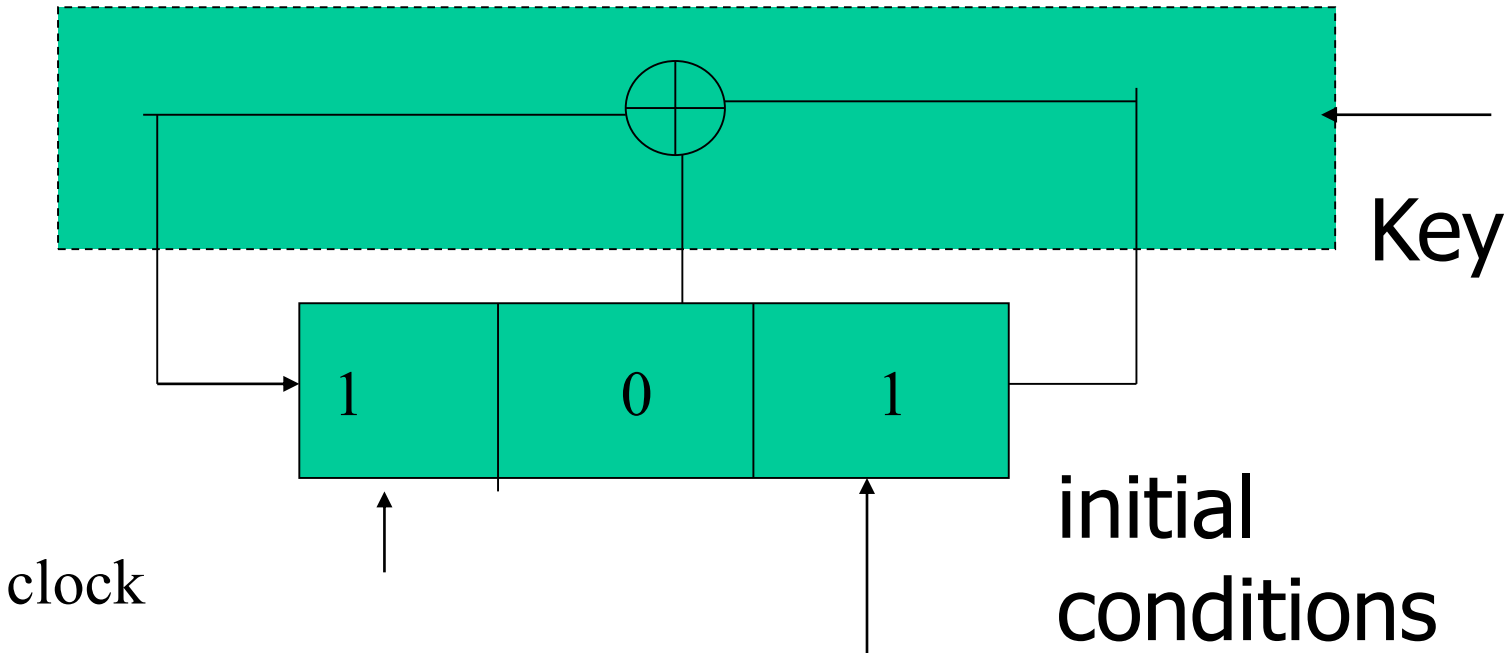
- No error propagation
- Bit by bit exclusive or (modulo 2) addition
- Variety of techniques available
- Possible to estimate the strength i.e the cryptanalytic effort needed to completely model the system

Stream ciphers

- Linear
- Non-linear
- Linear type uses linear feedback shift registers (LFSR)
- Nonlinear type can be based on LFSR and additional hardware

3-stage LFSR

- Primitive Polynomial is x^3+x^2+1



3-stage LFSR

- 101
- 110
- 111
- 011
- 001
- 100
- 010
-

- seed (initial condition)
- period= $2^3-1=7$ states

Initil condition shall be a non-zero value

LFSR

- Primitive Polynomial x^3+x+1 generates the reverse sequence.
- Key may for example select the Primitive polynomial and initial conditions.
- Several primitive polynomials available for each length LFSR. $\Phi(2^n-1)/n$ where Φ stands for *Euler's totient function*. E.g n = 5, $\Phi(31)/5 = 30/5 = 6$.
- $(1+x^2+x^5)$, $(1+x^2+x^3+x^4+x^5)$,$(1+x+x^2+x^4+x^5)$
- Good Statistics i.e.Distribution of '1's and '0' s
- See Menezes et al Bruce Schneier for a small list of trinomials

Testing for primitivity of P(x)

- P(x) of order 6 say
- It shall exactly divide $x^{63}+1$. ($63 = 2^6-1$)
- It should not exactly divide any x^a+1 where a is a factor of 63 or multiple of factors of 63. i.e, it should not divide exactly divide x^7+1, $x^{21}+1$, x^9+1

Exercise

- Is $x^{24}+x^{10}+1$ primitive?
- Is $x^{13}+x^{10}+x^{7}+1$ primitive?

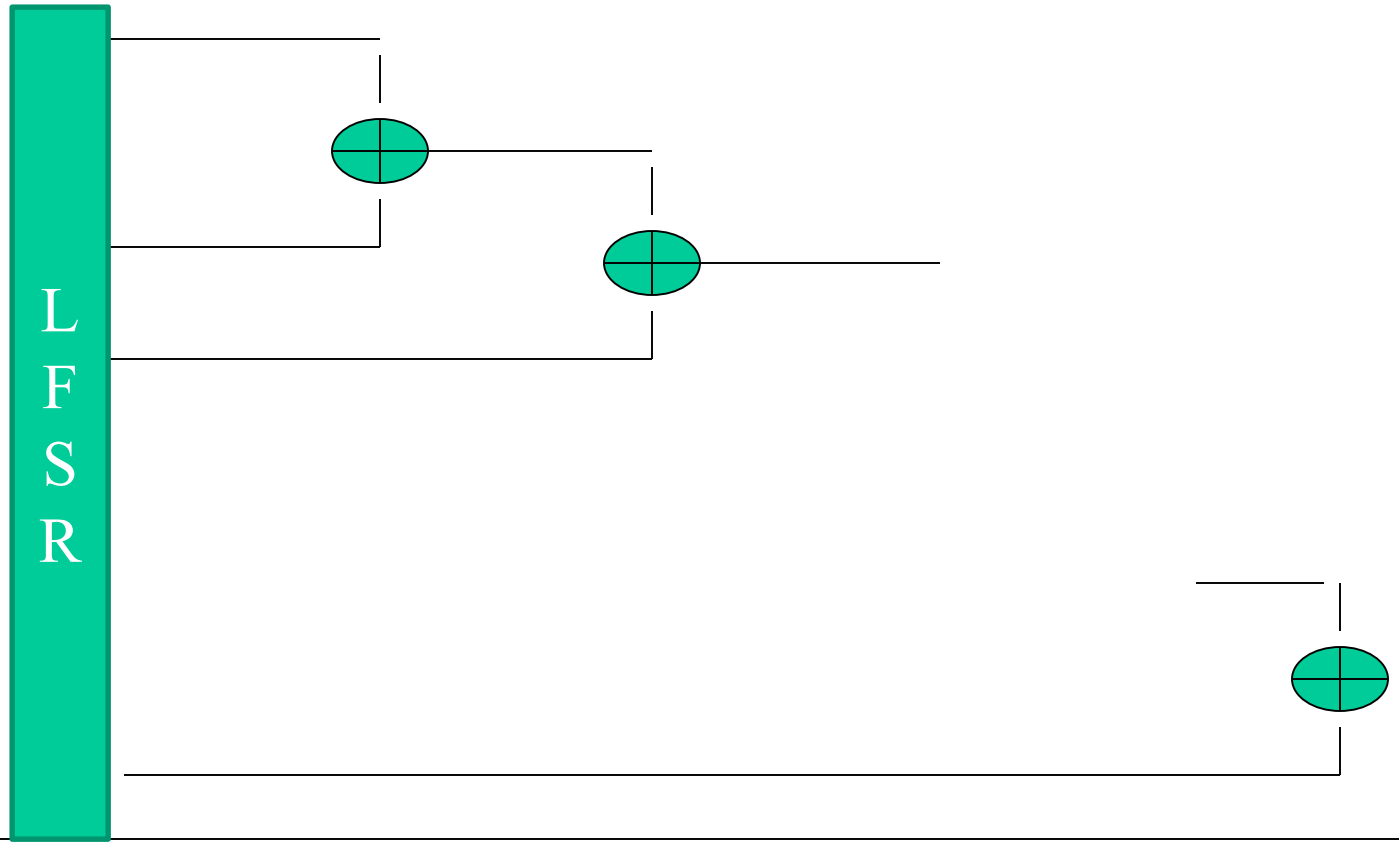
LFSR

- Can be broken if $2n$ consecutive bits are known for a n bit LFSR.
- Solve n linear equations (remember + is not plus!!!)
- $O(n^2)$ operations
- ***Linear equivalence*** or ***Linear Complexity*** is thus defined as the number of bits ($2n$) needed to completely solve a (n stage) LFSR.

LFSR

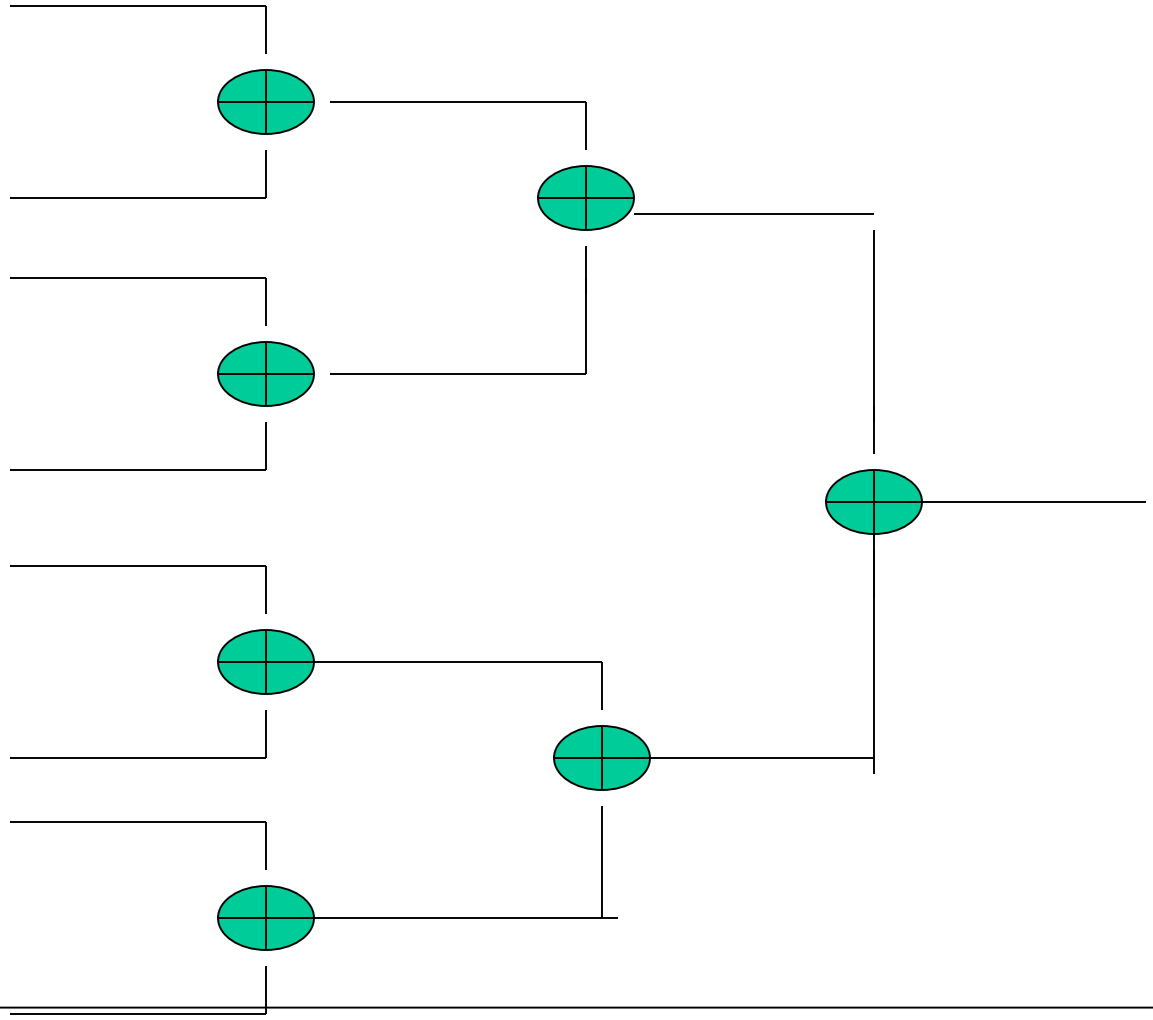
- Special purpose computer using modulo 2 bitwise computations
- Initial conditions have little security value since any two consecutive *2n* states can help us solve the system even though the period is 2^n-1.
- Berlekamp –Massey Algorithm
- Do not rely too much on Initial values as Key!!! But all commercial systems precisely do that.

Do not Do



- Delay is (n-1) Exclusive ORs

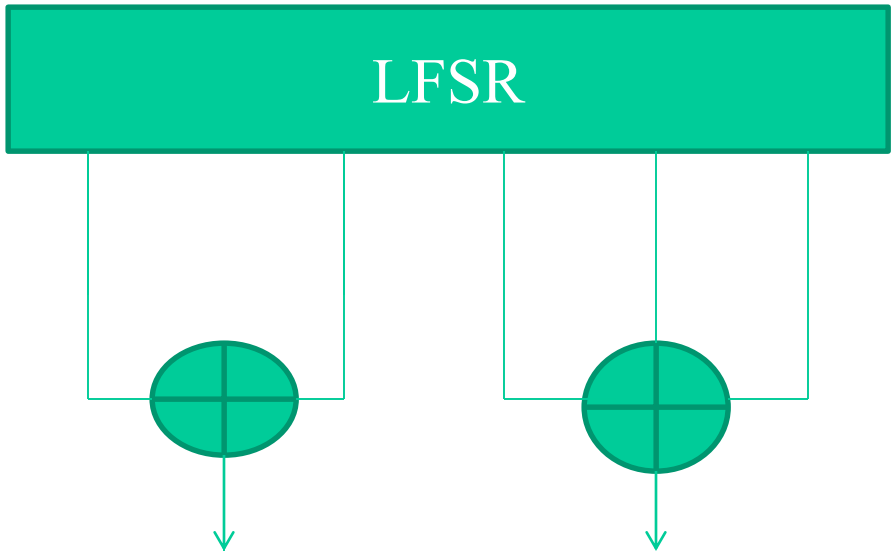
Do this



- Delay is $\log_2$ (n) Ex-Or delay, area is same.

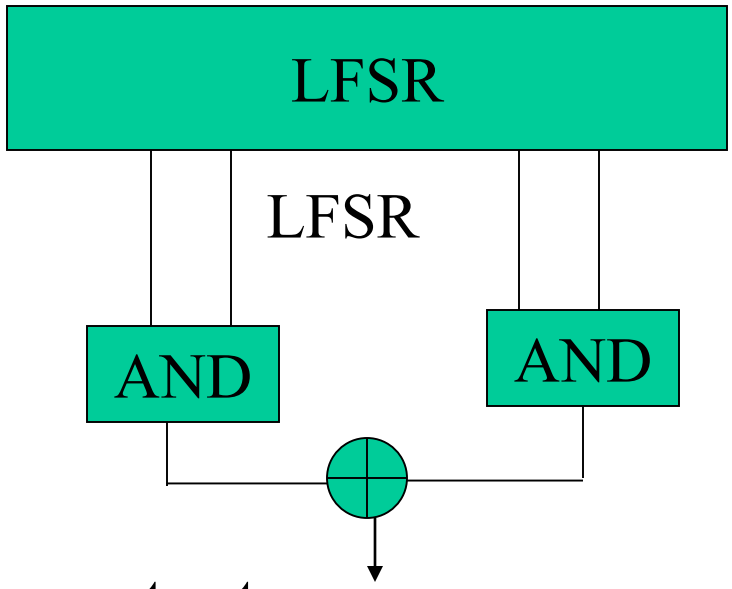
Generating delayed sequences

- EXOR outputs of two or three or more taps to get delayed sequences.



Nonlinear sequence generation

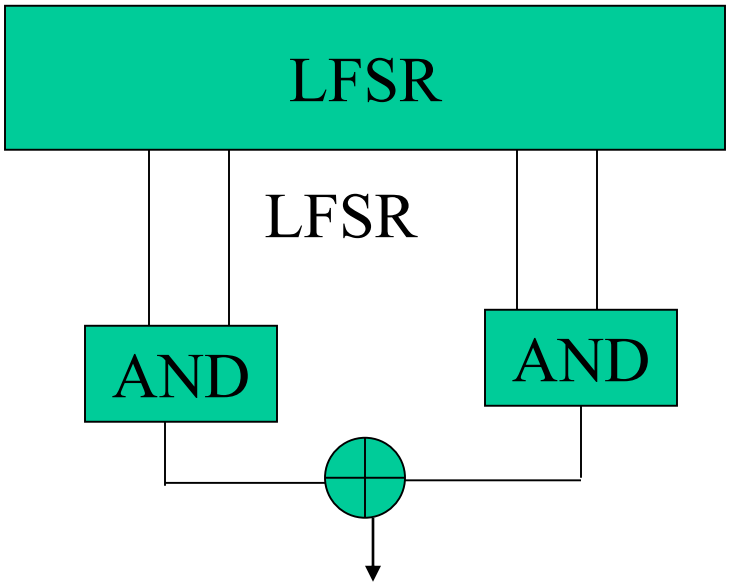
- Method1: Use LFSRs and Feed-forward logic (AND gates) on mutually exclusive outputs and excusive-or all of them



Nonlinear sequence generation

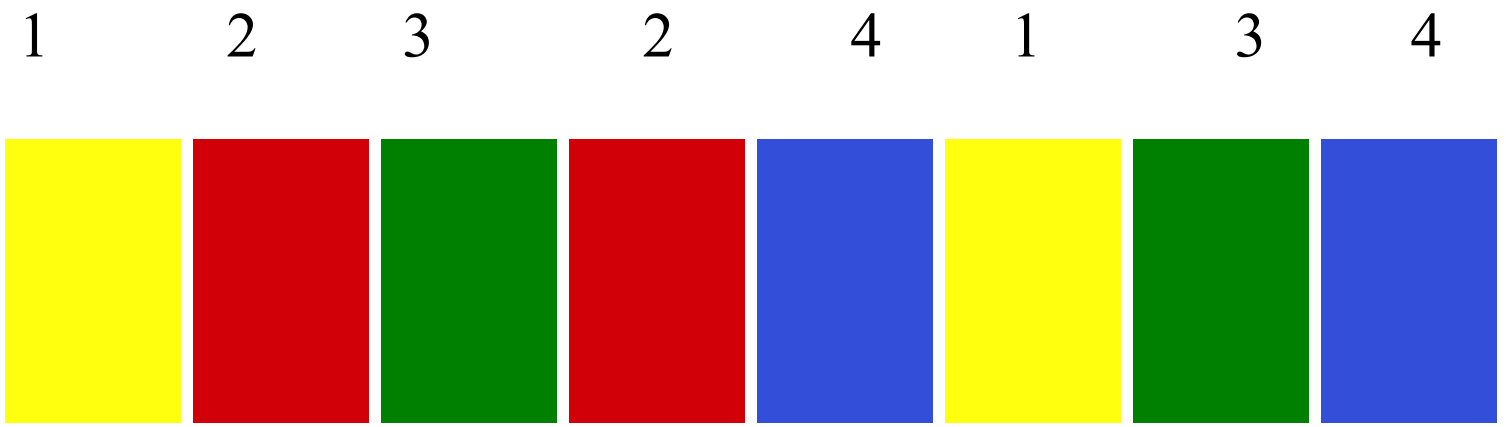
- AND gates introduce non-linearity
- Probability of zeroes more than one at the output - a price paid for non-linearity!
- Linear Equivalent is enhanced
- For a *n* stage LFSR, using one two-input AND gates linear equivalent of *n(n+1)/2* can be achieved e.g. 32 bit *LFSR* can have a *LE* of 528!

Generating Nonlinear sequence using feedforward



- n even, use n/2 AND gates and connect to taps at prescribed distances. EXOR all. Use Langford arrangement

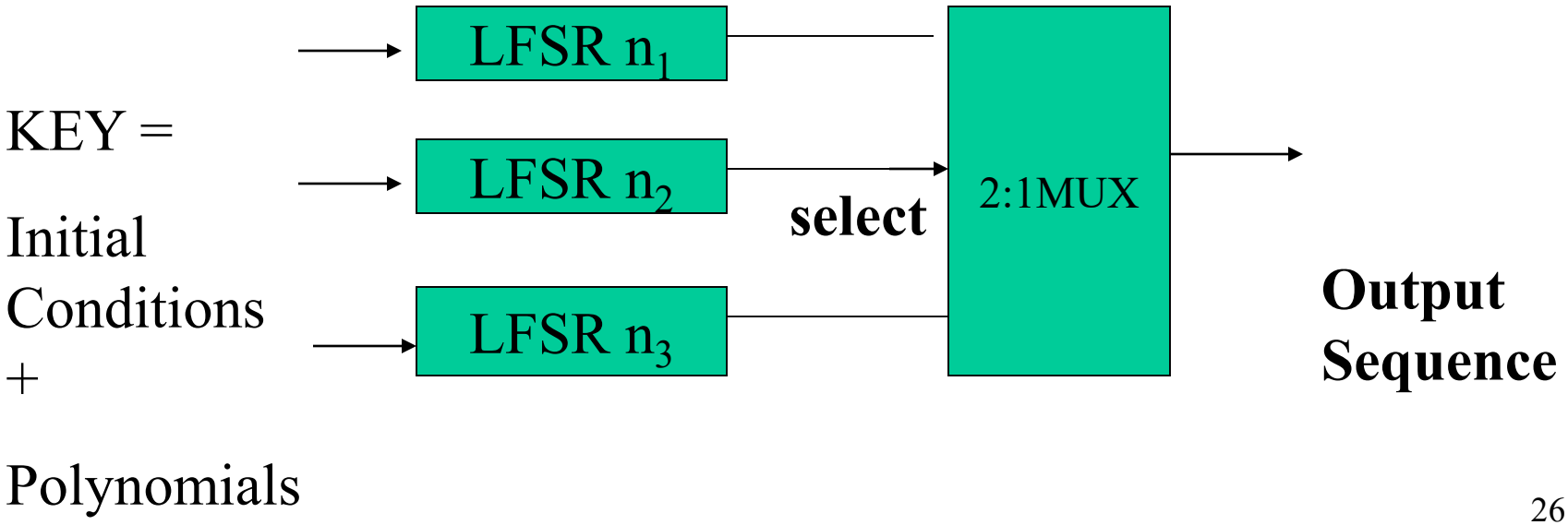
Example Langford arrangement for 8 bit LFSR



- Connect 2-input AND gates to both 1s, both 2s, both 3s and both 4s. EXOR all outputs of And gates. Vey Good Distribution (randomness).

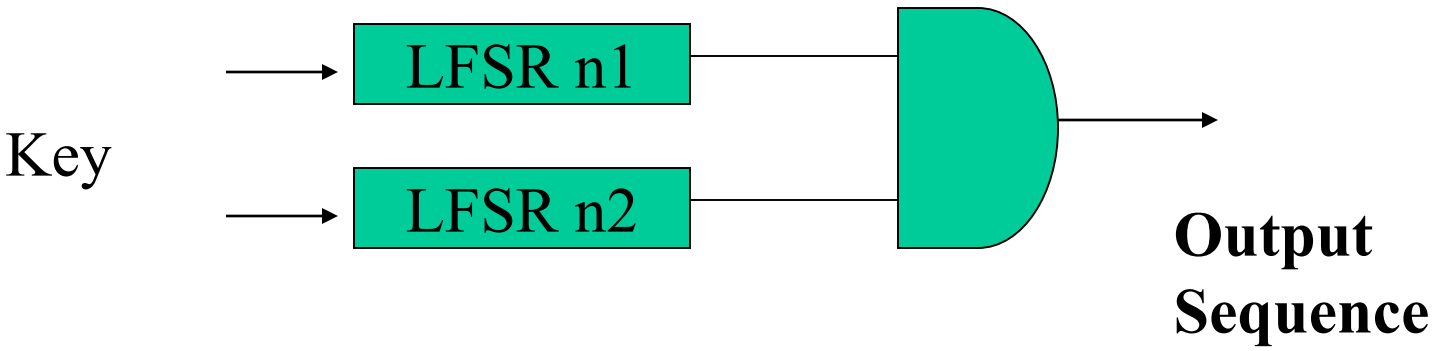
Nonlinear sequence generation

- Method II: Use Geffe Generator
- Uses Three LFSRs with mutually prime lengths
- LE is $n_2 n_1 + (n_2+1) n_3$
- Period is $(2^{n1}-1)(2^{n2}-1)(2^{n3}-1)$



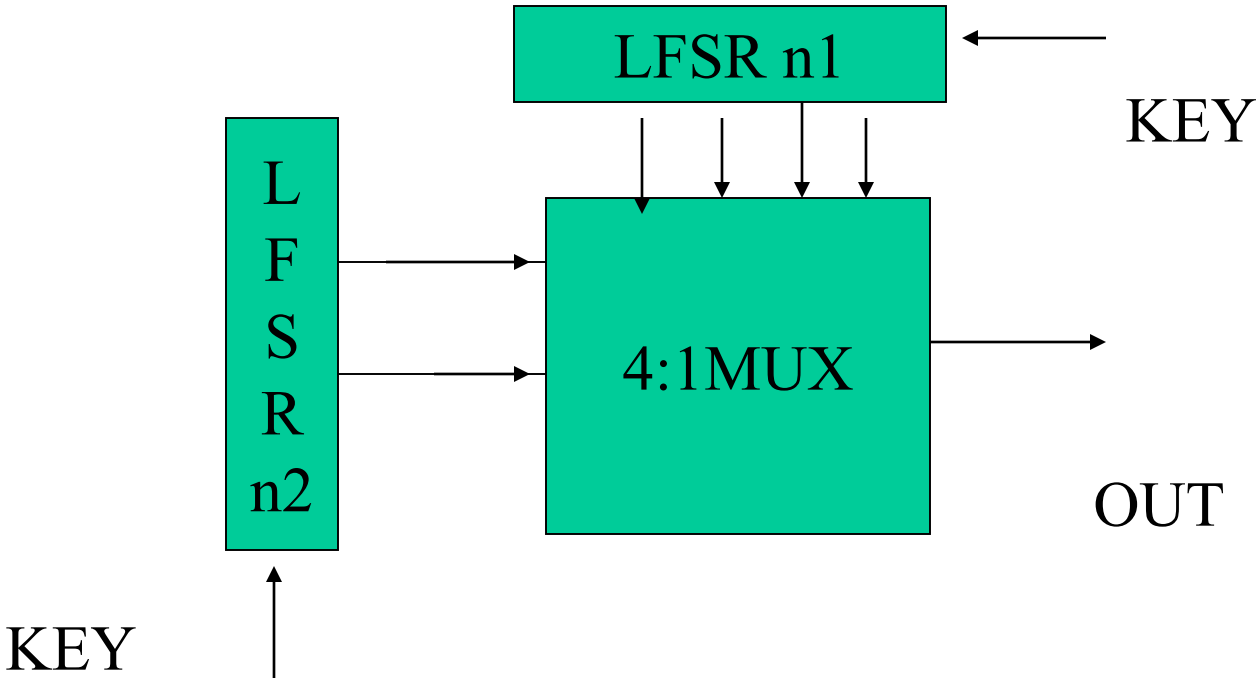
Nonlinear sequence generation

- Method III: Use two LFSRs and AND their outputs; A corollary of Geffe
- LE is $n_1 n_2$



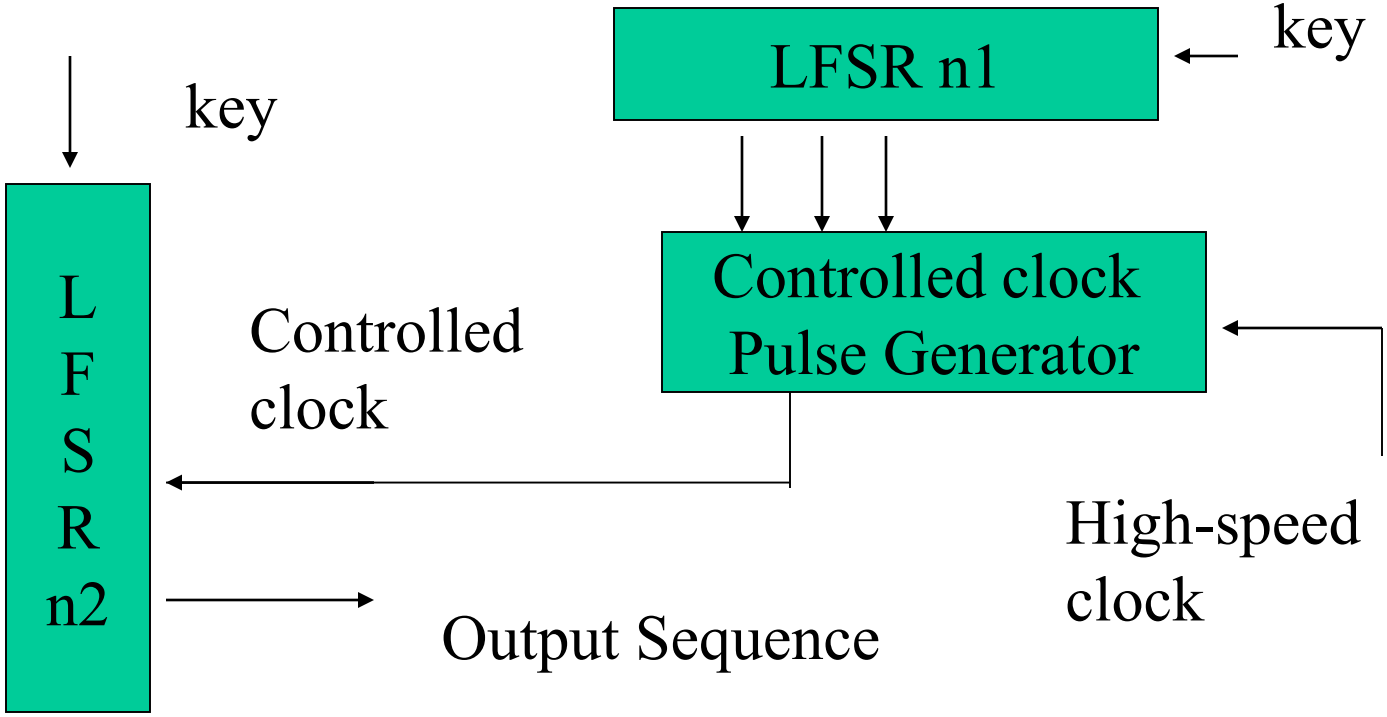
Nonlinear sequence generation

- Method IV: Multiplexer-based Sequence generator
- Uses Two LFSRs
- LE is very high = $n_1(1+\sum_{i=1}^{h} C_{il})$
- Period is $(2^{n1}-1).(2^{n2}-1)$



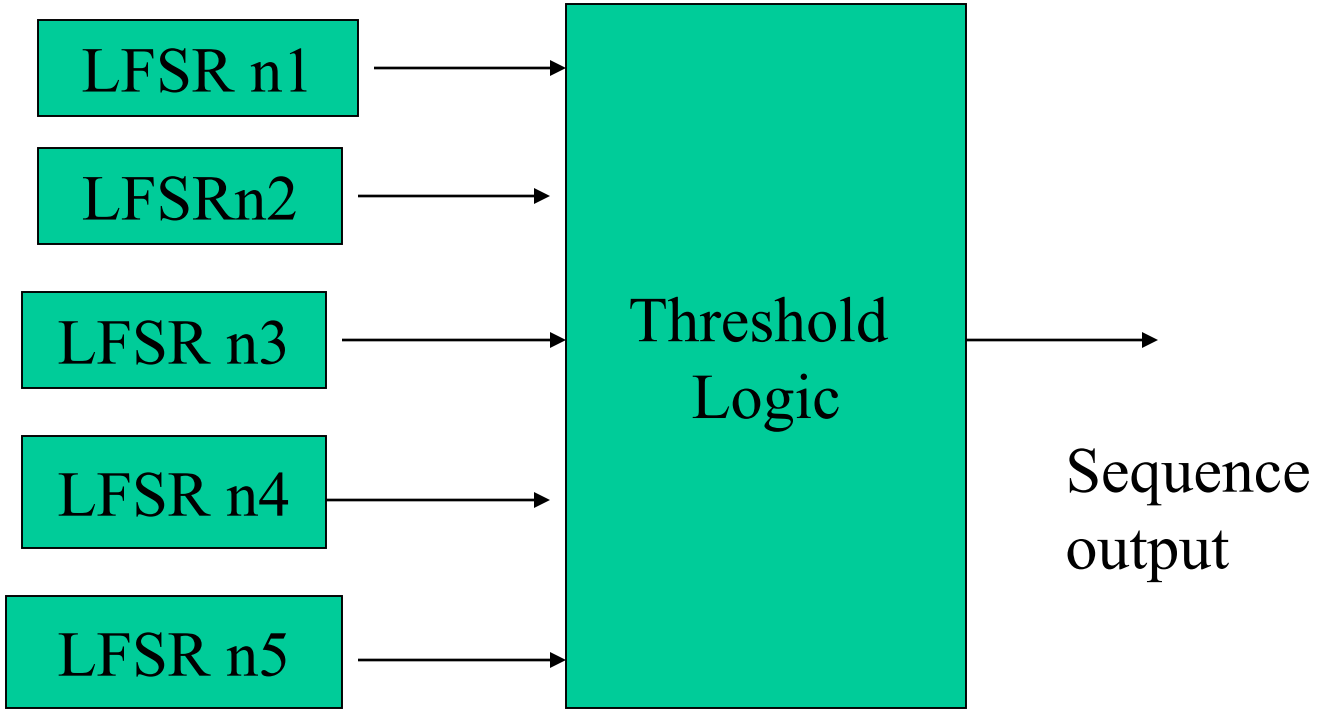
Nonlinear sequence generation

- Method V: CCSR (Clock controlled Shift Register)
- Use two LFSRs, period $(2^{n1}-1)(2^{n2}-1)$
- LE is extremely large $n_1.(2^{n2}-1)!$



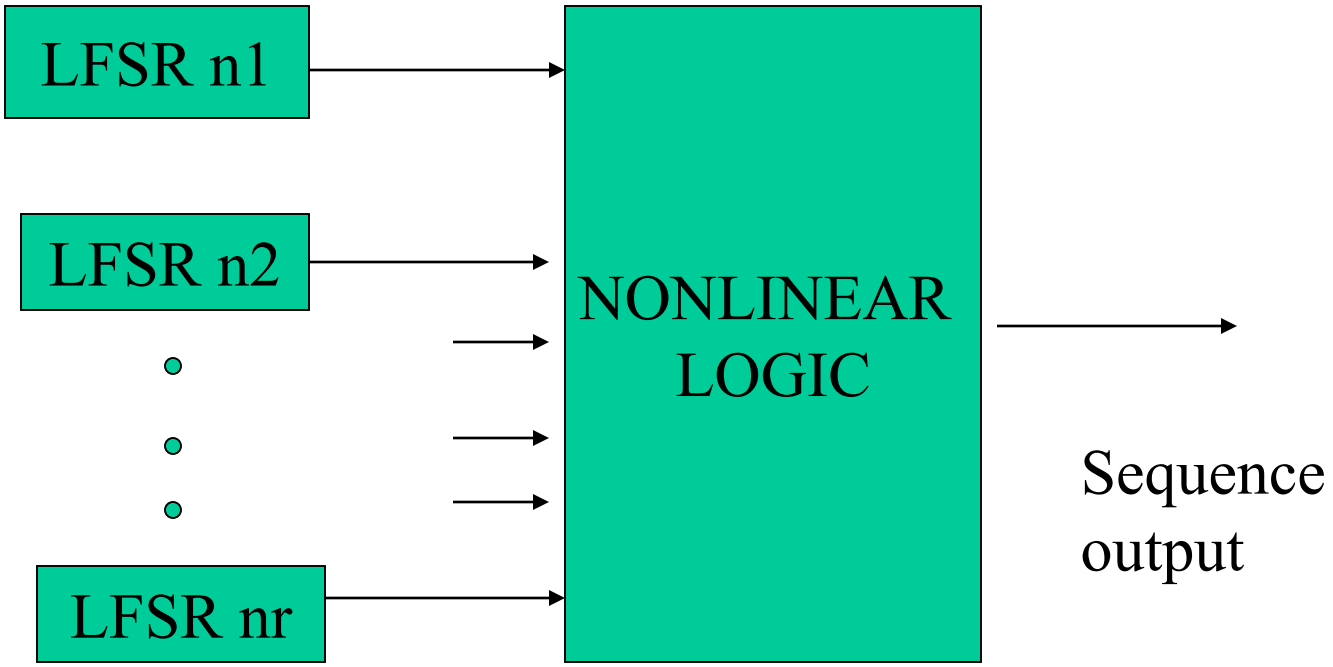
Nonlinear sequence generation

- Method VI: Threshold Generator
- LE is very large
- Period is product of all periods.
- Threshold logic outputs 1 if more than half the inputs are 1.



Nonlinear sequence generation

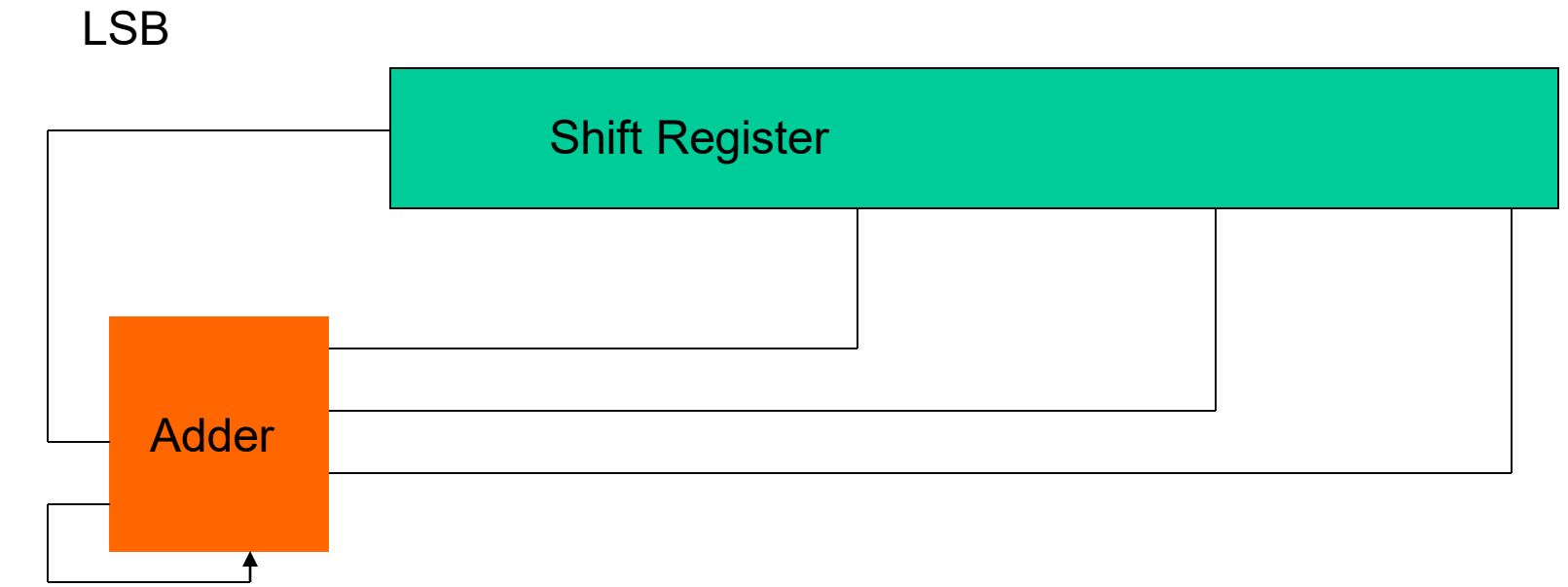
- Method VII: Use Correlation immune combining functions (Siegenthaler)



Linear complexity of Combiners

- If combining function is given
- X = x1+x2x3+x1x5x6+..etc
- LC is obtained by substituting the lengths of shift registers in place of variable.
- Caution: Leakage of low-order terms to output/ correlation with x1. Avoid first order terms. You can add 1 also. It inverts the sequence obtianed from combining other terms..

FCSR implementations



- **Feedback with carry shift register (FCSR)**
- **Relatively recent**
- **Taps chosen depend on the prime chosen**
- **Load with some initial value in SUM register of adder**
- **Connection integer is $2q_1+2^2q_2+\ldots2^nq_n-1$**
- **Example if q = 11, connection integer of a 3bit SR is 011 = 0+4+8-1 = 11**

Choice of p

- Only certain primes p give period of (p-1)
- Insight: we are finding for varying x, $(1/2^x) \bmod p$ and picking the LSB
- e.g choose p = 13
- Find (1/2) mod 13 =7, (7/2) mod 13 = 10, 5, 9,11,12,6,3,8,4,2
- Choose LSBs to obtain the sequence 110111001000
- Suitable for FCSR since 12 bit sequence is obtained.
- You can perform marriage of LFSRs and FCSRs!!!
- See Bruce Schneier for list of primes suitable for FCSR upto 10,000

Exercise

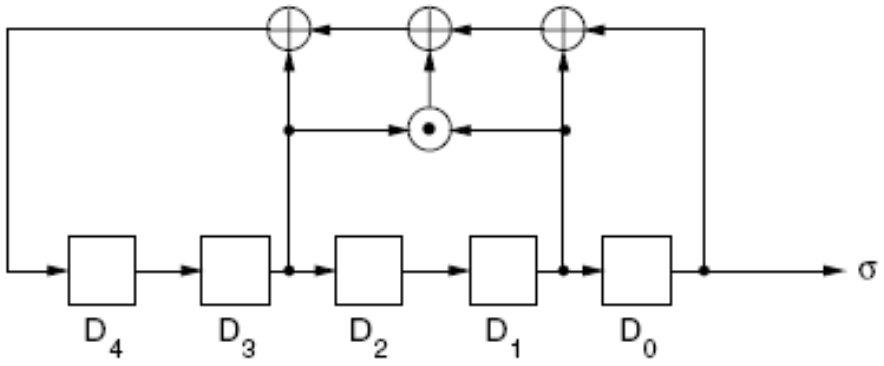
- Is $p = 89$ fit for FCSR?

NFSR

Example 1. Consider the binary 5-stage NLFSR with feedback function

$$F(x_0, x_1, x_2, x_3, x_4) = x_0 + x_1 + x_3 + x_1x_3.$$

The feedback shift register is shown in the following figure:



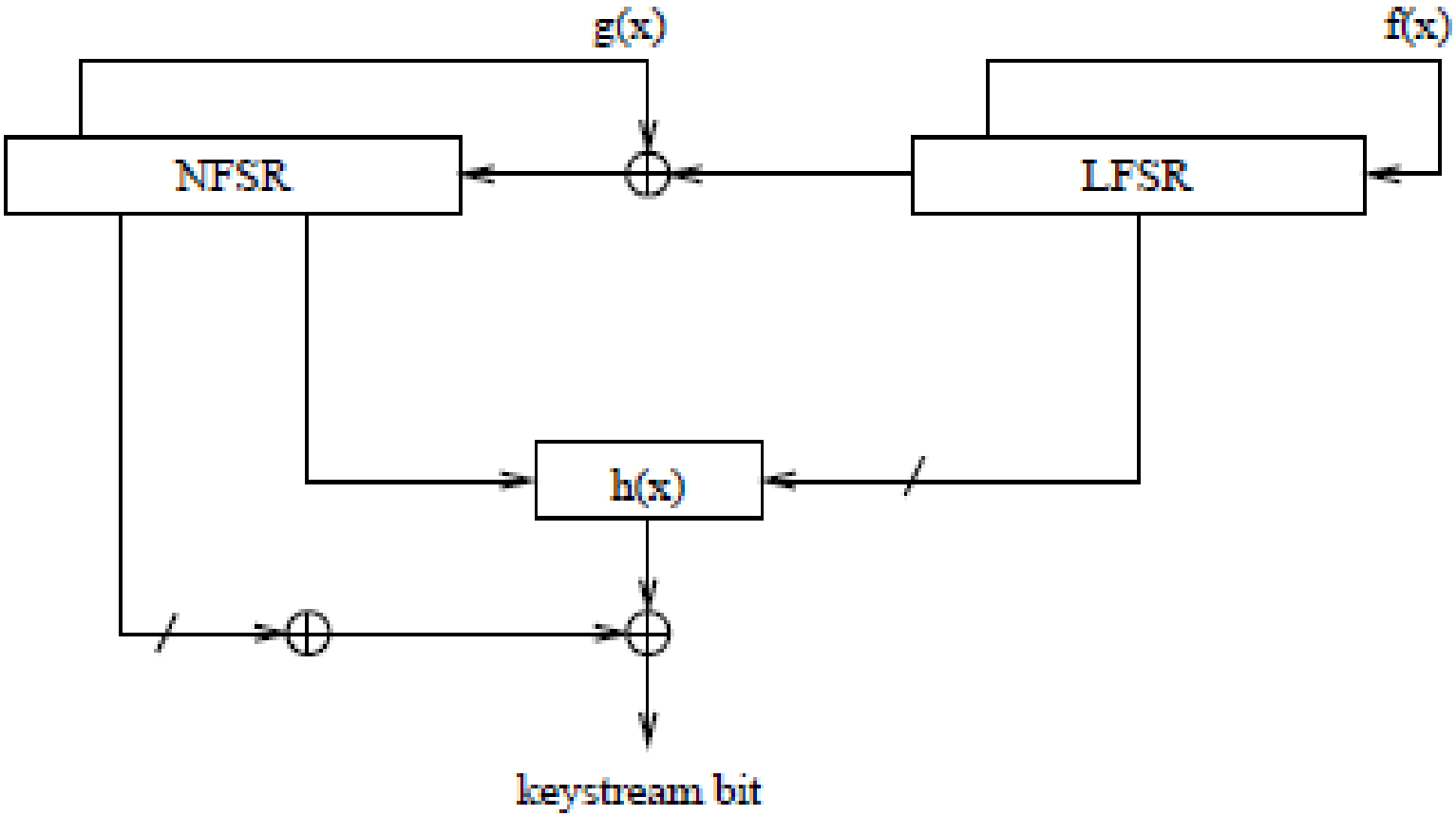
The standard output sequence corresponding to the initial state vector $s_0 = (0, 0, 0, 0, 1)$ for the given feedback shift register is

$$\sigma = (0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1)^{\infty},$$

- Nonlinear feedback Shift register

e-stream ciphers

Grain



LFSR primitive polynomial

$$f_0(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}.$$

$$g_0(x) = 1 + x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}.$$

$$g_1(x) = 1 + x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{66} + x^{71} + x^{80} + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}.$$

- Polynomials of Non-linear Feedback Shift registers

Cipher Output bit for V0

$$z_t^0 = b_t + h_0(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}),$$

where the filter boolean function $h_0(x_0, \cdots, x_4)$ is defined as

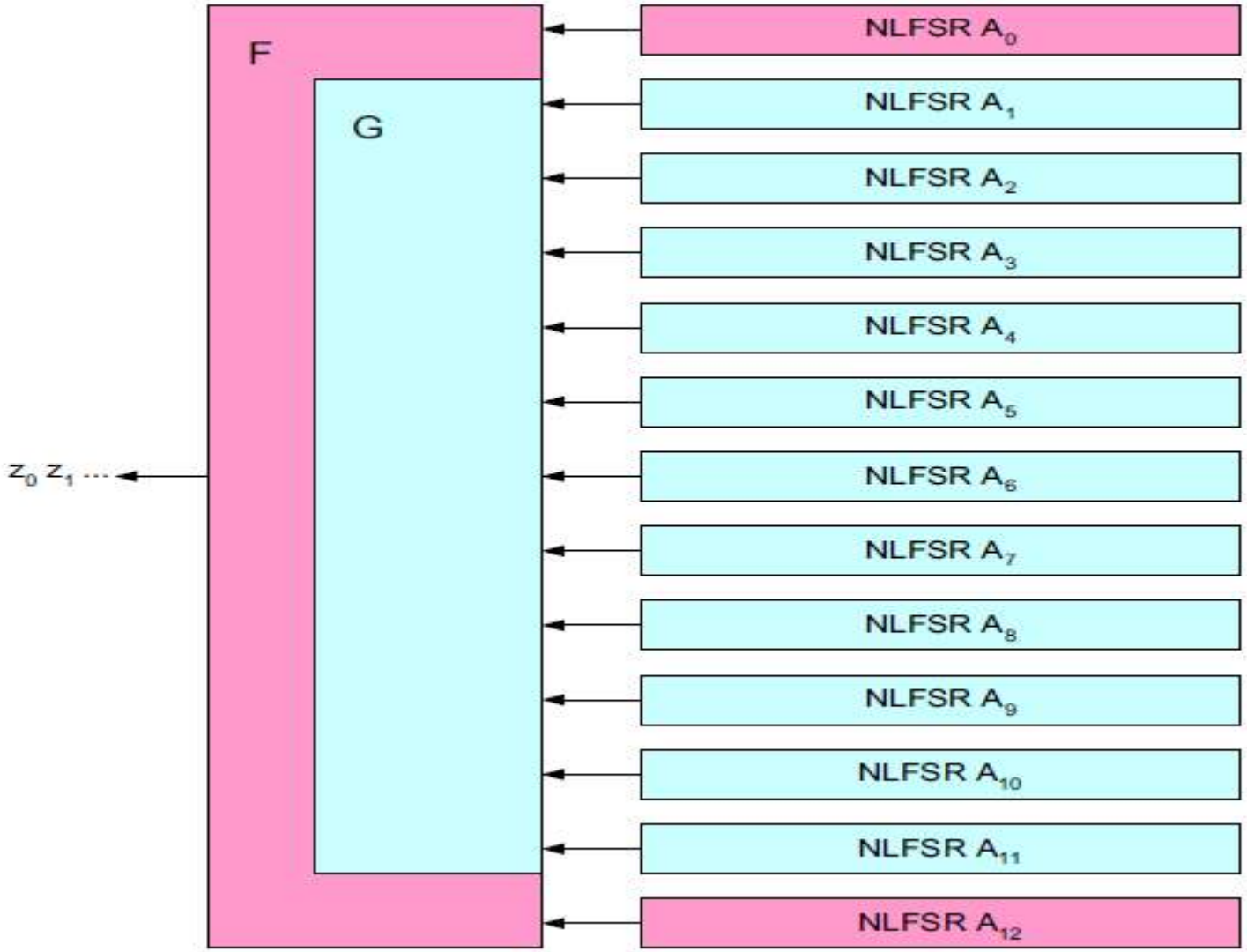
$$h_0(x_0, \cdots, x_4) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4.$$

Cipher Output bit for V1

$$z_t^1 = \sum_{i \in \mathcal{A}_1} b_{t+i} + h_1(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}),$$

where $\mathcal{A}_1 = \{1, 2, 4, 10, 31, 43, 56\}$.

Achterbahn 128/80



$$
\begin{aligned}
F(x_0, x_1, \ldots, x_{12}) = {} & x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_{11} + x_{12} + x_0x_5 \\
& + x_2x_{10} + x_2x_{11} + x_4x_8 + x_4x_{12} + x_5x_6 + x_6x_8 + x_6x_{10} + x_6x_{11} \\
& + x_6x_{12} + x_7x_8 + x_7x_{12} + x_8x_9 + x_8x_{10} + x_9x_{10} + x_9x_{11} + x_9x_{12} \\
& + x_{10}x_{12} + x_0x_5x_8 + x_0x_5x_{10} + x_0x_5x_{11} + x_0x_5x_{12} + x_1x_2x_8 \\
& + x_1x_2x_{12} + x_1x_4x_{10} + x_1x_4x_{11} + x_1x_8x_9 + x_1x_9x_{10} + x_1x_9x_{11} \\
& + x_1x_9x_{12} + x_2x_3x_8 + x_2x_3x_{12} + x_2x_4x_8 + x_2x_4x_{10} + x_2x_4x_{11} \\
& + x_2x_4x_{12} + x_2x_7x_8 + x_2x_7x_{12} + x_2x_8x_{10} + x_2x_8x_{11} + x_2x_9x_{10} \\
& + x_2x_9x_{11} + x_2x_{10}x_{12} + x_2x_{11}x_{12} + x_3x_4x_8 + x_3x_4x_{12} + x_3x_8x_9 \\
& + x_3x_9x_{12} + x_4x_7x_8 + x_4x_7x_{12} + x_4x_8x_9 + x_4x_9x_{12} + x_5x_6x_8 \\
& + x_5x_6x_{10} + x_5x_6x_{11} + x_5x_6x_{12} + x_6x_8x_{10} + x_6x_8x_{11} + x_6x_{10}x_{12} \\
& + x_6x_{11}x_{12} + x_7x_8x_9 + x_7x_9x_{12} + x_8x_9x_{10} + x_8x_9x_{11} + x_9x_{10}x_{12} \\
& + x_9x_{11}x_{12} + x_0x_5x_8x_{10} + x_0x_5x_8x_{11} + x_0x_5x_{10}x_{12} + x_0x_5x_{11}x_{12} \\
& + x_1x_2x_3x_8 + x_1x_2x_3x_{12} + x_1x_2x_7x_8 + x_1x_2x_7x_{12} + x_1x_3x_5x_8 \\
& + x_1x_3x_5x_{12} + x_1x_3x_8x_9 + x_1x_3x_9x_{12} + x_1x_4x_8x_{10} + x_1x_4x_8x_{11} \\
& + x_1x_4x_{10}x_{12} + x_1x_4x_{11}x_{12} + x_1x_5x_7x_8 + x_1x_5x_7x_{12} + x_1x_7x_8x_9 \\
& + x_1x_7x_9x_{12} + x_1x_8x_9x_{10} + x_1x_8x_9x_{11} + x_1x_9x_{10}x_{12} + x_1x_9x_{11}x_{12} \\
& + x_2x_3x_4x_8 + x_2x_3x_4x_{12} + x_2x_3x_5x_8 + x_2x_3x_5x_{12} + x_2x_4x_7x_8 \\
& + x_2x_4x_7x_{12} + x_2x_4x_8x_{10} + x_2x_4x_8x_{11} + x_2x_4x_{10}x_{12} + x_2x_4x_{11}x_{12} \\
& + x_2x_5x_7x_8 + x_2x_5x_7x_{12} + x_2x_8x_9x_{10} + x_2x_8x_9x_{11} + x_2x_9x_{10}x_{12} \\
& + x_2x_9x_{11}x_{12} + x_3x_4x_8x_9 + x_3x_4x_9x_{12} + x_4x_7x_8x_9 + x_4x_7x_9x_{12} \\
& + x_5x_6x_8x_{10} + x_5x_6x_8x_{11} + x_5x_6x_{10}x_{12} + x_5x_6x_{11}x_{12}.
\end{aligned}
$$

- Combining Function

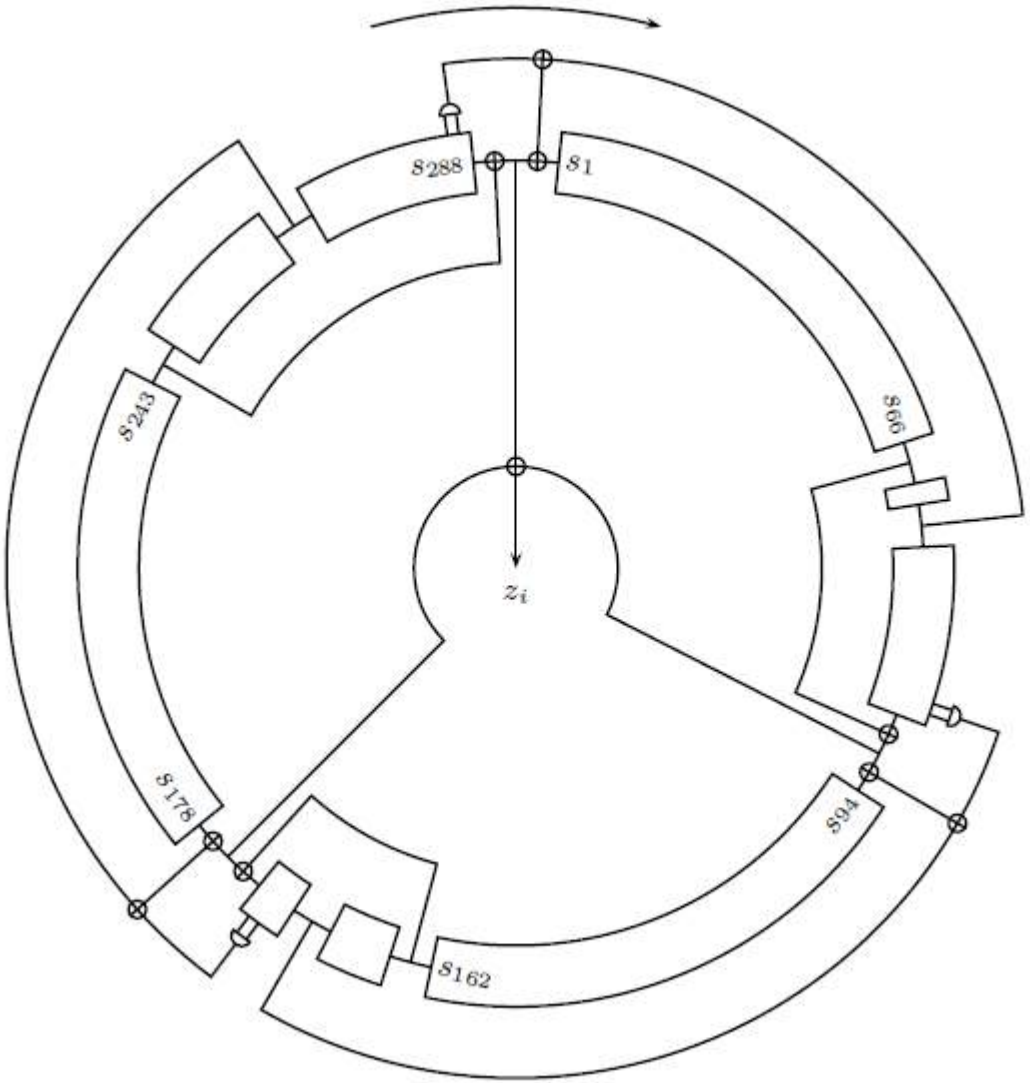
The combining function F has the following properties:

(i) F is balanced;
(ii) F has algebraic degree 4;
(iii) F is correlation immune of order 8;
(iv) F has nonlinearity 3584;
(v) F has algebraic immunity 4;

Combining Function of Achterbahn 80

$$
\begin{aligned}
G(x_1, x_2, \ldots, x_{11}) = {} & x_1 + x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_{11} + x_2x_{10} + x_2x_{11} \\
& + x_4x_8 + x_5x_6 + x_6x_8 + x_6x_{10} + x_6x_{11} + x_7x_8 + x_8x_9 + x_8x_{10} \\
& + x_9x_{10} + x_9x_{11} + x_1x_2x_8 + x_1x_4x_{10} + x_1x_4x_{11} + x_1x_8x_9 \\
& + x_1x_9x_{10} + x_1x_9x_{11} + x_2x_3x_8 + x_2x_4x_8 + x_2x_4x_{10} + x_2x_4x_{11} \\
& + x_2x_7x_8 + x_2x_8x_{10} + x_2x_8x_{11} + x_2x_9x_{10} + x_2x_9x_{11} + x_3x_4x_8 \\
& + x_3x_8x_9 + x_4x_7x_8 + x_4x_8x_9 + x_5x_6x_8 + x_5x_6x_{10} + x_5x_6x_{11} \\
& + x_6x_8x_{10} + x_6x_8x_{11} + x_7x_8x_9 + x_8x_9x_{10} + x_8x_9x_{11} + x_1x_2x_3x_8 \\
& + x_1x_2x_7x_8 + x_1x_3x_5x_8 + x_1x_3x_8x_9 + x_1x_4x_8x_{10} + x_1x_4x_8x_{11} \\
& + x_1x_5x_7x_8 + x_1x_7x_8x_9 + x_1x_8x_9x_{10} + x_1x_8x_9x_{11} + x_2x_3x_4x_8 \\
& + x_2x_3x_5x_8 + x_2x_4x_7x_8 + x_2x_4x_8x_{10} + x_2x_4x_8x_{11} + x_2x_5x_7x_8 \\
& + x_2x_8x_9x_{10} + x_2x_8x_9x_{11} + x_3x_4x_8x_9 + x_4x_7x_8x_9 + x_5x_6x_8x_{10} \\
& + x_5x_6x_8x_{11}.
\end{aligned}
$$

TRIVIUM



for $i = 1$ to N **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

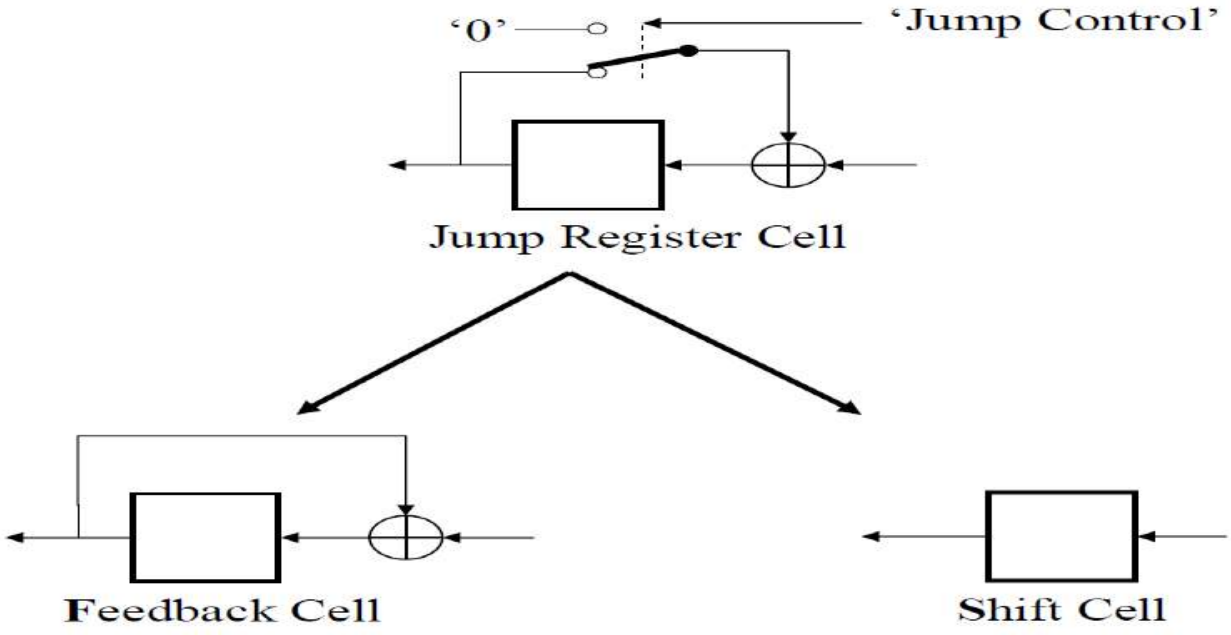
end for

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$$
$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$$
$$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$$
for $i = 1$ to $4 \cdot 288$ do

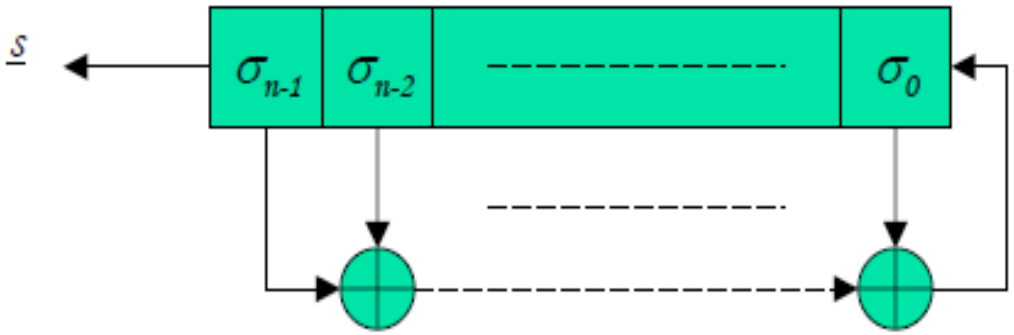
$$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$$
$$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$$
$$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$$
$$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$$
$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$$
$$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$$
end for

- Initialization

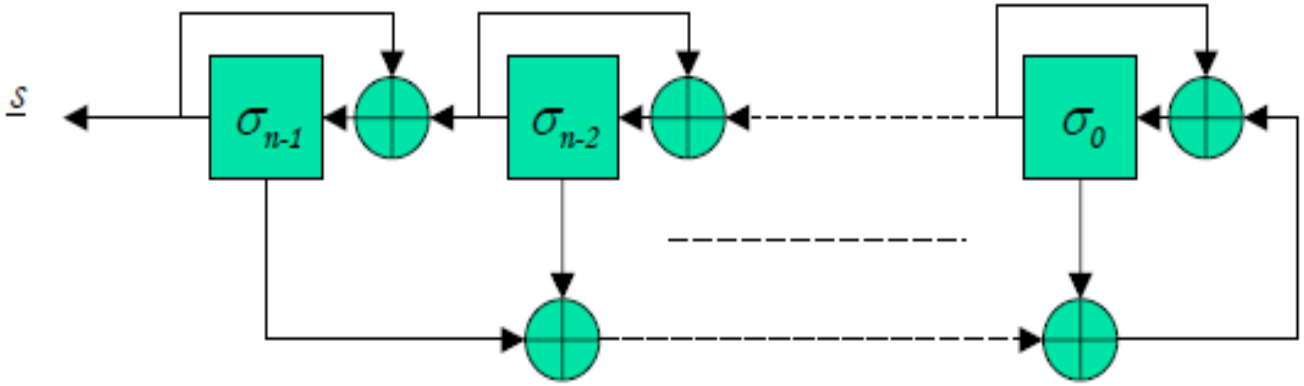
JUMP LFSR


'0'
'Jump Control'
Jump Register Cell
Feedback Cell
Shift Cell
$F(x) = \frac{x}{x+1}$
$S(x) = x$

Jump register

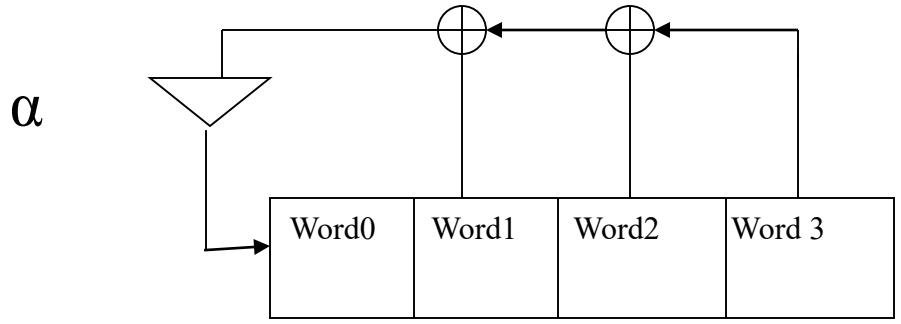


A



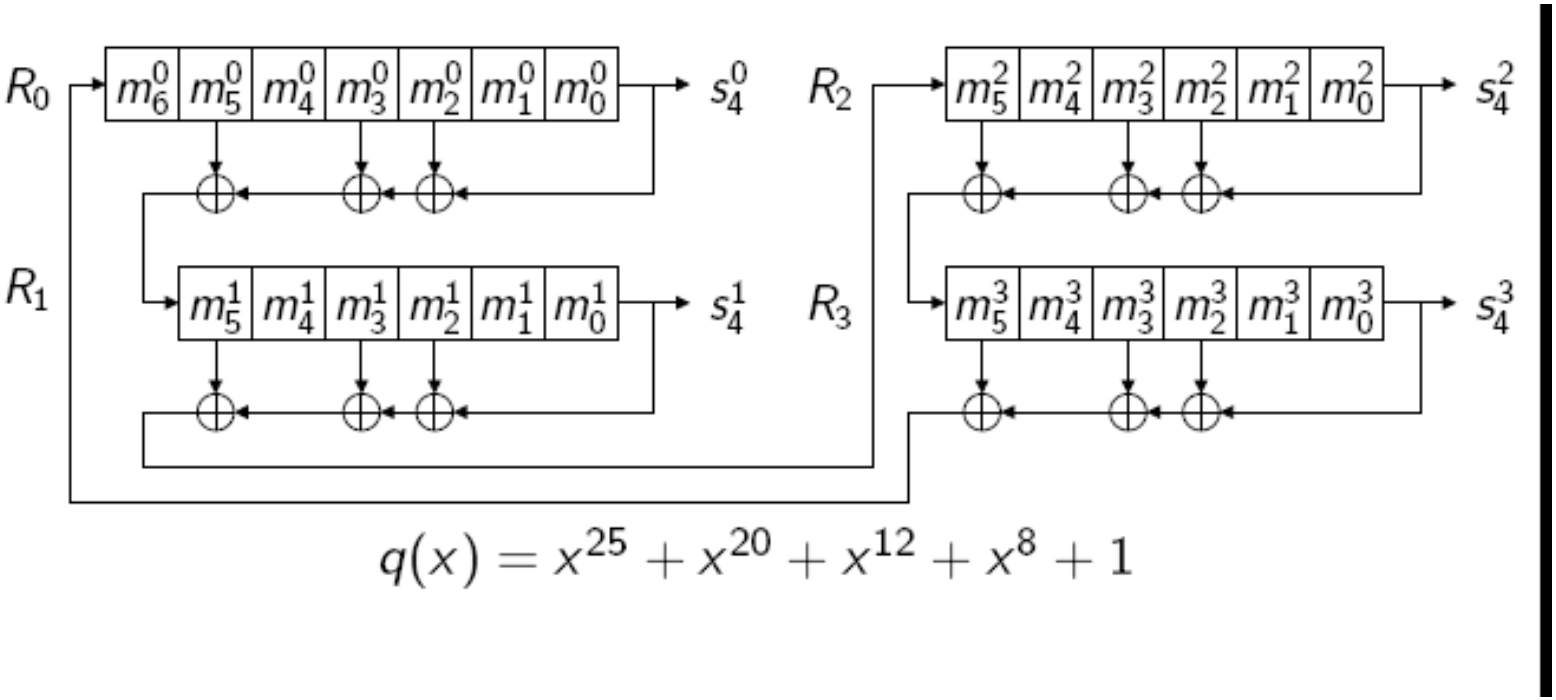
A+I

Word Based Shift Registers

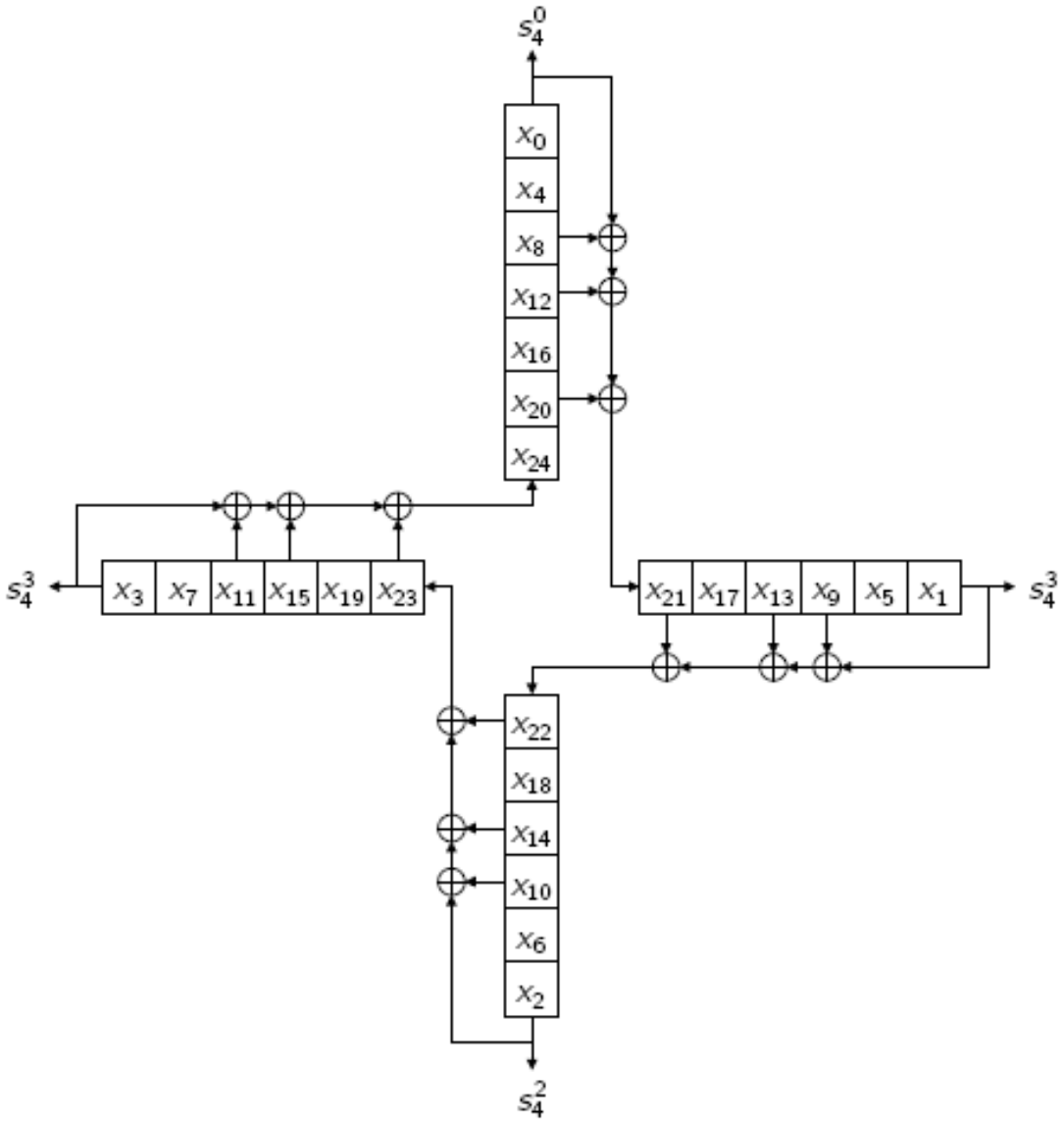


GF(2) poly x^2+x+1

- Testing for coefficients same as in the case of primitive polynomial testing.
- Used in SNOW and many other ciphers.



- Special LFSR arrangements



A. $f(x) = x^{25} + x^{20} + x^{12} + x^8 + 1$

- Windmill generator

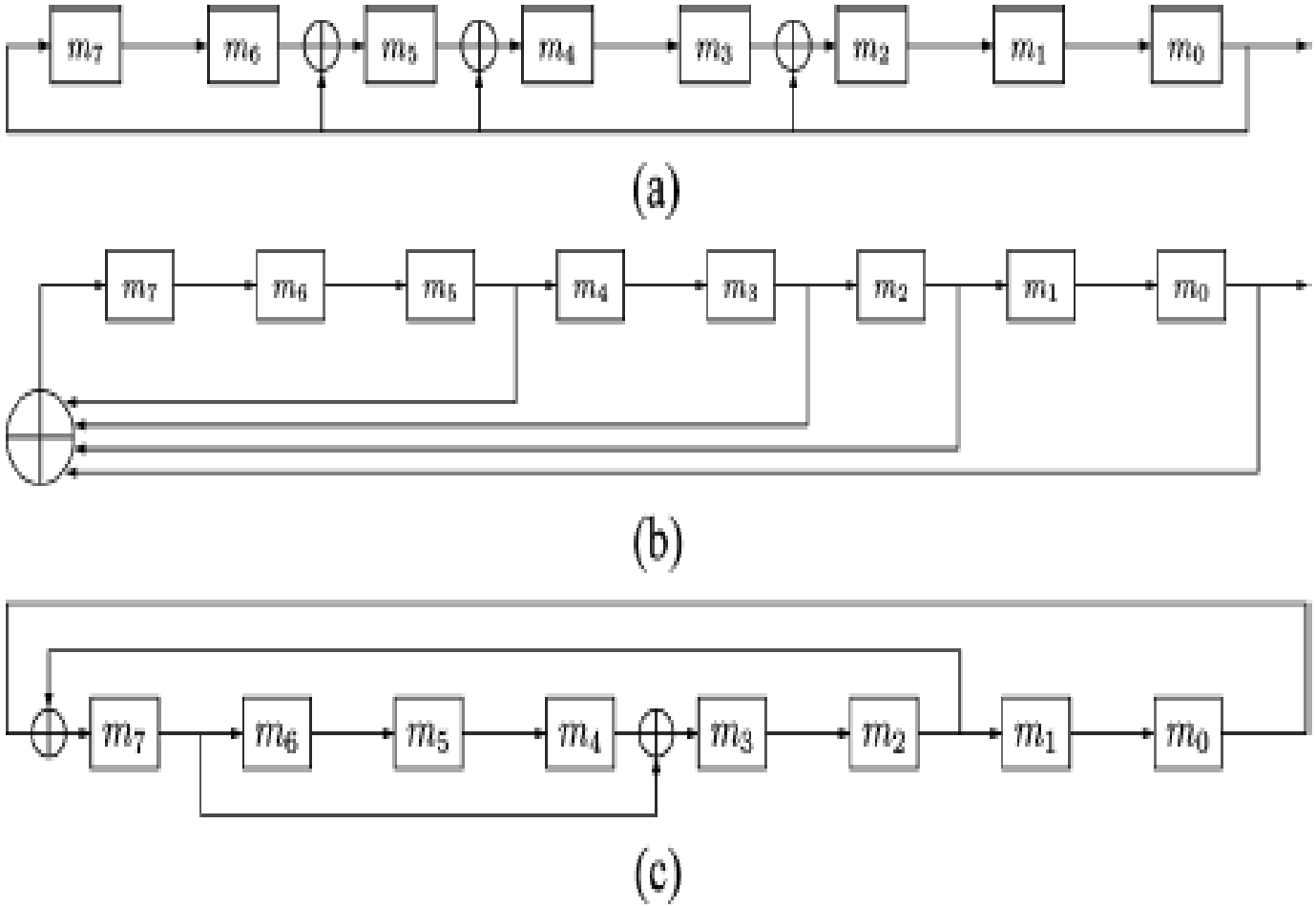


Fig. 5. Three LFSR with connection polynomial $Q(X) = X^8 + X^6 + X^5 + X^3 + 1$. (a) Galois LFSR $\mathcal{L}_0$. (b) Fibonacci LFSR $\mathcal{L}_1$. (c) Ring LFSR $\mathcal{L}_2$.

Diffusion delay

Definition:
Number of clock cycles needed to totally change the content of the shift register

Reduces synchronization time.

Marriage of Various building blocks

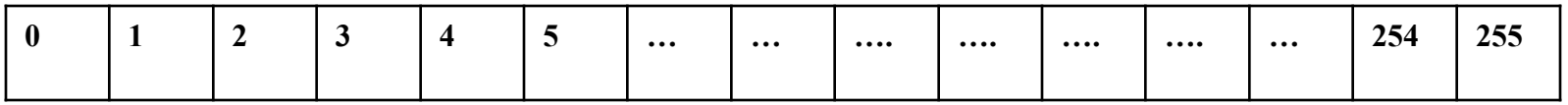
- LFSR
- NFSR (non-linear feedback Shift register)
- FCSR

About RC4 (Rivest Cipher)

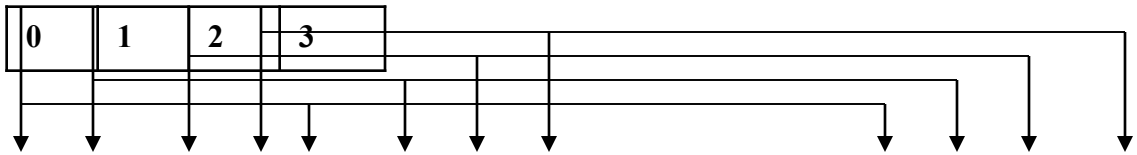
- Variable length key 1 byte to 256 bytes.
- An initial state vector S(i) having numbers 0,1, 2,….254, 255
- Temporary vector T also of 256 bytes
- Key K is used to fill up T totally by repeating K if needed
- Then Initial permutation of S is derived.
- Then one after another 256 bytes are generated which will be used to mask the clear text bytes.

RC4

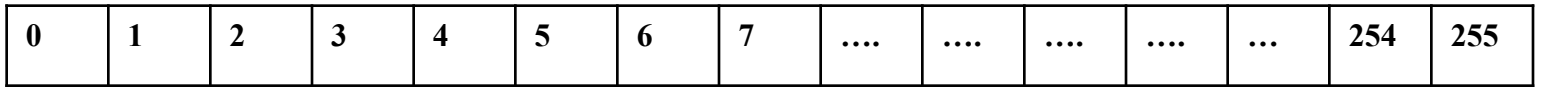
S



K



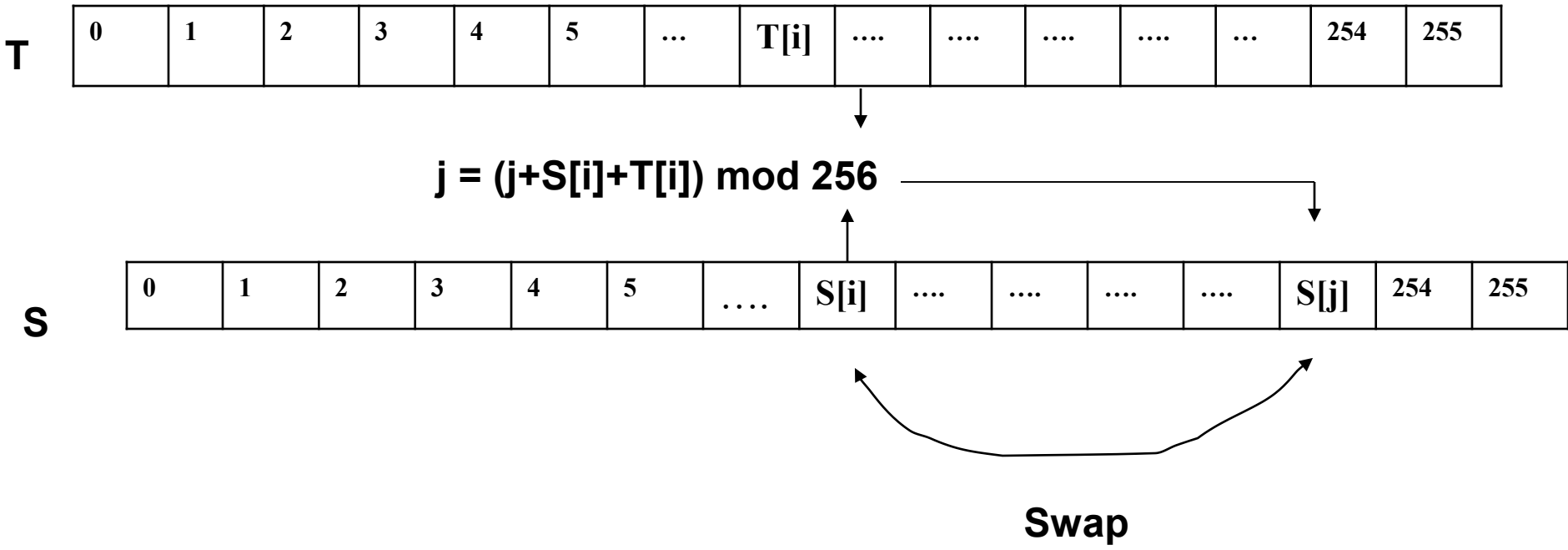
T



Initial state of S and T

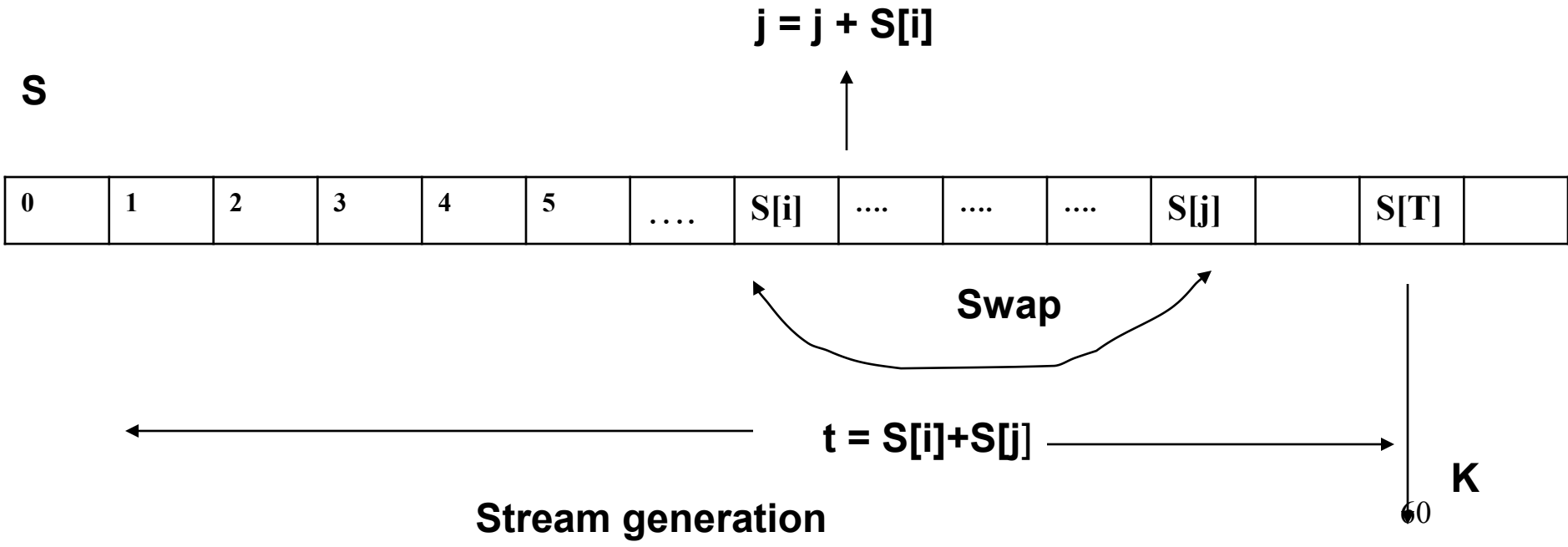
RC4

Swap S[i] and S[j] for all *i* = 0,1,…255. *j* = 0 in the beginning



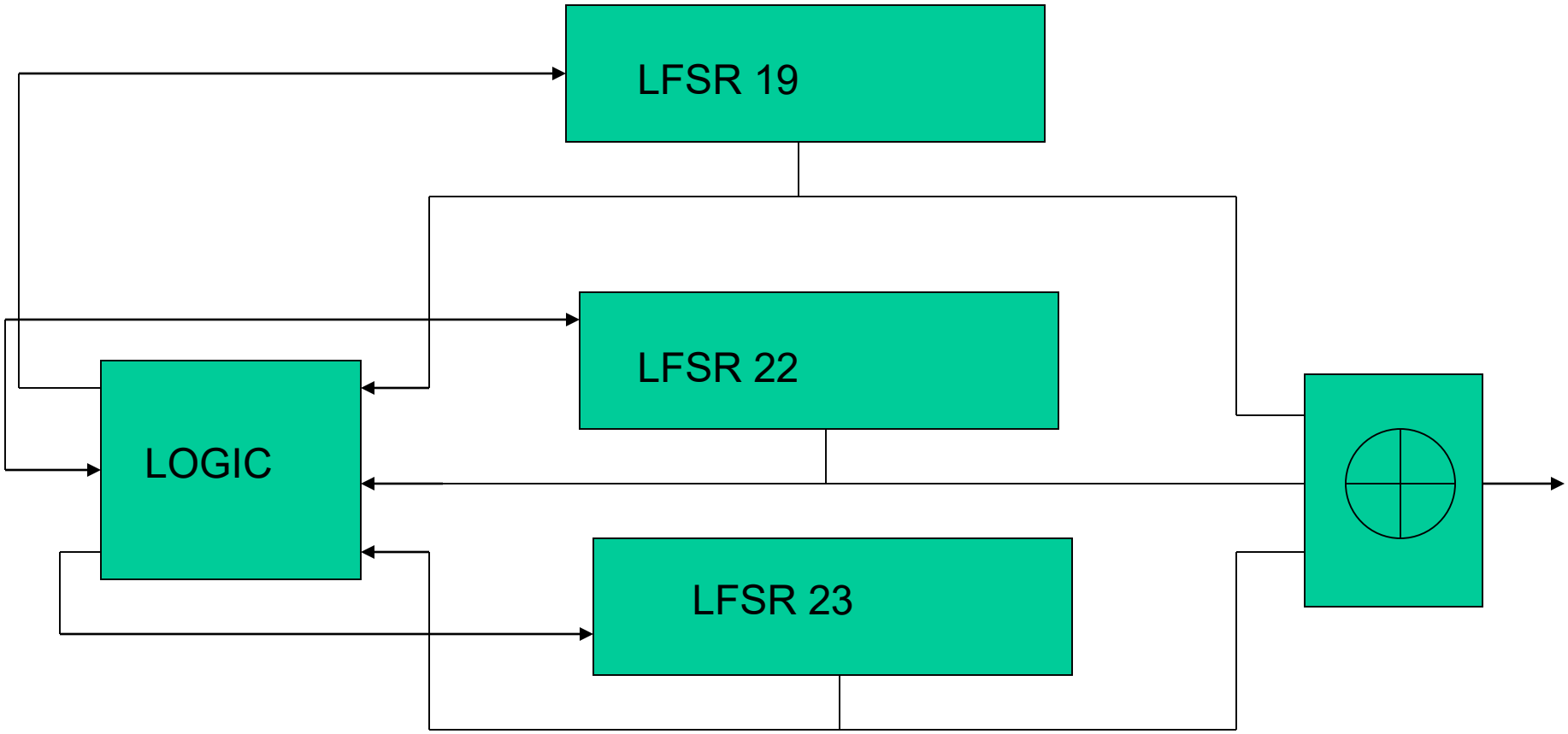
Initial permutation of S

RC4



Applications of stream ciphers

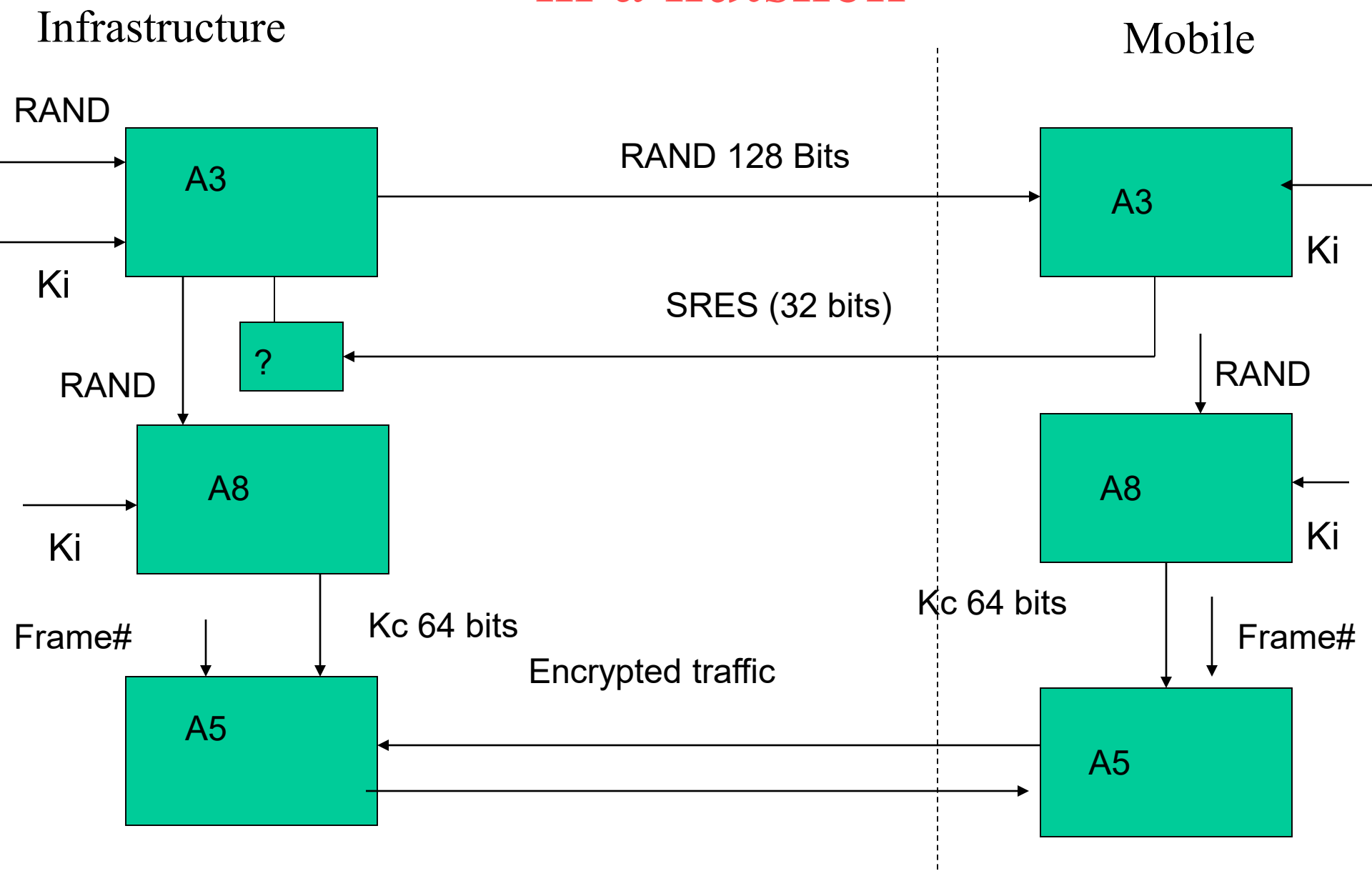
A5 Algorithm of GSM



- Stop and go clocked Shift registers
- One fixed Primitive polynomial for each

- $x^{19}+x^{18}+x^{17}+x^{14}+1$
- $x^{22}+x^{21}+1$
- $x^{23}+x^{22}+x^{21}+x^{8}+1$

GSM Authentication and encryption in a nutshell



GSM Authentication

- Network sends RAND (128 bits)
- K_i is secret key
- K_i, RAND used with Algorithm A3 to produce SRES (32 bits)
- K_i cannot be accessed by the user.

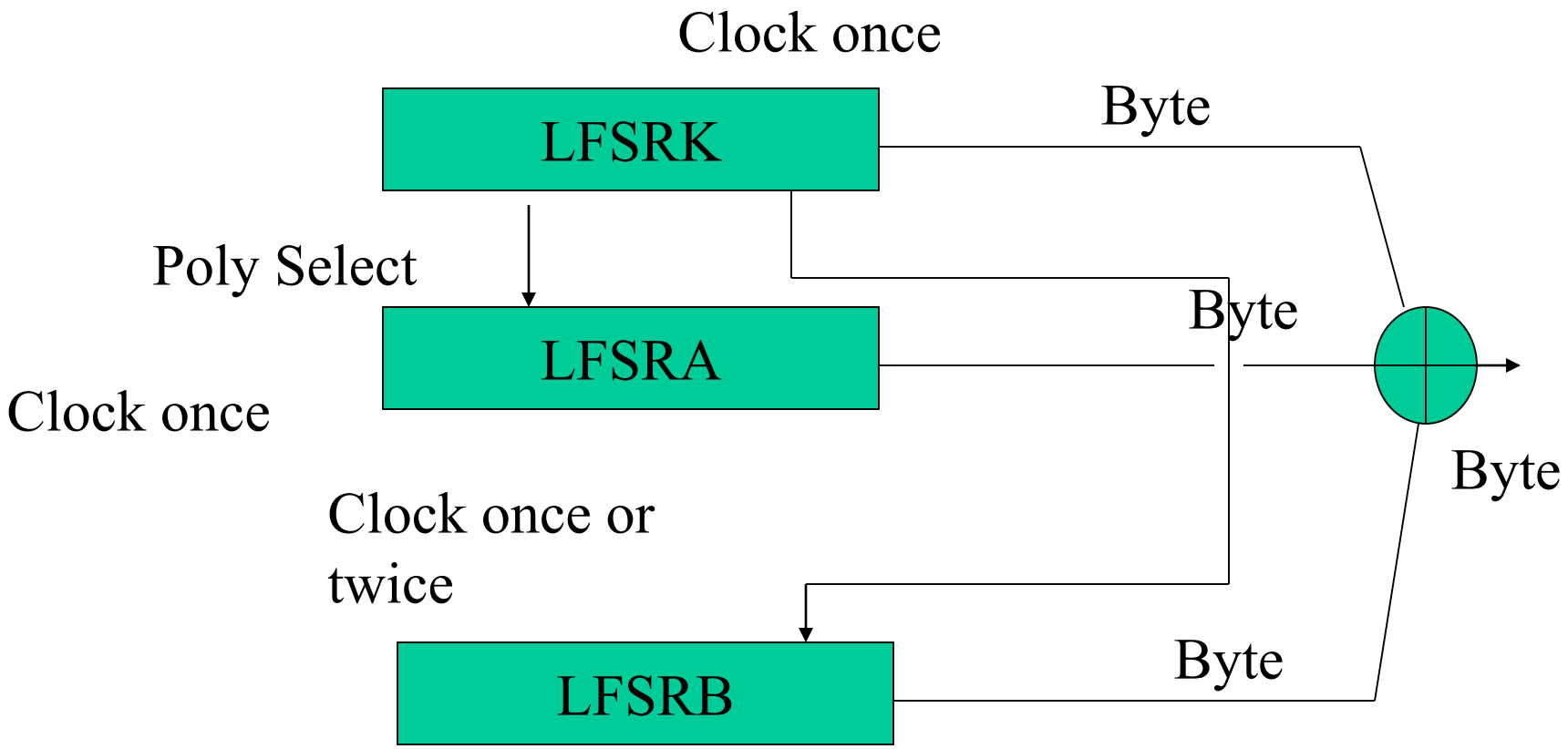
GSM Encryption

- Cipher Key K_c generated using K_i and RAND by algorithm A8
- K_c is 64 bits
- Frame number (22bits) and K_c used with A5 to generate 114 bit cipher sequence
- Speech is masked by cipher sequence and transmitted

ORYX

- The feedback function for LFSR*K* is
- $x32 + x28 + x19 + x18 + x16 + x14 + x11 + x10 + x9 + x6 + x5 + x + 1$:
- The feedback functions for LFSR*A* are
- $x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1$
- and
- $x32+x27+x26+x25+x24+x23+x22+x17+x13+x11+x10+x9+x8+x7+x2+x+1$:
- The feedback function for LFSR*B* is
- $x32 + x31 + x21 + x20 + x16 + x15 + x6 + x3 + x + 1$:

ORYX



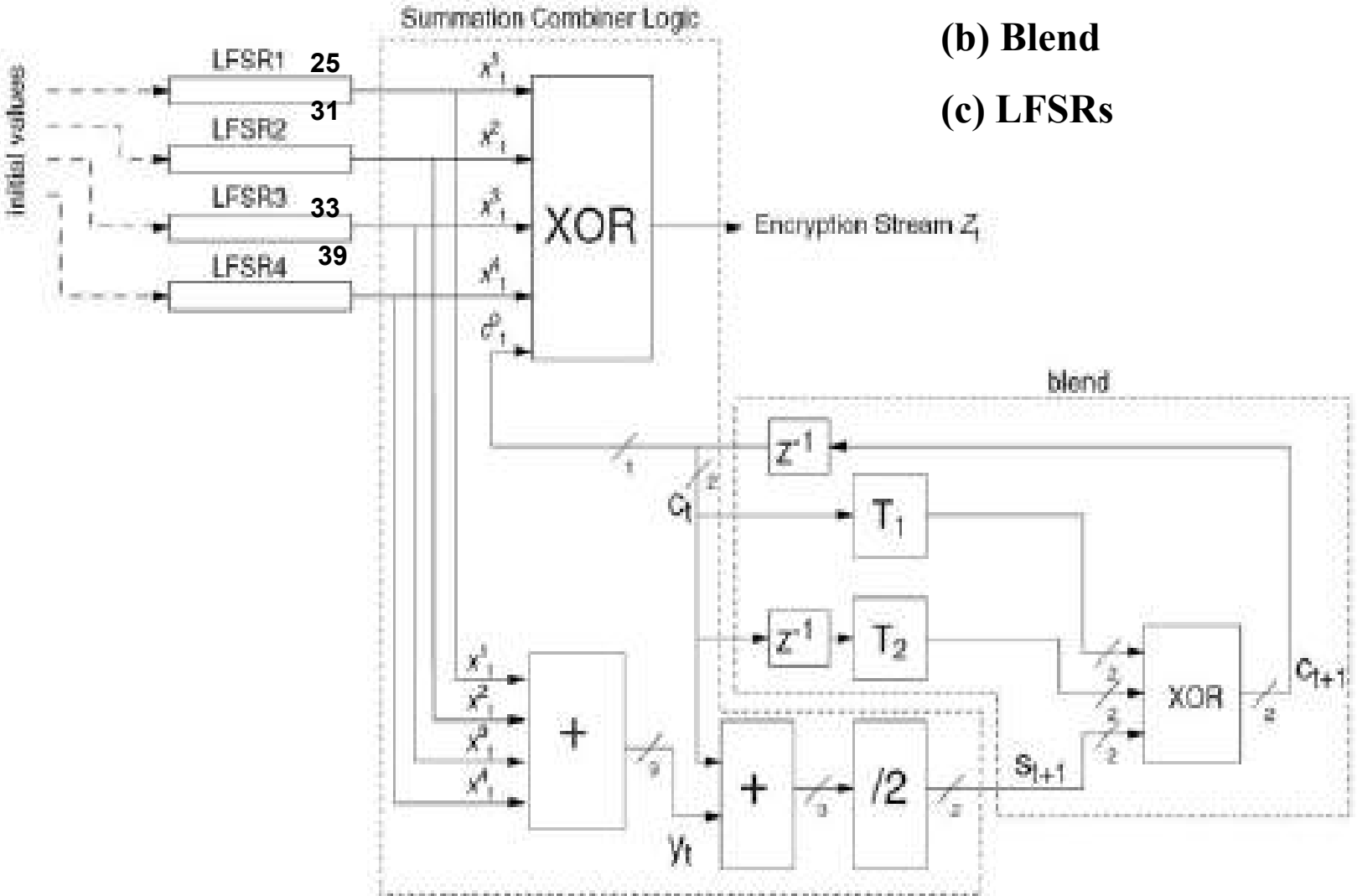
- 96 bit key space

Blue Tooth Sequence generator

(a) Summation combiner logic

(b) Blend

(c) LFSRs

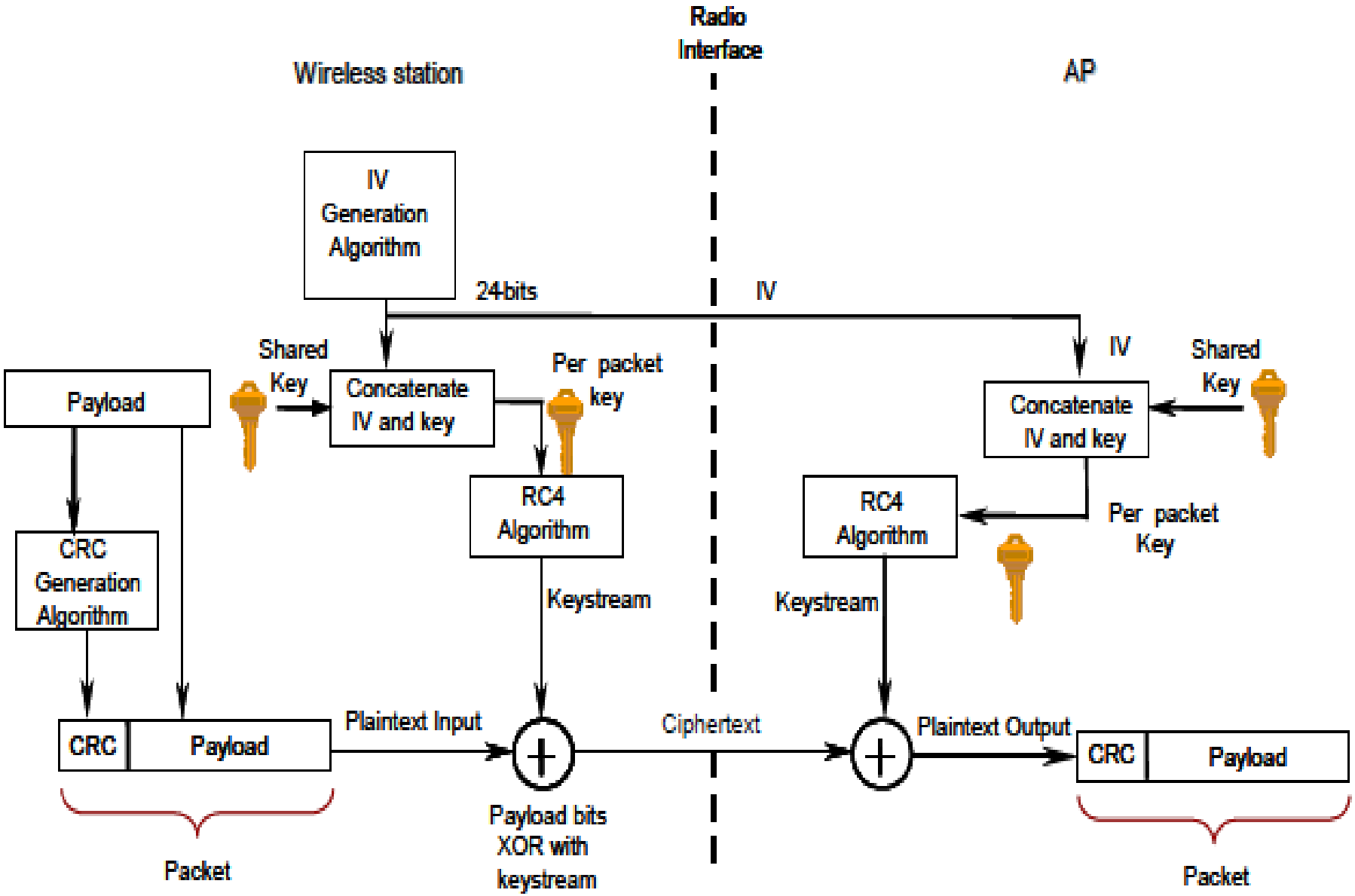


Details

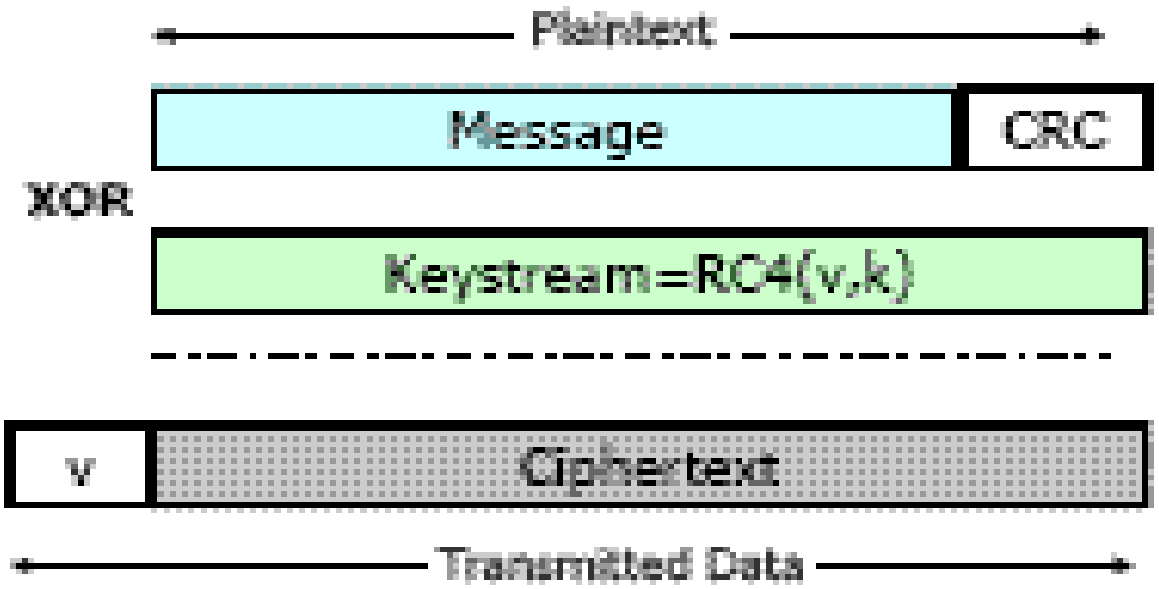
- $Y_t = x_1 + x_2 + x_3 + x_4$
- $C_t = c_1 c_o$
- $S_{t+1} = \text{Trunc}[\,(Y_{t+} C_t)/2]$
- $C_{t+1} = S_{t+1}$ *Exor* $T_1(C_t)$ *Exor* $T_2(C_{t-1})$
- T_1: $\{x_1, x_0\} \rightarrow \{x_1, x_0\}$
- T_2: $\{x_1, x_0\} \rightarrow \{x_0,\ x_0$ *exor* $x_1\}$
- T_1, T_2 are linear mappings
- z^{-1} is delay

Details of LFSRs used

LFSR Number	LFSR Length	LFSR Feedback Polynomial	Hamming Weight
1	25	t25 + t20 + t12 + t8 + 1	5
2	31	t31 + t24 + t16 + t12 + 1	5
3	33	t33 + t28 + t24 + t4 + 1	5
4	39	t39 + t36 + t28 + t4 + 1	5


Radio Interface
Wireless station
AP
IV Generation Algorithm
24bits
IV
Shared Key
Concatenate IV and key
Per packet key
Payload
CRC Generation Algorithm
RC4 Algorithm
Keystream
IV
Shared Key
Concatenate IV and key
RC4 Algorithm
Per packet Key
Keystream
CRC
Payload
Plaintext Input
Ciphertext
Plaintext Output
CRC
Payload
Packet
Payload bits XOR with keystream
Packet

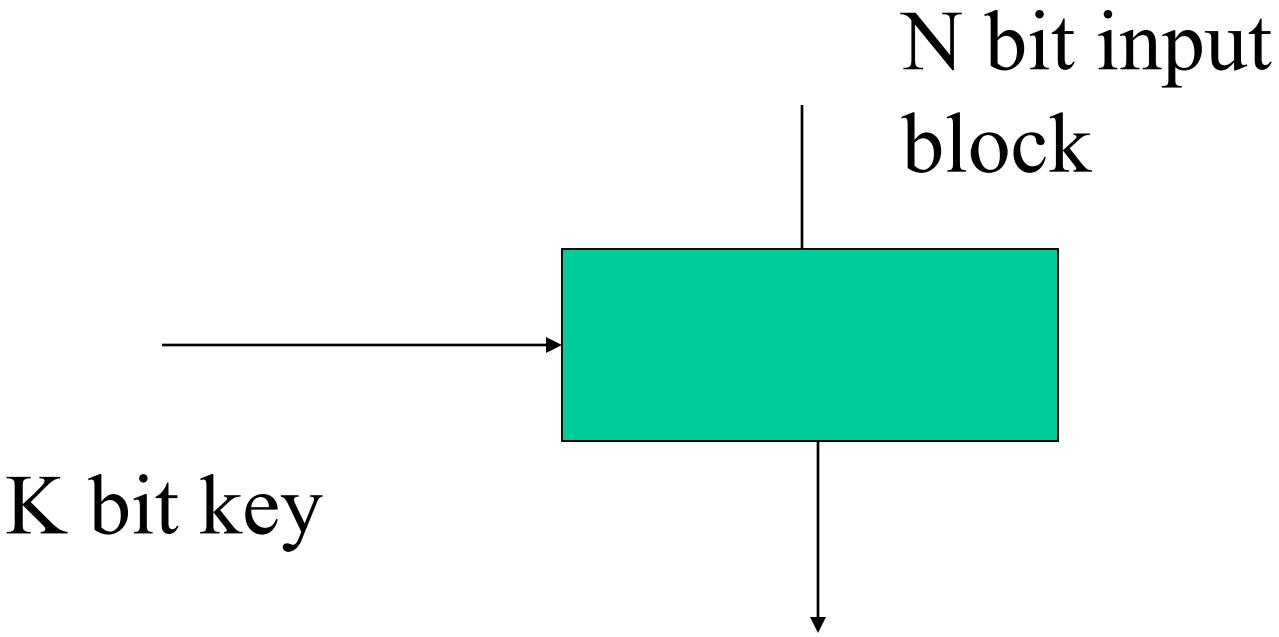
Picture of WEP



Note that v (24 bits) Is transparent.

Block ciphers

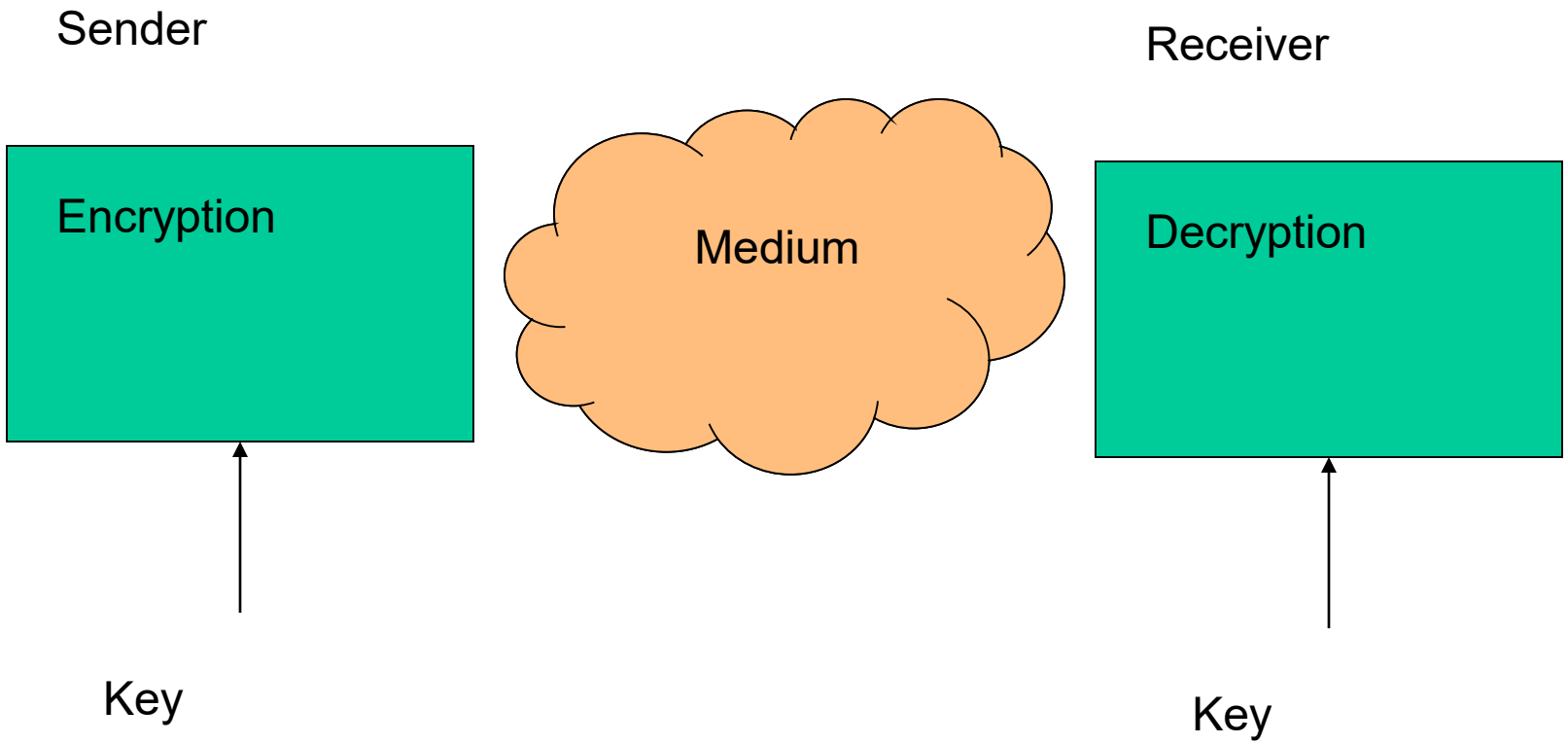
Block ciphers



Symmetric key encryption algorithms

- Data encryption standard(DES)
- Triple DES
- International data encryption algorithm (IDEA)
- Blowfish
- Many more
- RIJNDAEL - the advanced encryption standard

What are Symmetric Key Based Systems?



- Both Encryption and Decryption keys are same.

General Features/Specifications

- Block length in bits
- Key length in Bits
- Rounds
- Operations in Each round
- Key Schedule for all rounds
- Round Key generation
- Decryption
- Modes of operation
- Any Weak Keys
- Complexity / Execution time Benchmarks

Confusion and diffusion

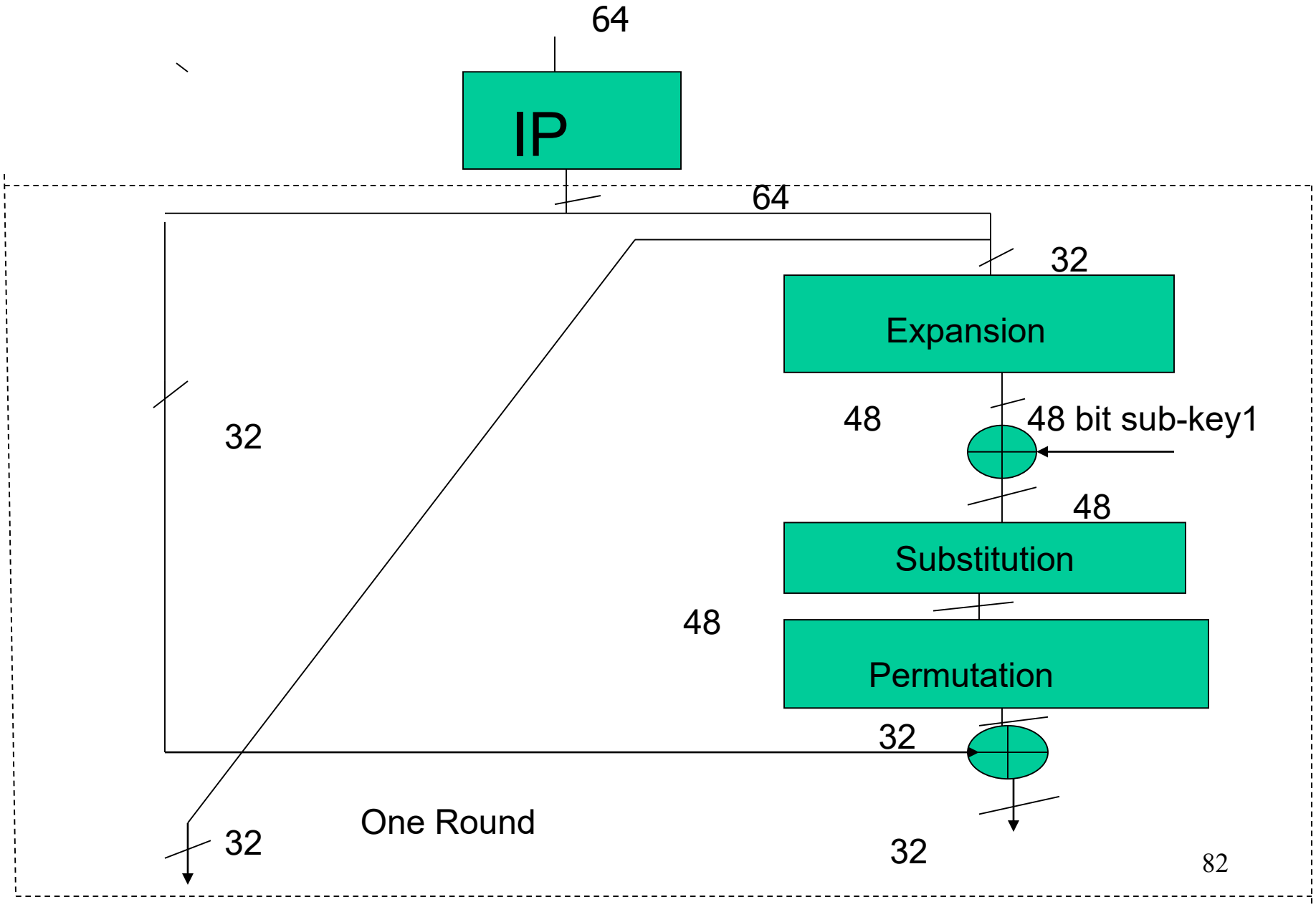
- Relationship between plain text and cipher text shall be complicated; if we change a character of the plaintext, then several characters of the ciphertext should change - Diffusion.
- Permutations do this job
- Relationship between key and cipher text shall be complicated to thwart discovery of key; each character of the ciphertext should depend on several parts of the key –confusion
- Substitutions do this job.

DES

Features

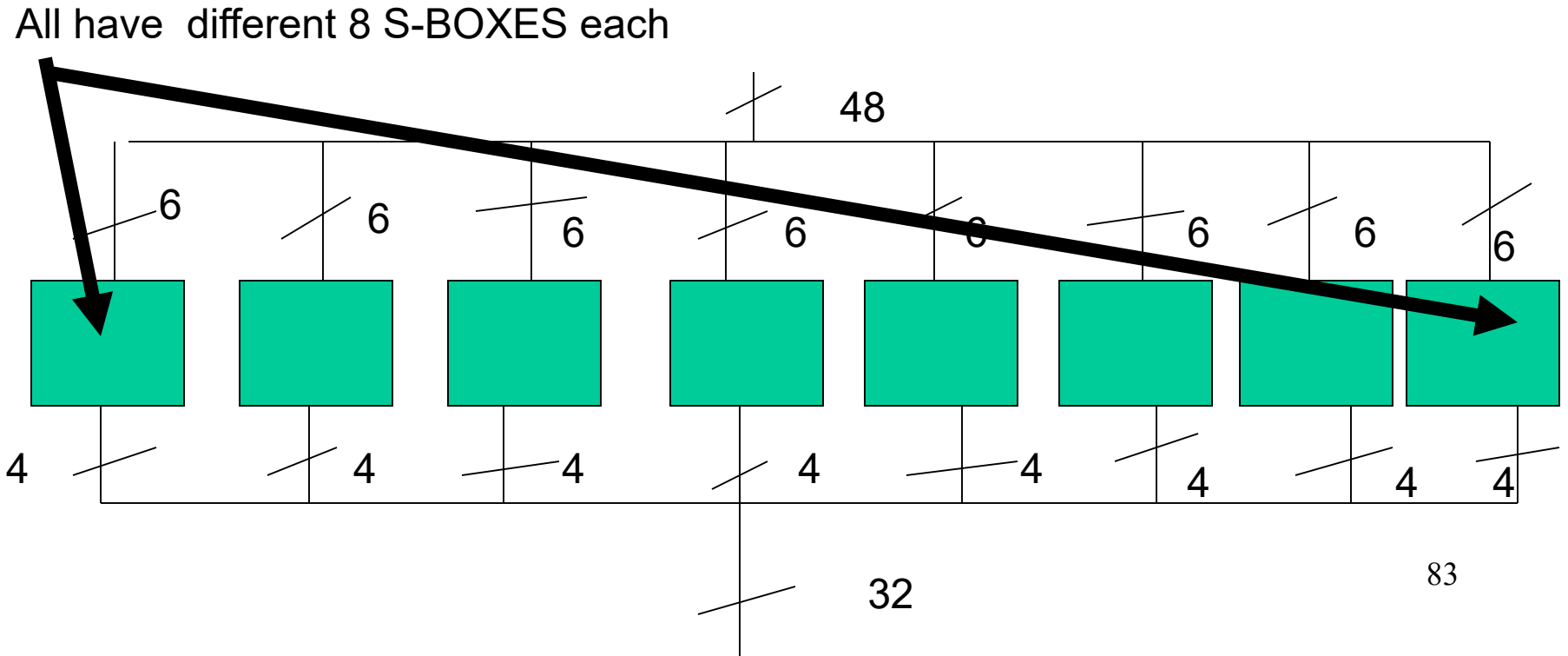
- **Oldest**
- **Symmetric key system (both encryption and decryption use the same key)**
- **64 bit data blocks**
- **56 bit key(64 bit key including 8 parity bits)**
- **Initial Permutation**
- **16 rounds each with a subkey**
- **Uses expansion, substitution boxes, masking,left-right block shuffling**
- **Last round left and right words are swapped**
- **Followed by IP^{-1}**
- **16 trivial keys not recommended**

DES Structure



DES

- **use of substitution to obtain 32 bits from 48 bits**
- **4 bit substitutions (s-Boxes) are used**
- **8 s-boxes are used**
- **Each s-box needs 16 nibbles of memory**

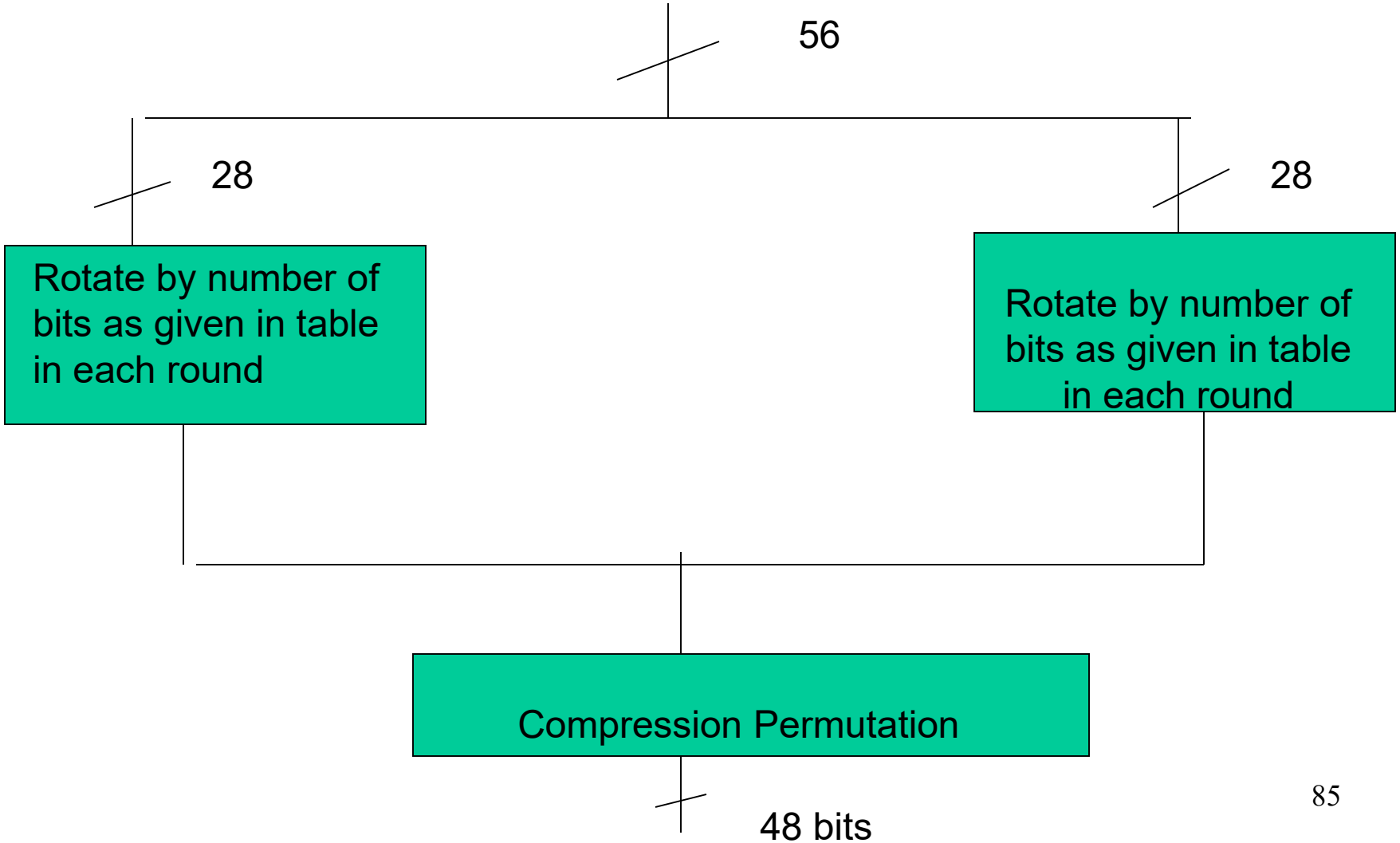


DES

- First and last bits of each 6 bit word address one of the rows. Middle 4 bits pick the column contents.
- **S-BOX 1: 4 nibble S-boxes, Can be considered as 6X4 S-Box also.**

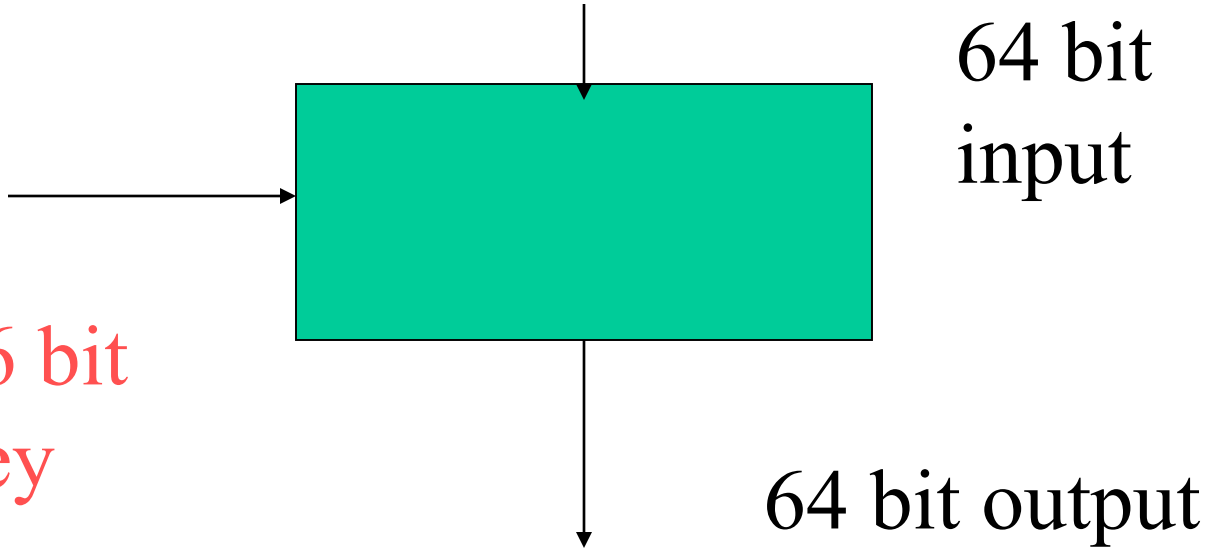
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Sub Key generation



DES

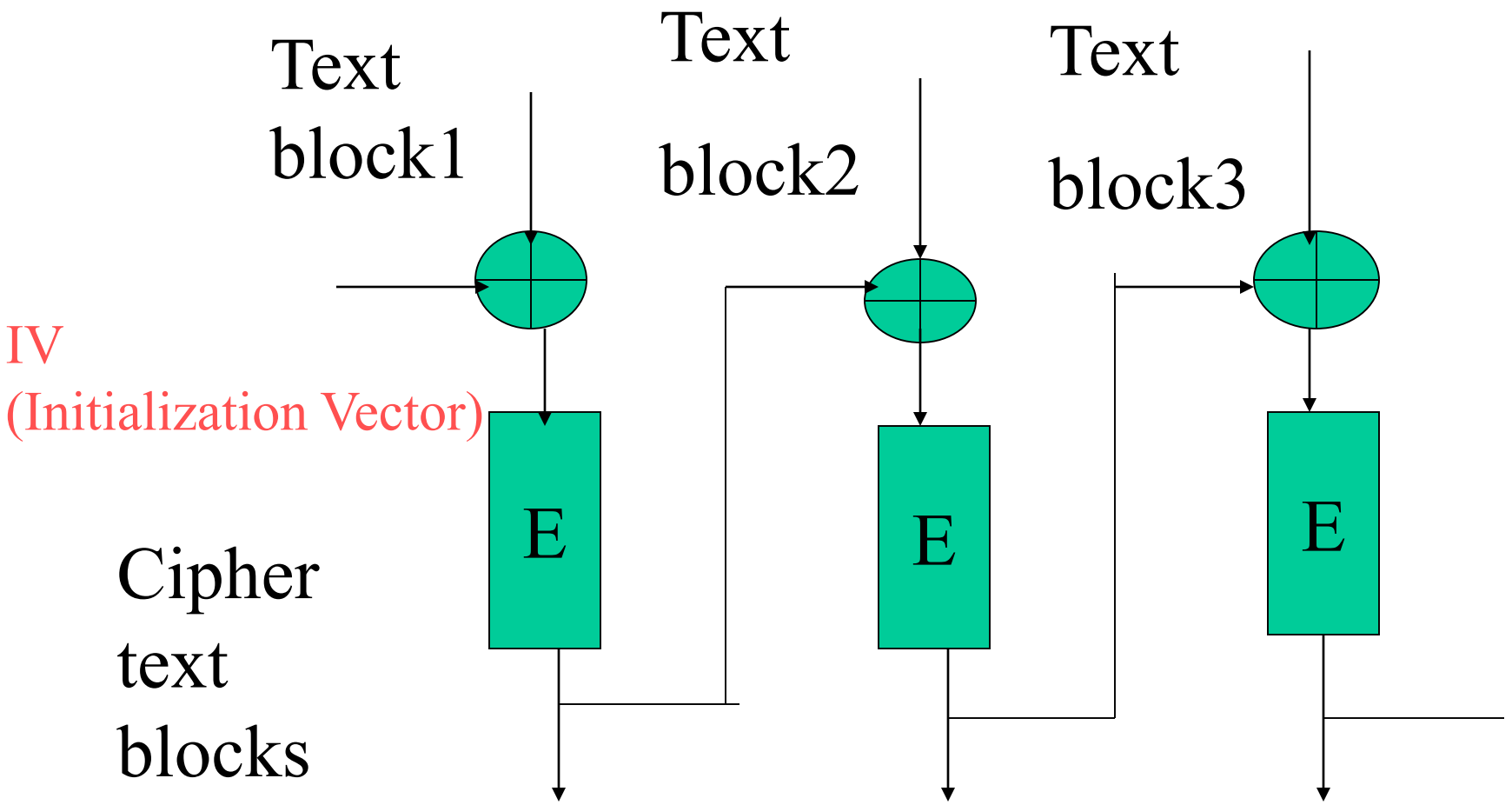
- Four modes of operation:
- A) ECB (Electronic Code Book)



This is what we have described till now!

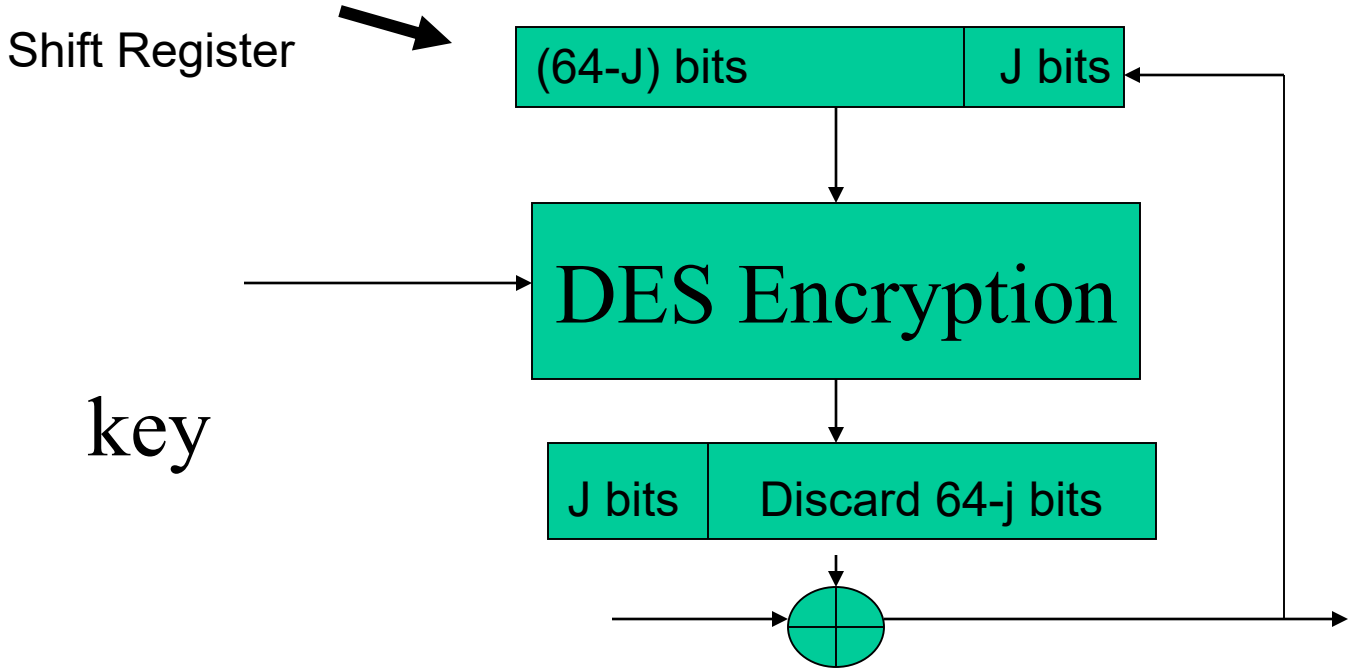
DES

- B)CBC(Cipher Block chaining)



DES

- C) CFB(CIPHER FEEDBACK MODE)

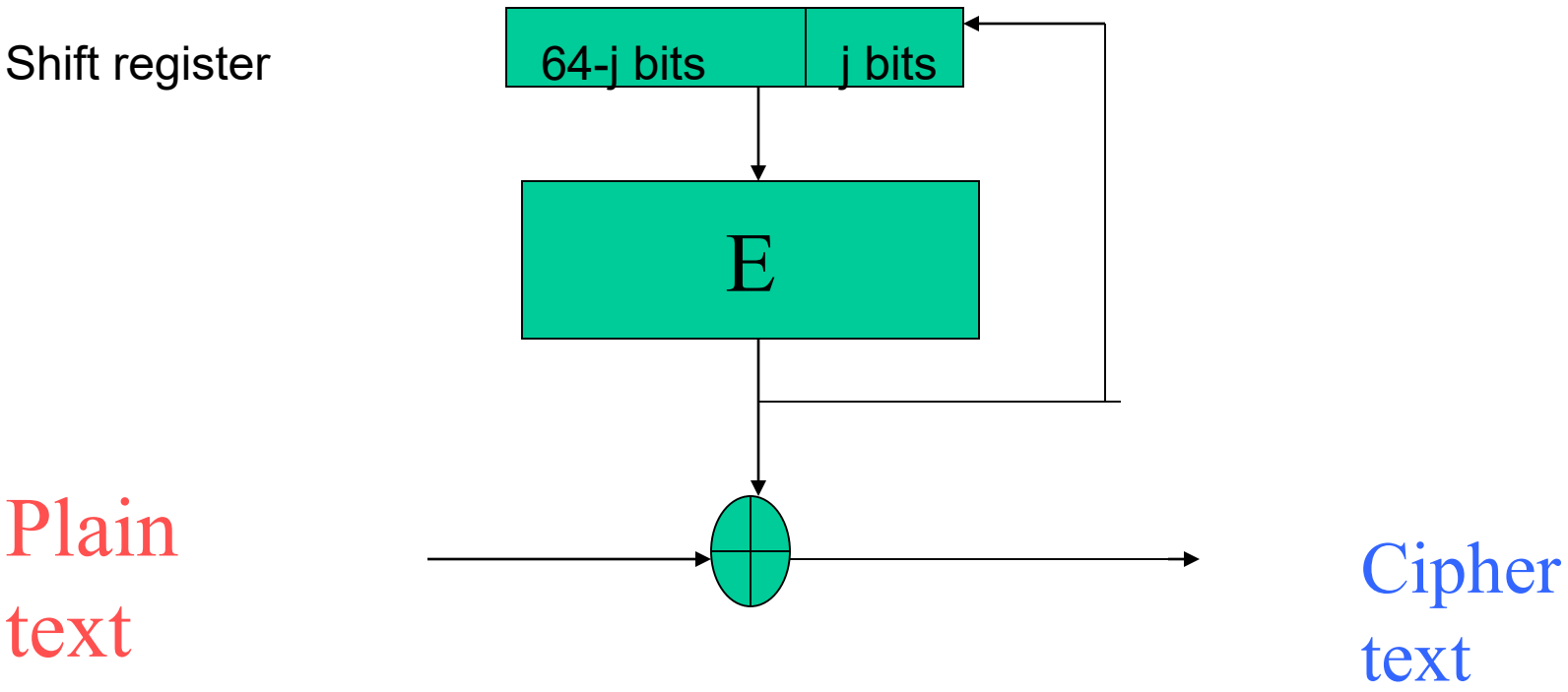


Plain text j bits

Cipher text j bits

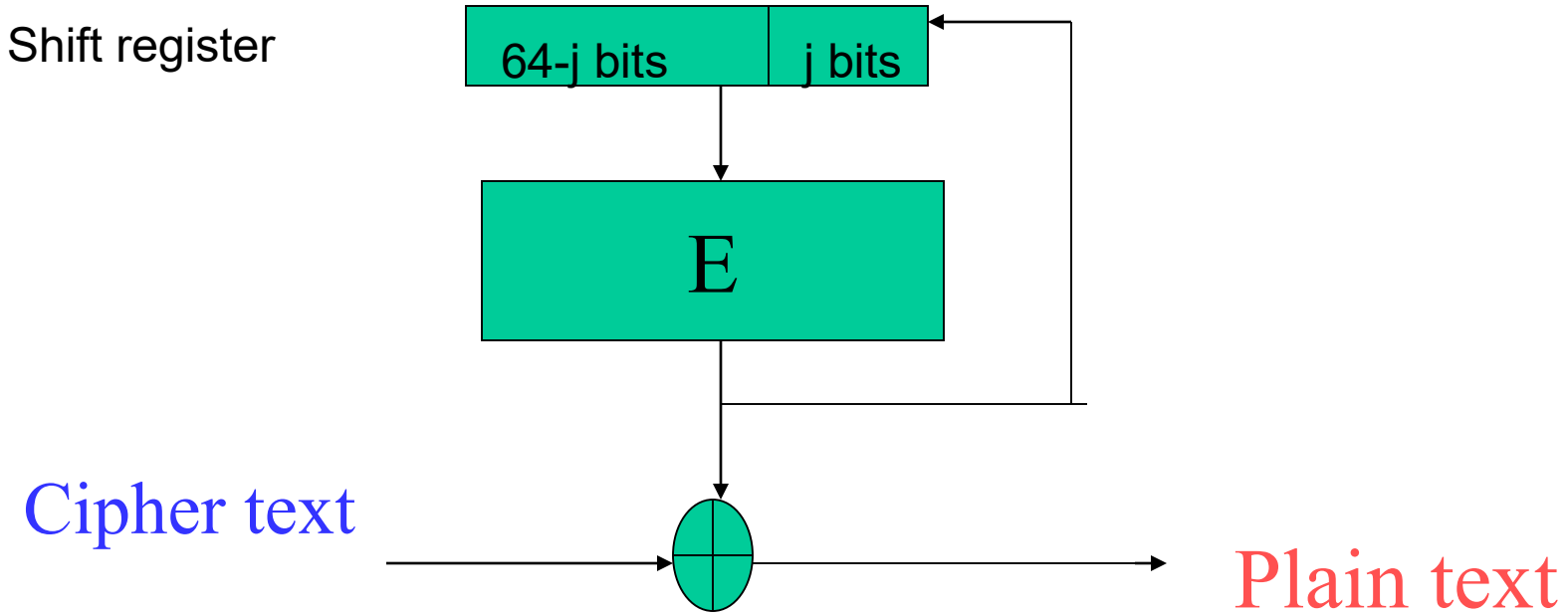
DES

- D) OFB (Output feedback) Encryption



DES

- OFB (Output feedback) Decryption



OFB has no error extension.

DES

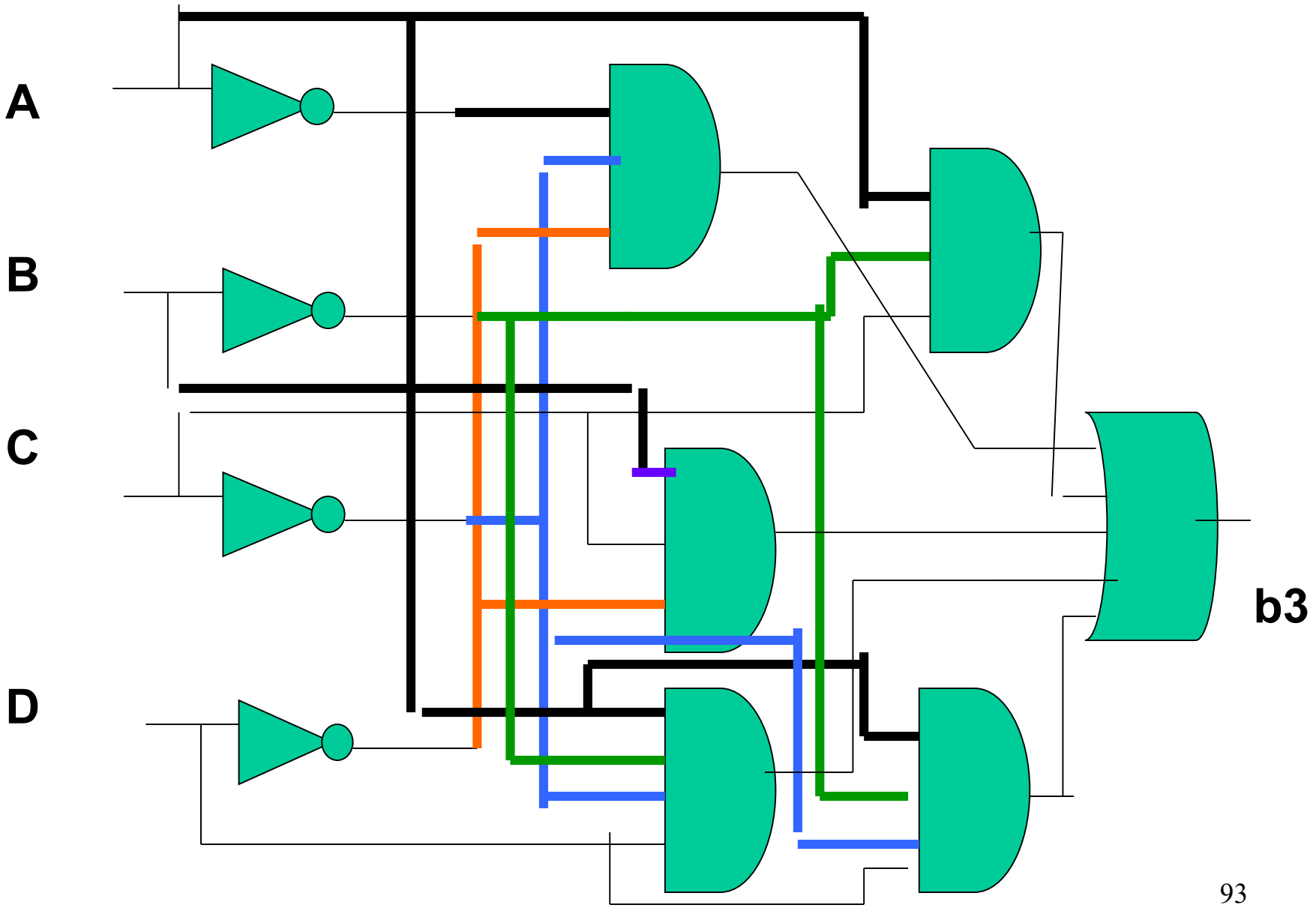
- Weak keys (2 sub keys only)
- Semi-weak keys (Four subkeys only)
- Complement keys
- More rounds increase the resistance to differential cryptanalysis
- S-boxes can decide resistance to linear cryptanalysis

What is a S-BOX?

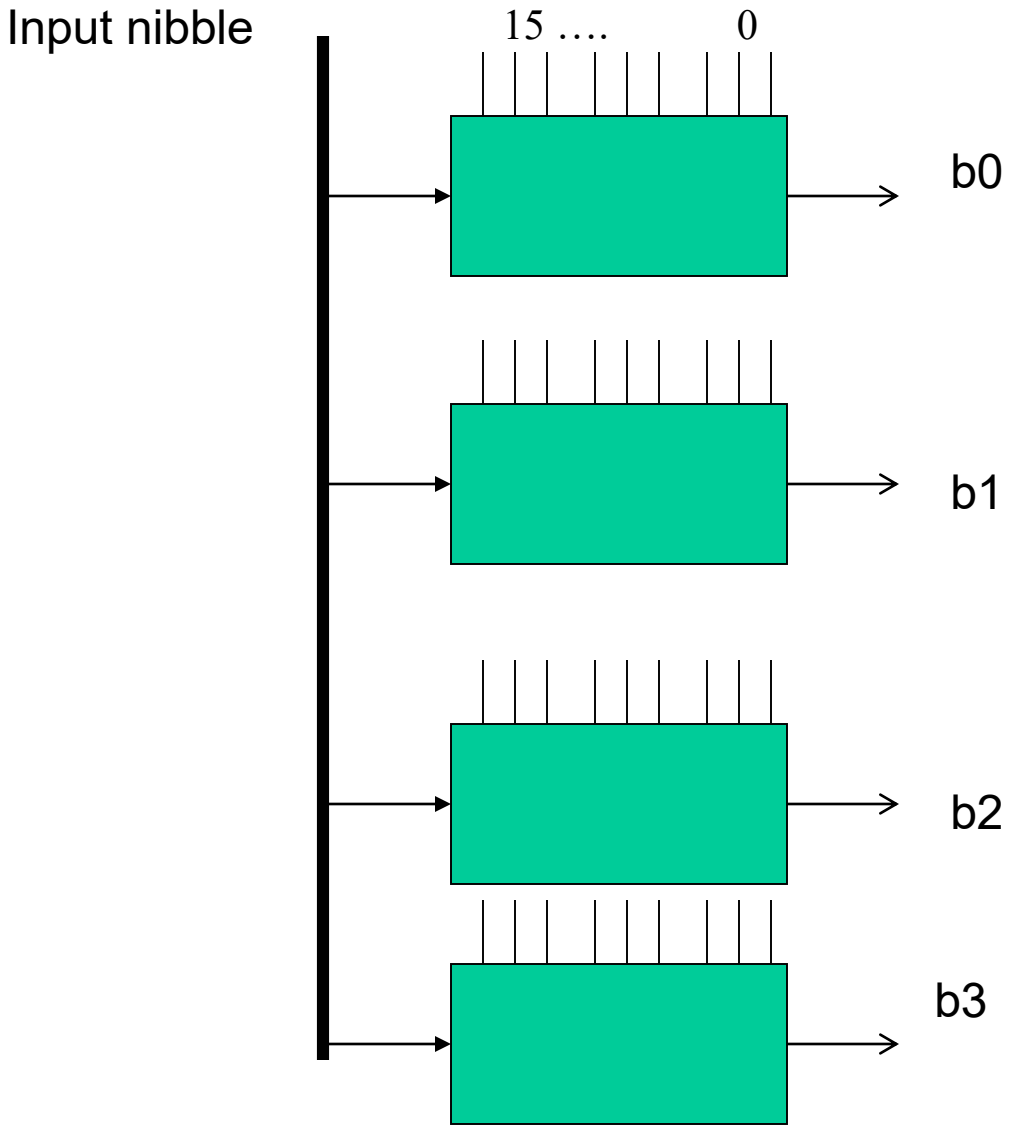
- S1 First row
- 14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7
- Analyze the Sequences of b_3, b_2, b_1, b_0
- The logic function of b_3 assuming an input from a counter counting from zero to 15 as follows:
- **b_3 = D′C′A′ +D′CA+ D′CBA′ +DC′A+DCB′A**

b3b2b1b0
1110
0100
1101
0001
0010
1111
1011
1000
0011
1010
0110
1100
0101
1001
0000
0111

$$b_3 = A'C'D' + AB'C + BCD' + AB'C'D + ABC'D$$



S-BOX based on Multiplexer



Hardwire all inputs of

Mux 16:1 to logic one and zero

as needed.

S-Box One column Algebraic Normal form

X3	X2	X1	X0	Sequence	Step1	Step2	Step 3	Step4	Answer
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	X0
0	0	1	0	1	1	0	0	0	
0	0	1	1	0	1	0	0	0	
0	1	0	0	0	0	0	1	1	X2
0	1	0	1	1	1	1	0	0	X2x0
0	1	1	0	1	1	1	1	1	X2X1
0	1	1	1	1	0	1	1	1	X2X1X0
1	0	0	0	0	0	0	0	1	X3
1	0	0	1	1	1	1	1	0	
1	0	1	0	0	0	0	0	0	
1	0	1	1	1	1	0	0	0	
1	1	0	0	0	0	0	0	1	X3X2
1	1	0	1	1	1	1	0	0	
1	1	1	0	0	0	0	0	1	X3X2X1
1	1	1	1	0	0	1	1	0	

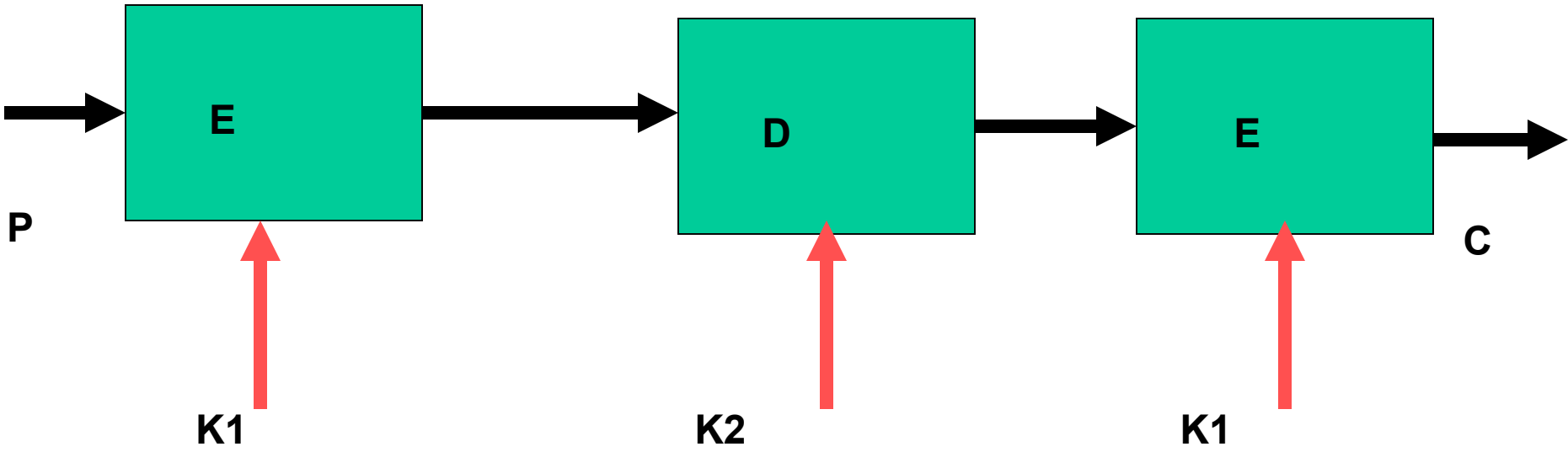
1+X0+X2+X3+X2X0+X2X1+X2X1X0+X3X2+X3X2X1 Plus is Exclusive -OR

Triple DES

- Uses 112 bit key split as two 56 bit keys
- Encryption: E (key 1)-D(key 2)-E(key1)
- Decryption: E(key 1)-D(key 2)- D(key 1)
- Three independent keys also can be used.
- The purpose of using two keys was to ensure compatibility between old DES users and new Triple DES users (K1 = K2)

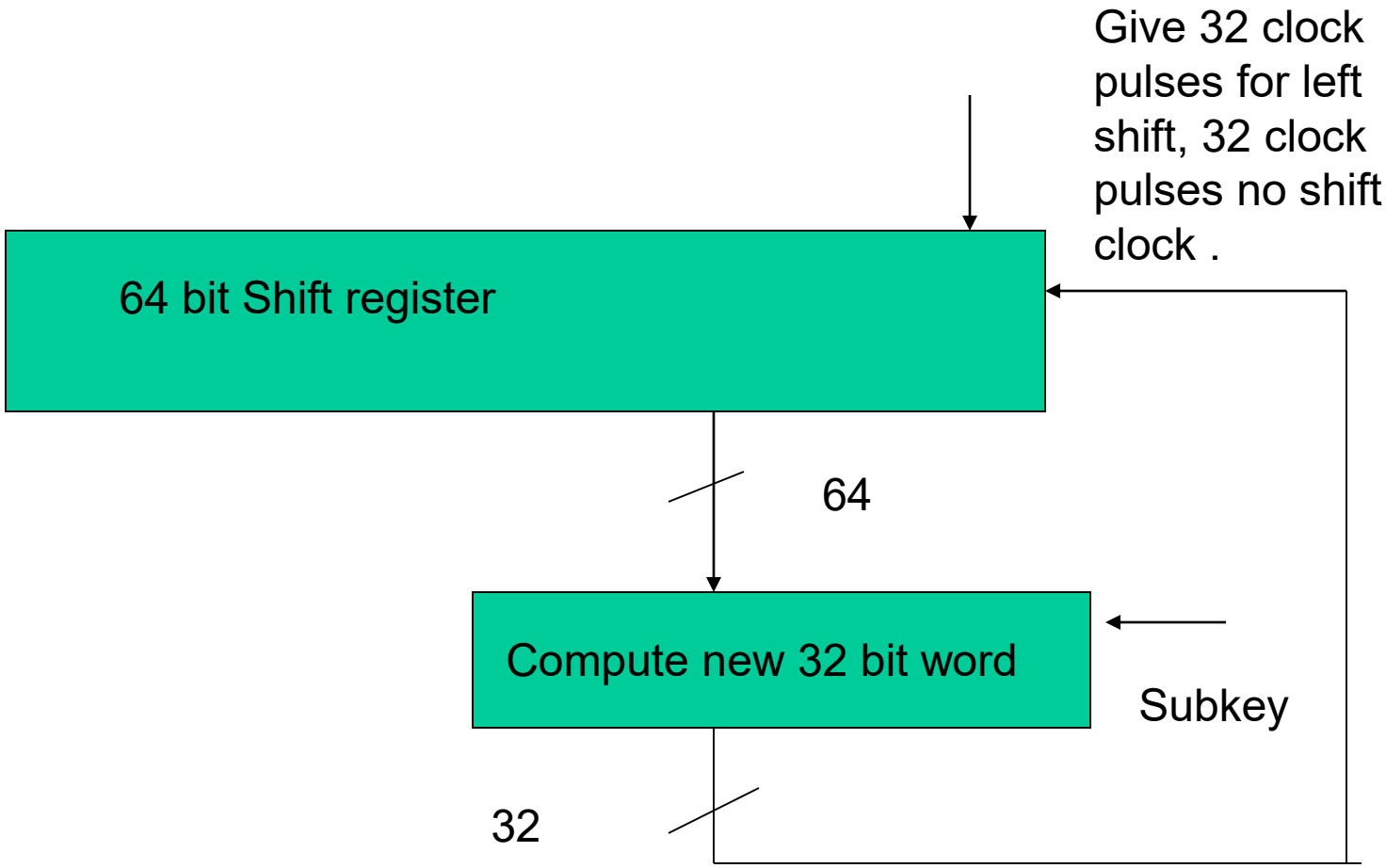
Triple DES

C=EK1[DK2[EK1[P]]]



- 112 bit key (caution: different K1 and K2!!!)

Block cipher can be implemented as a stream cipher

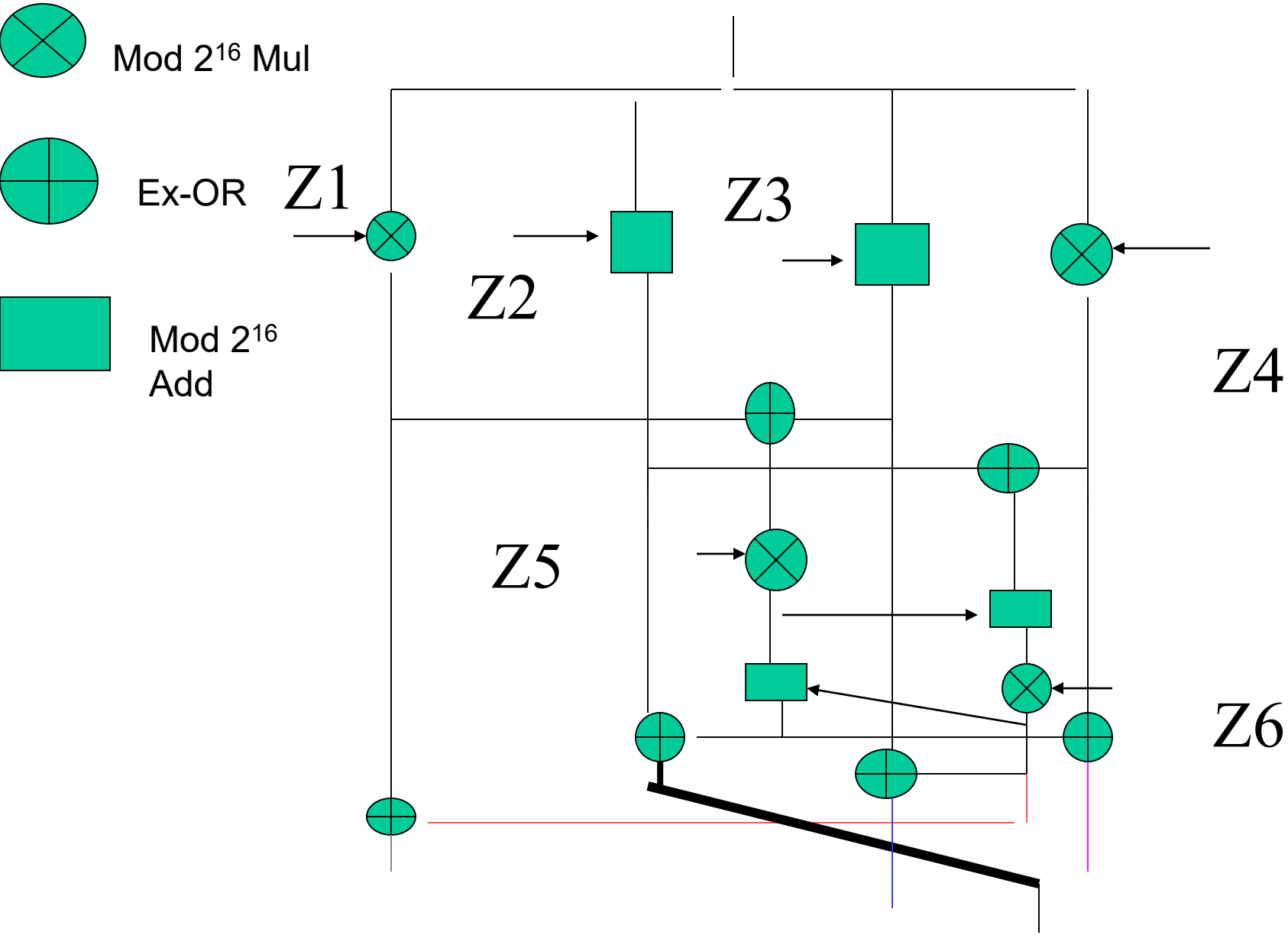


- A Block cipher is a NLFSR(non-linear Feedback Shift register)!

IDEA (international data encryption algorithm)

- 128 bit key
- 64 bit block encryption
- Designed for easy implementation on commercial digital signal processors
- Eight rounds

IDEA



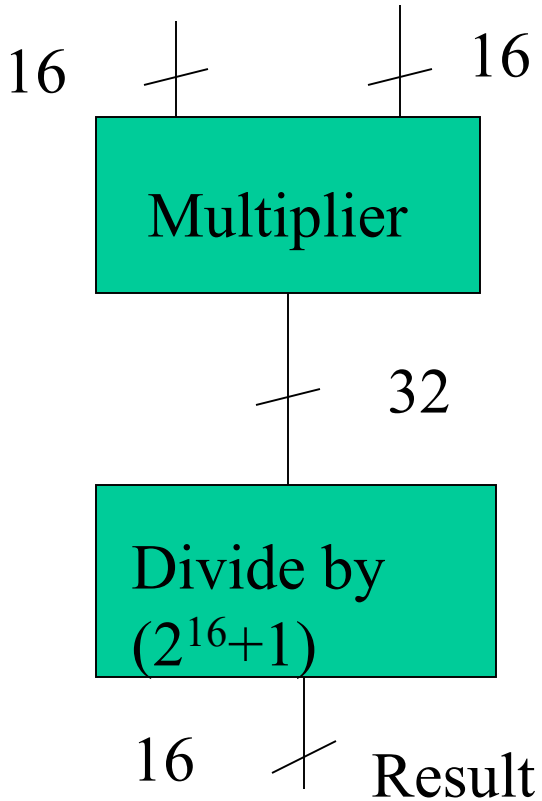
IDEA

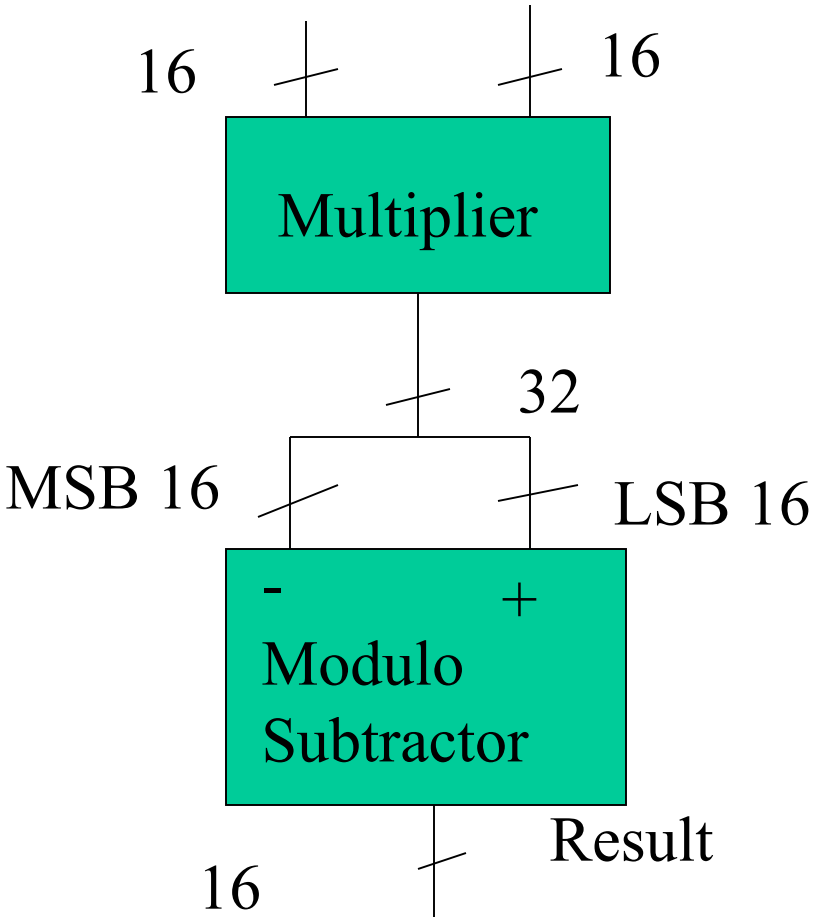
- modulo 2^{16} addition: add and ignore carry; MSBS have dependency on more number of bits than LSBs. Known as T-Function (triangular Function)
- modulo $(2^{16}+1)$ multiplication:
- conventional method: multiply a and b first ,divide by $(2^{16}+1)$
- simple method: multiply a and b first to get 32 bit word.
- subtract MSB from LSB,if negative add $(2^{16}+1)$

IDEA

- Example:
- $1011_2 \times 1100_2 \bmod (2^4+1)$
- $1011_2 \times 1100_2 = 1000\ 0100_2$
- $0100_2 - 1000_2 = -(0100)_2$
- Since answer is negative add 17.
- Thus $11 \times 12 \bmod 17 = 13$.
- One multiplication, one subtraction and one conditional addition are needed

Modulo ($2^{16}+1$) multiplication





Permutation

- Same as S-Box
- Every bit shall move from its original position.
- Time consuming for bit wise permutation
- Fast Permutation algorithms available for software implementations
- Dynamic change of permutation by addition of 3 bit random number mod 8 to all the addresses. Does not need storage for expanding the space.
- Example 4 7 0 3 5 6 1 2 add 2 mod 8 to get 6 1 2 5 7 0 3 4

Pseudo Hadamard Transform

- The bit string must be of even length, so it can be split into two bit strings *a* and *b* of equal lengths, each of *n* bits. To compute the transform, *a*' and *b*', from these we use the equations:
- $a' = a + b \pmod{2^n}$
- $b' = a + 2b \pmod{2^n}$
- To reverse this, clearly:
- $b = b' - a' \pmod{2^n}$
- $a = 2a' - b' \pmod{2^n}$

Extension of PHT

$$\begin{vmatrix} 4 & 2 & 2 & 1 \\ 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{vmatrix} \bmod 2^{32}$$

Rijndael (AES)

- Variable block length (128,192,256 bits)
- Variable key length(128,192 or 256 bits)
- Block cipher
- Data and key arranged as rows and columns
- Byte level design
- Suitable for DSP or Microprocessor based or ASIC implementation

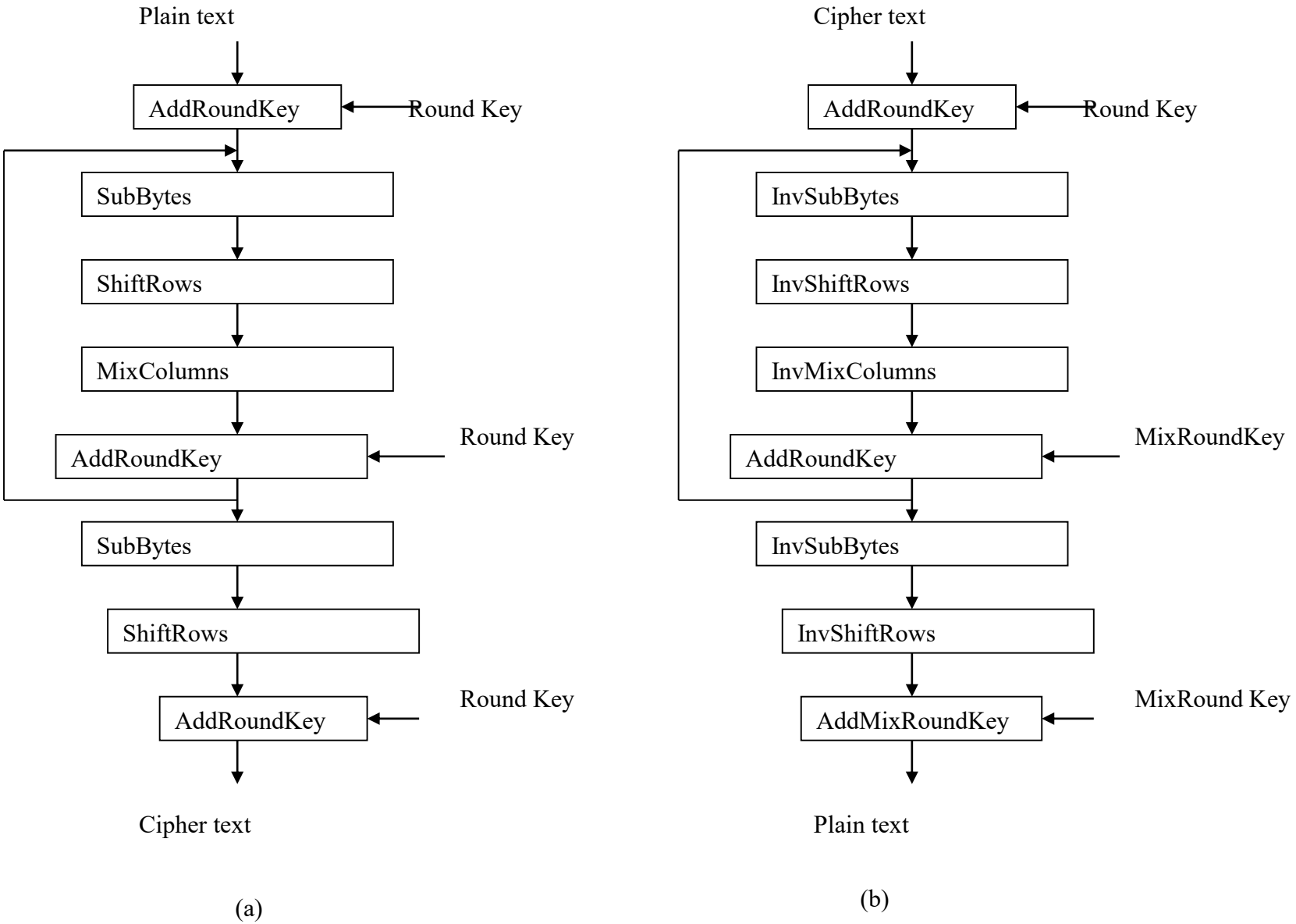


Fig.1. AES algorithm: (a) Encryption (b) Equivalent Inverse Decryption

AES

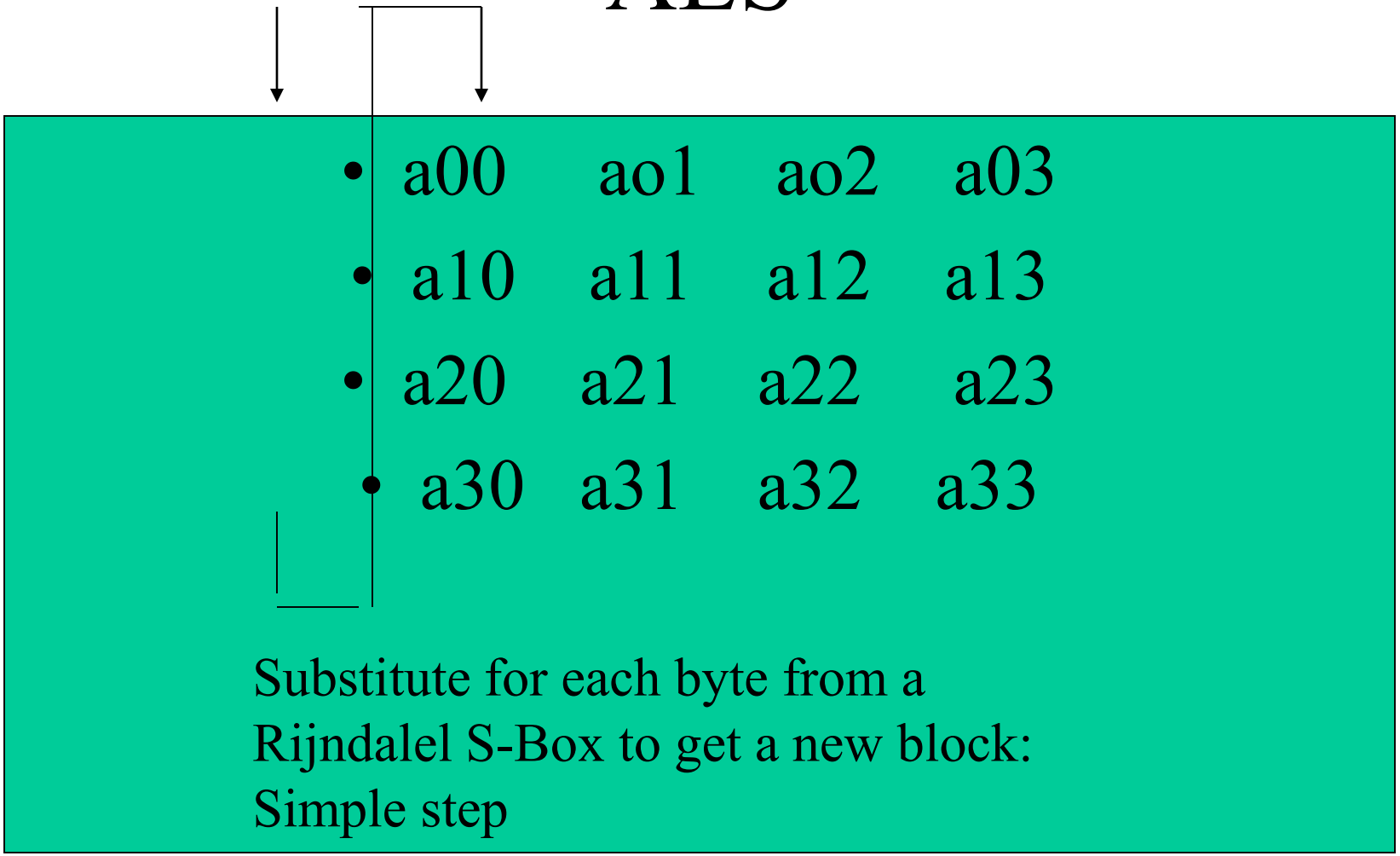
- Four Rows
- N_b columns : N_b = Block length/32
- N_k columns : N_k = Key length /32
- Number of rounds dependent on N_b and N_k:

N_k \ N_b	4	6	8
4	10	12	14
6	12	12	14
8	14	14	14

AES

- Rounds shown in Table +1 needed
- Each round consists of four operations:
- 1)Byte Substitution
- 2) Shift row
- 3)Mix column
- 4) Add Round key (modulo 2 bit by bit)

AES


a00 ao1 ao2 a03
a10 a11 a12 a13
a20 a21 a22 a23
a30 a31 a32 a33
Substitute for each byte from a
Rijndalel S-Box to get a new block:
Simple step

AES

- Shift row

First row no shift					
Second row One byte left circular shift					
2 byte left circular shift Third row					
Fourth row Three byte left circular shift					

Original

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

The result is the permutation

1 6 11 16 5 10 15 4 9 14 3 8 13 2 7 12

AES

- Mix column Transformation
- All bytes are treated as polynomials
- Example the byte $b_7b_6b_5b_4b_3b_2b_1b_0$ is the polynomial $b_7x^7{}_+b_6x^6{}_+b_5x^5{}_+b_4x^4{}_+b_3x^3{}_+b_2x^2{}_+b_1x_+b_0$
- Columns are considered as polynomials over $GF(2^8)$
- The irreducible 8^{th} degree polynomial used is $x^8+x^4+x^3+x+1$

AES *MixColumn*

- $b(x)=[c(x) \times a(x)] \bmod (x^4+1)$
- $c(x) =$ “03” $x^3 +$ “01”$x^2 +$ “01”$x+$”02”
- we thus obtain all new columns corresponding to $a(x)$.
- This is a 32 bit S-BOX which would have needed huge ROM if directly implemented.

Example

- $d(x)=[a(x).b(x)] \bmod (x^4+1)$
- $a(x) = a_3.x^3 + a_2.x^2 + a_1.x + a_0$
- $b(x) = b_3.x^3 + b_2.x^2 + b_1.x + b_0$
- $d(x)=c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0$
- $c_0= a_0b_0$, $c_4=a_3b_1+a_2b_2+a_1b_3$
- $c_1=a_1b_0+a_0b_1$, $c_5= a_3b_2+a_2b_3$
- $c_2=a_2b_0+a_1b_1+a_0b_2$, $c_6=a_3b_3$
- $c_3=a_3b_0+a_2b_1+a_1b_2+a_0b_3$
- All + are Exclusive OR
- But $x^4=1, x^5=x, x^6=x^2 \bmod (x^4+1)$

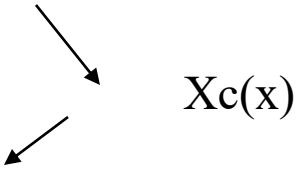
- $c_0 = a_0b_0+a_3b_1+a_2b_2+a_1b_3$
- $c_1 = a_1b_0+a_0b_1+a_3b_2+a_2b_3$
- $c_2 = a_2b_0+a_1b_1+a_0b_2+a_3b_3$
- $c_3 = a_3b_0+a_2b_1+a_1b_2+a_0b_3$
- Each of the above is a multiplication in GF(8)
- Fortunately, all b_i s are simple.
- 02H or 03 H or 01H or 01H

Example

- **Column 87 6E 46 A6 maps to 47 37 94 ED**
- **Proof for obtaining 47:**
- **(87).(02)+(6E).(03)+(46).(01)+(A6).(01)**
- **87.02= 1 0000 1110 = 0000 1110 + 0001 1011= 0001 0101**
- **6E.03=6E+6E.02=0110 1110+0 1101 1100 = 1011 0010**
- **Total= 0001 0101+1011 0010 + 0100 0110 +1010 0110 = 0100 0111**
-

AES

a00	a01	**a02**	a03	ao4	a05
a10	a11	**a12**	a13	a14	a15
a20	a21	**a22**	a23	a24	a25
a30	a31	**a32**	a33	a34	a35



b00	b01	**b02**	b03	bo4	b05
b10	b11	**b12**	b13	b14	b15
a20	b21	**b22**	b23	b24	b25
b30	b31	**b32**	b33	b34	b35

AES

- Add Round key
- Bit wise “exclusive or” of the complete block.
- Simple operation

AES

- Initial Round key addition
- Total key words 44.
- Initial add round key+ 10 round keys
- How to generate all round key words from 128 bit (4 word) basic key?

AES Key Schedule

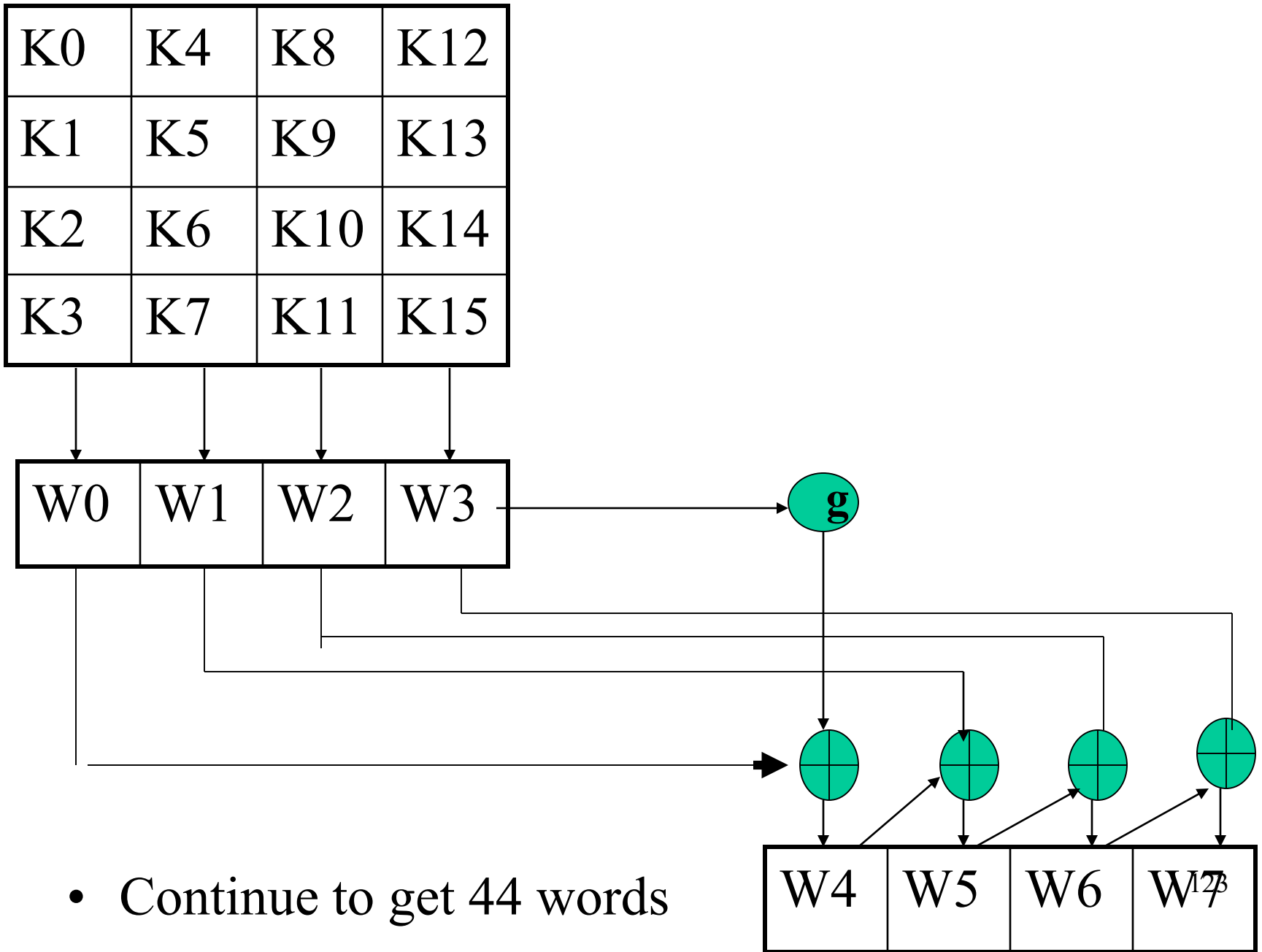
- We need 44 number of 32 bit words W
- First four words are given key data itself
- Temp= w(i-1)
- W(i) = temp *Exor* W(i-4) for all *i* except multiples of 4
- For i= multiples of 4, temp = subbyte (rotbyte (temp)) exor Rcon(i/4);
- Rot byte is one byte circular left shift of the word

AES

- Rcon is a word with three Least significant bytes zero. Most significant byte is as per table.

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RC(j)	**01**	**02**	**04**	**08**	**10**	**20**	**40**	**80**	**1B**	**36**

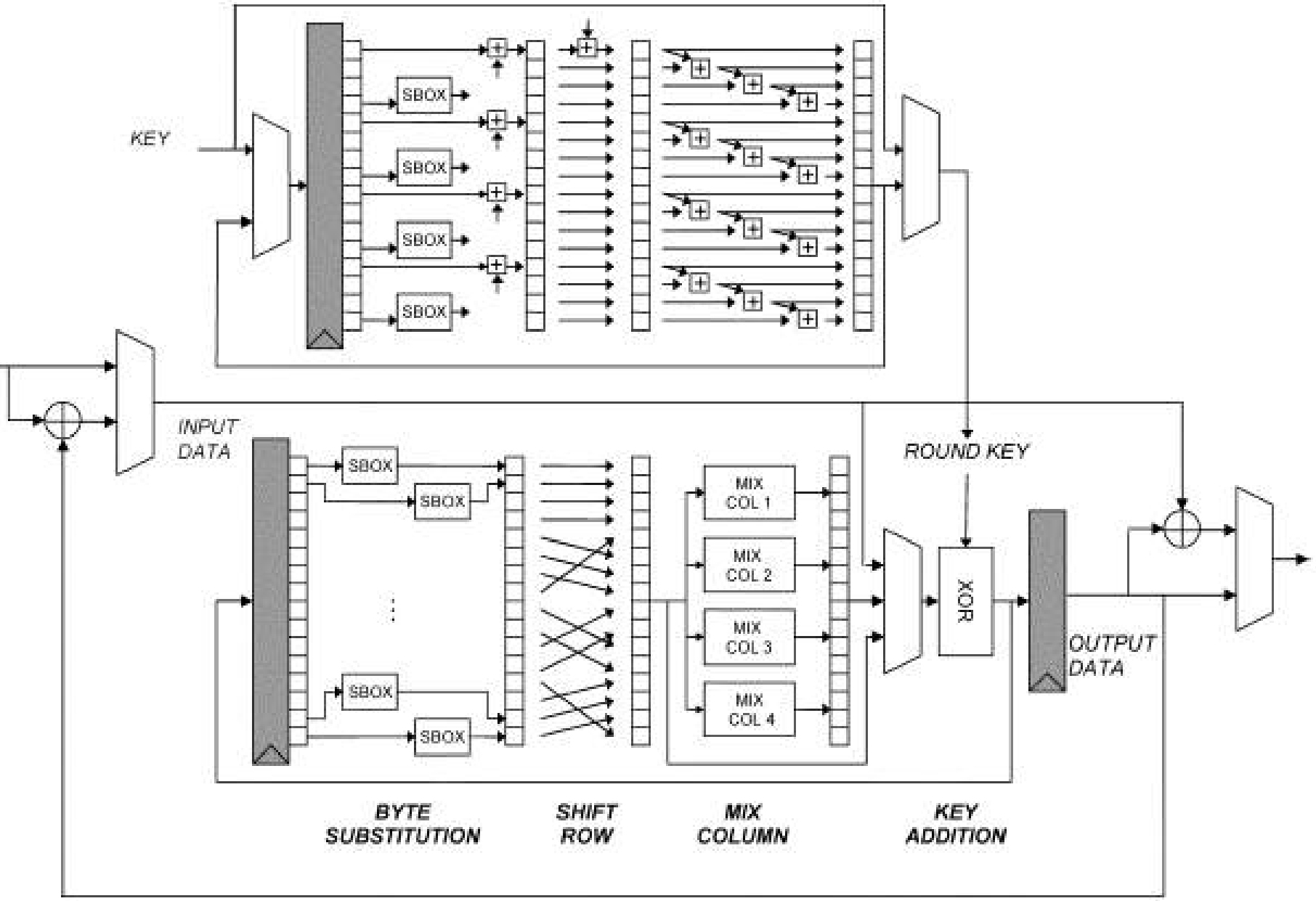
Key Generation method



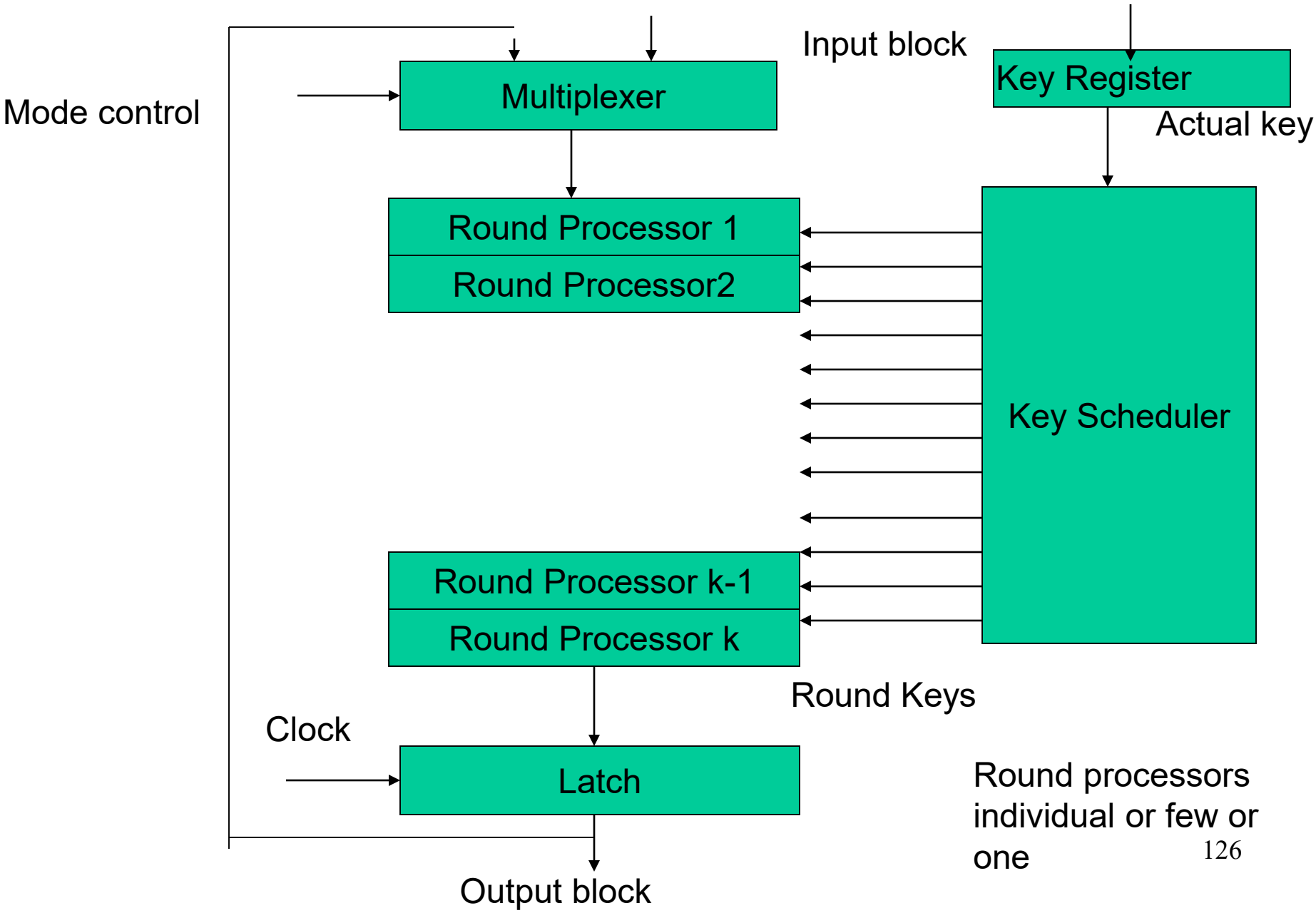
- Continue to get 44 words

AES

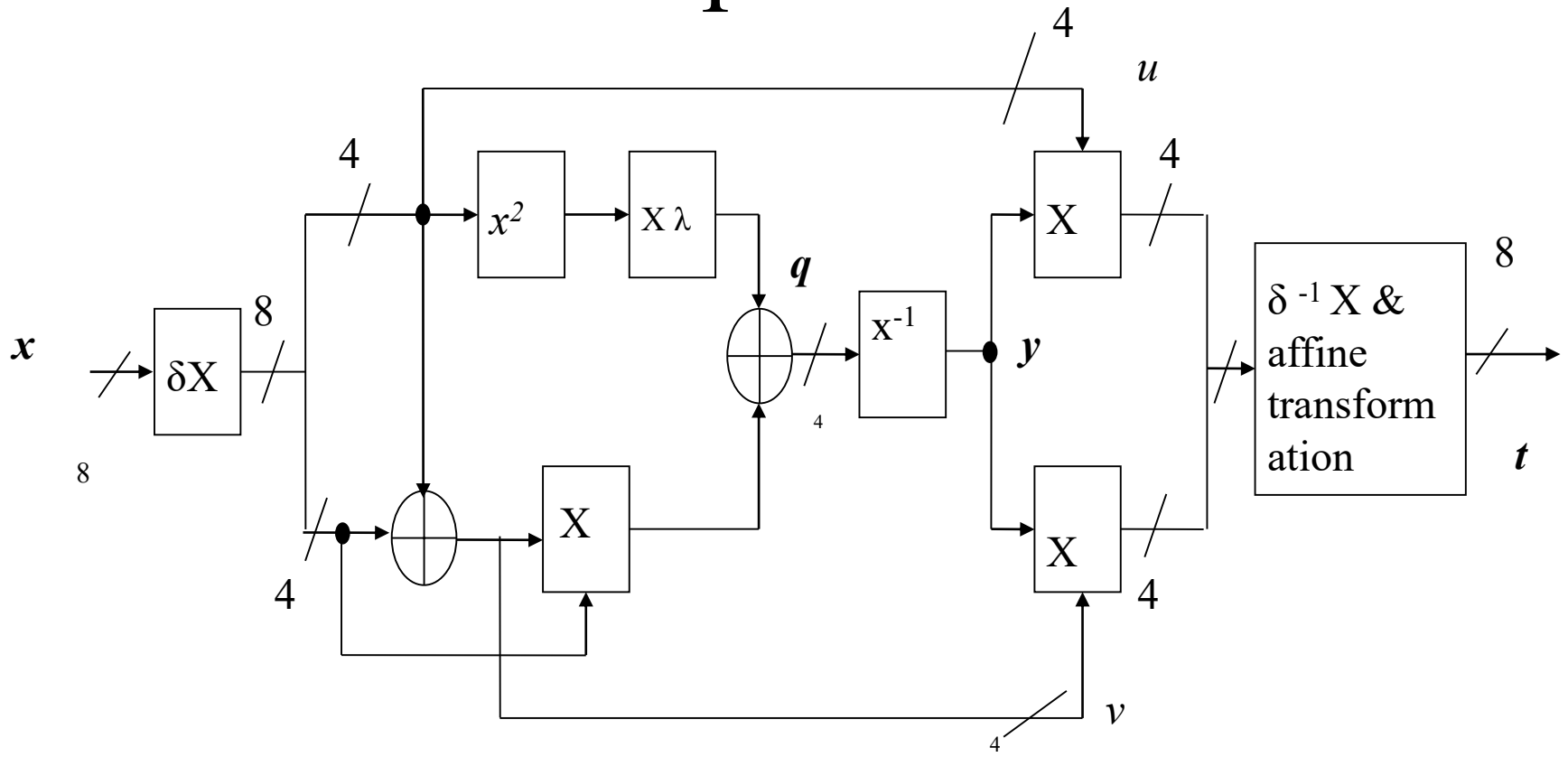
- Decryption involves reverse steps
- Can be implemented on 8 bit or 32 bit processors.
- There are lots of Hardware related seminars to improve the speed of implementation.
- Cryptanalysis is under progress all over the world.


KEY
SBOX
SBOX
SBOX
SBOX
INPUT
DATA
ROUND KEY
SBOX
SBOX
SBOX
SBOX
MIX
COL 1
MIX
COL 2
MIX
COL 3
MIX
COL 4
XOR
OUTPUT
DATA
BYTE
SUBSTITUTION
SHIFT
ROW
MIX
COLUMN
KEY
ADDITION

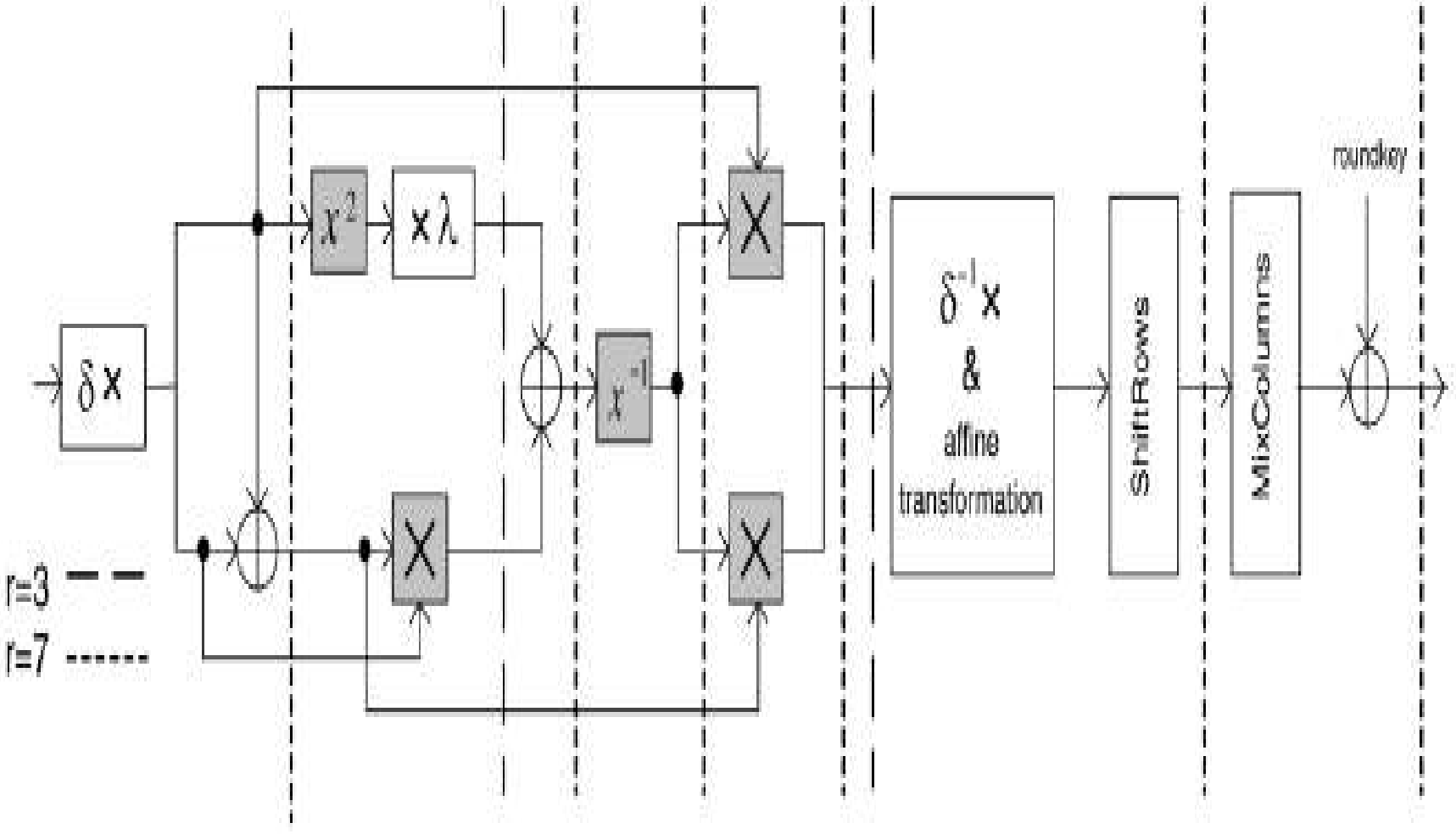
Typical Architecture ASIC or FPGA



S-Box implementation



(a)



Pipeline cuts at various places to equalize the delay

Comparison of stream and Block Ciphers

Criteria	Stream Cipher	Block Cipher
Error Propagation	No	Yes (not in stream cipher mode)
Period	Can be very large	$<2^{128}$ for a 128 bit block
Strength Evaluation Measures	Linear Complexity	Other methods Brute Force
Attacks	Correlation, Guess and determine, Table based, Algebraic, Berlekamp-Massey Algorithm etc	Differential and Linear Cryptanalysis, thorough search
Speed	Very High 2Gb/s even	Gigabits/sec
Ease of Implementation	FPGAs, DSPs, ASICs, Few lines of code	Suitable for DSPs, FPGAs,ASICs

Block cipher can be implemented as a stream cipher

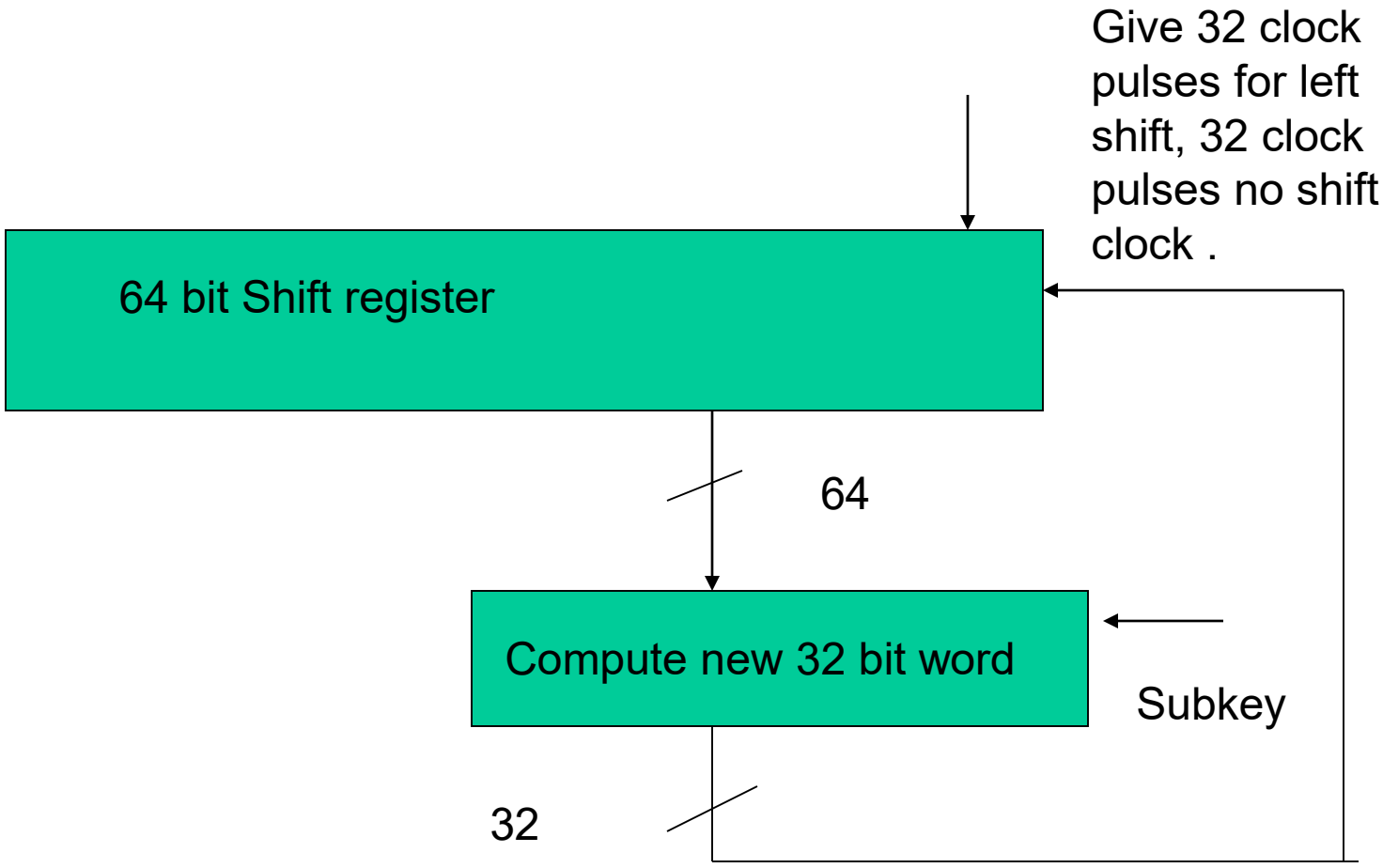
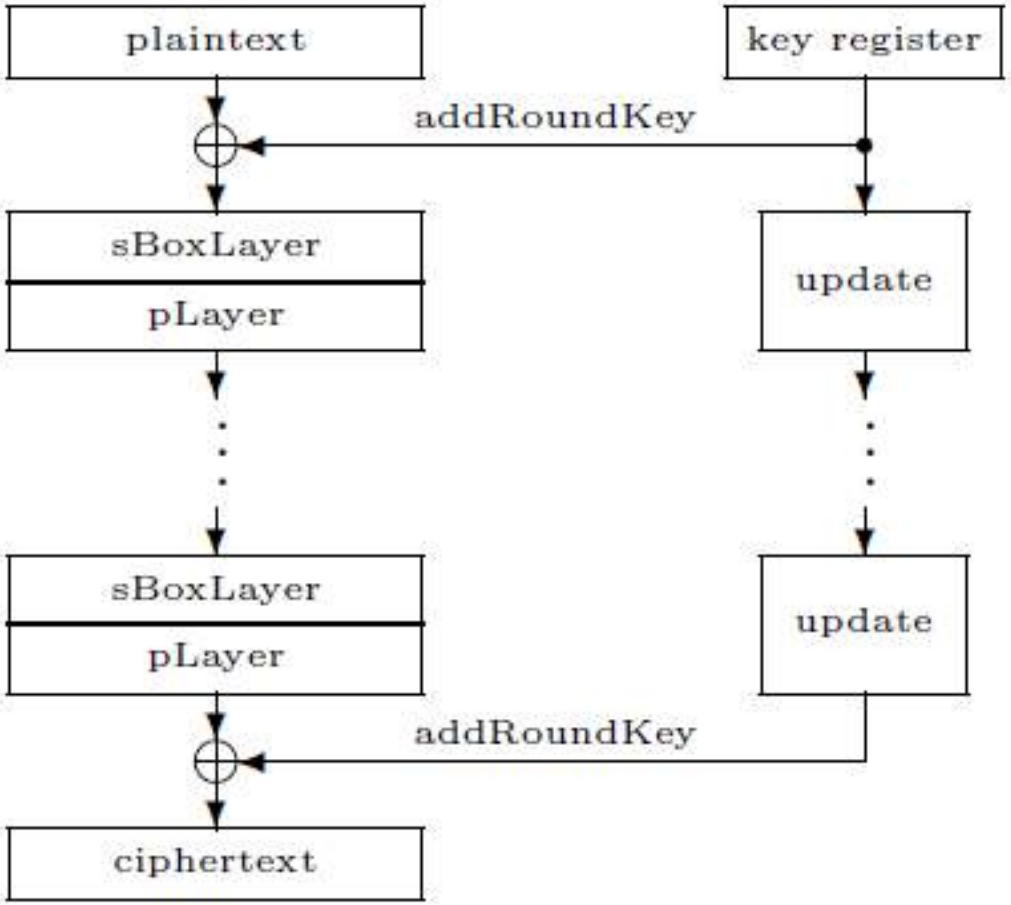


TABLE I

SIMULATION RESULTS AND COMPARISON BETWEEN DIFFERENT IMPLEMENTATIONS

Design	Device	Frequency (MHz)	Throughput (Mbps)	Slices	BRAMs	Efficiency (Mbps/Area)
Jarvinen et al.[5]	XCV-1000e-8	129.2	16500	11719	0	1.408
Zhang and Pahri [2]	XCV-1000e-8	168.4	21556	11022	0	1.956
This work (p=5)	XCV1000e-8	168.3	21542	9104	0	2.366
Hodjat and Verbauwhde [12]	XC2VP20-7	169.1	21645	9446	0	2.291
This work (p=5)	XC2VP20-7	220.7	28250	9028	0	3.129
Zambreno et al. [11]	XC2V4000	184.2	23572	16938	0	1.392
This work (p=5)	XC2V4000-6	211.6	27087	8503	0	3.186
Granado-Criado et al [3]	XC2V6000-6	194.7	24920	3576	80	1.804
This work (p=5)	XC2V6000-6	218.3	27942	7884	0	3.544
This work (p=9)	XC2V6000-6	305.1	39053	10662	0	3.663

PRESENT Block Cipher



PRESENT Block Cipher

sBoxlayer. The S-box used in PRESENT is a 4-bit to 4-bit S-box $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. The action of this box in hexadecimal notation is given by the following table.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

For sBoxLayer the current STATE $b_{63} \ldots b_0$ is considered as sixteen 4-bit words $w_{15} \ldots w_0$ where $w_i = b_{4*i+3}||b_{4*i+2}||b_{4*i+1}||b_{4*i}$ for $0 \leq i \leq 15$ and the output nibble $S[w_i]$ provides the updated state values in the obvious way.

pLayer. The bit permutation used in PRESENT is given by the following table. Bit i of STATE is moved to bit position $P(i)$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

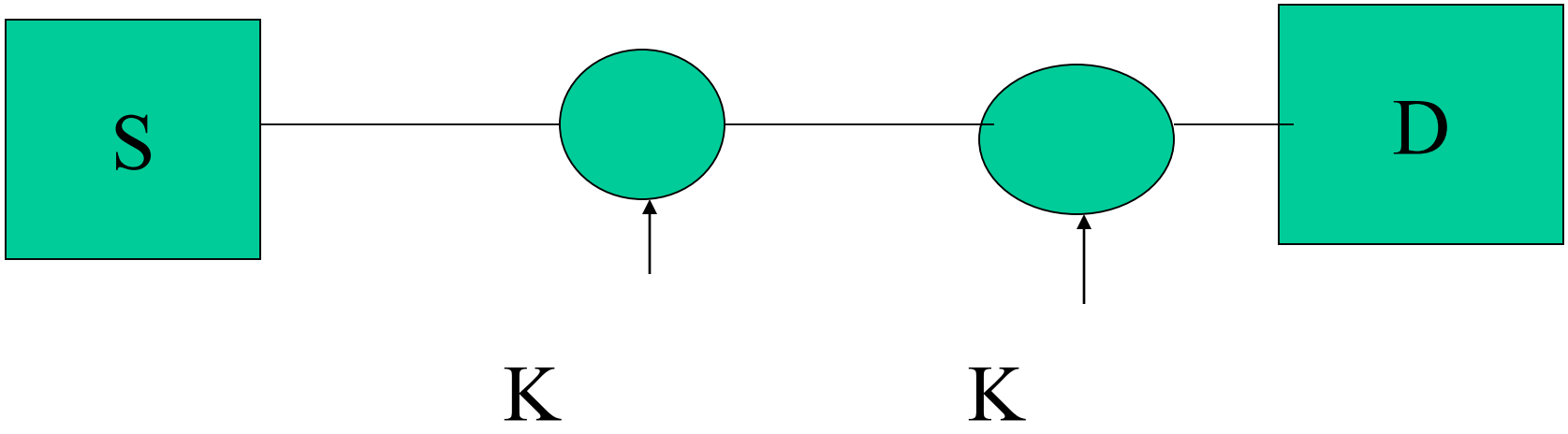
- 31 Rounds, Key 80 or 18 bits

Algebraic model of a 3-bit S-Box - 9 equations

$$
\left[\begin{array}{l|l}
1 & 11111111 \\
x_0 & 01010101 \\
x_1 & 00110011 \\
x_2 & 00001111 \\
y_0 & 10000111 \\
y_1 & 11001001 \\
y_2 & 11010100 \\
x_0y_0 & 00000101 \\
x_0y_1 & 01000001 \\
x_0y_2 & 01010100 \\
x_1y_0 & 00000011 \\
x_1y_1 & 00000001 \\
x_1y_2 & 00010000 \\
x_2y_0 & 00000111 \\
x_2y_1 & 00001001 \\
x_2y_2 & 00000100
\end{array}\right]
\rightarrow
\left[\begin{array}{l|l}
1 & 11111111 \\
x_0 & 01010101 \\
x_1 & 00110011 \\
1+x_0+x_1+y_0 & 00011110 \\
x_2 & 00001111 \\
1+x_1+y_1 & 00000101 \\
1+x_0+x_1+y_0+y_1+y_2 & 00000011 \\
x_0+x_0y_2 & 00000001 \\
\hline
x_2+y_0+y_1+x_0y_1 & 00000000 \\
1+x_1+y_1+x_0y_0 & 00000000 \\
1+x_0+x_1+y_0+y_1+y_2+x_1y_0 & 00000000 \\
x_0+x_0y_2+x_1y_1 & 00000000 \\
1+x_1+x_2+y_0+x_0y_2+x_1y_2 & 00000000 \\
y_0+y_2+x_0y_2+x_2y_0 & 00000000 \\
x_0+x_2+y_0+y_2+x_2y_1 & 00000000 \\
1+x_0+x_1+y_1+x_0y_2+x_2y_2 & 00000000
\end{array}\right]
$$

Authentication algorithms

Encryption and authentication

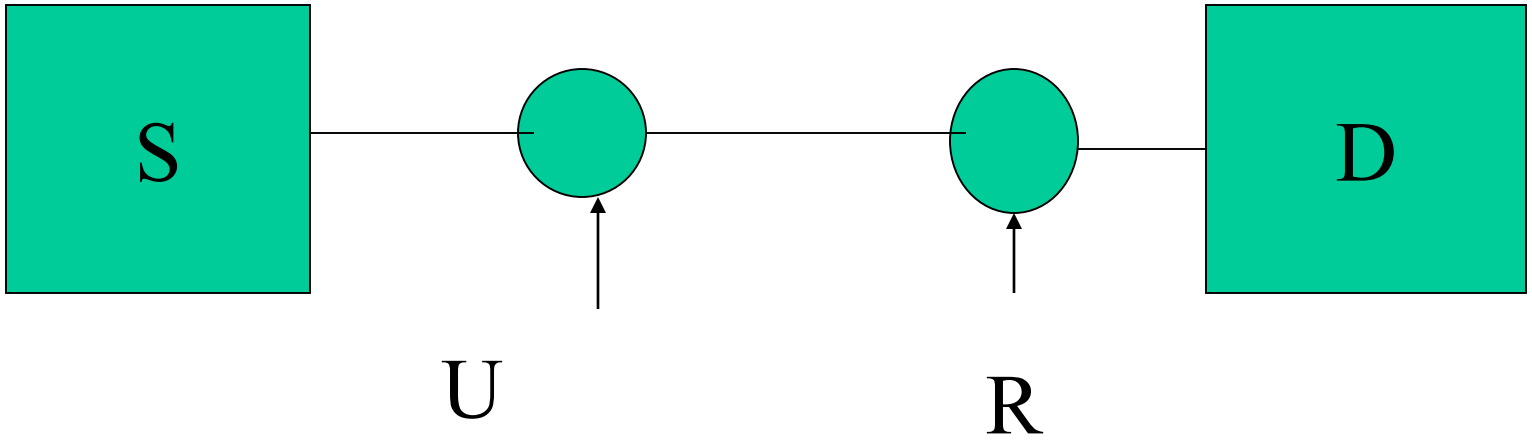


CONVENTIONAL ENCRYPTION

Encryption and authentication

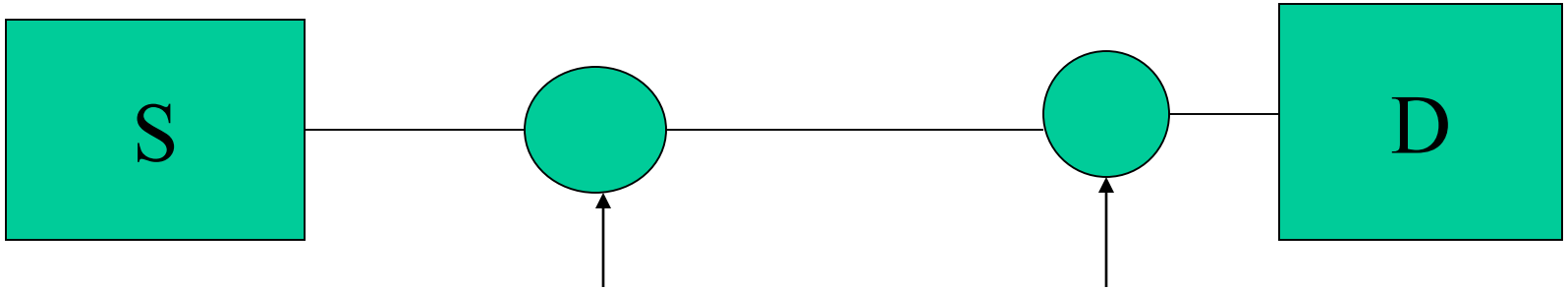
U stands for Public

R stands for Private



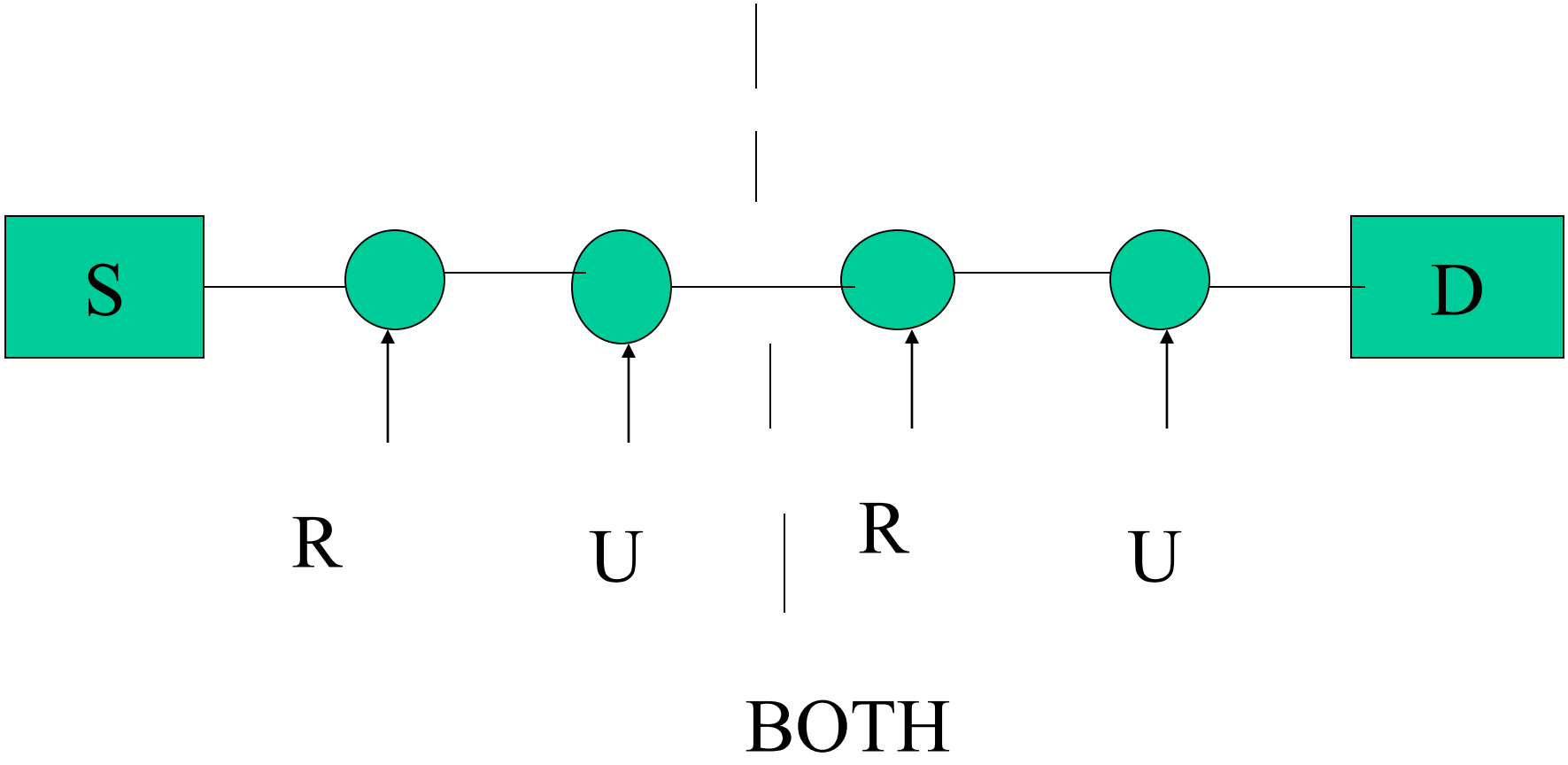
CONFIDENTIALITY

Encryption and authentication



AUTHENTICATION

Encryption and authentication



Popular method

- User generates a key pair himself
- Supplies public key with name etc to ca
- CA issues certificate for a fee
- User keeps private key himself
- Periodic renewal
- Cost of having software

Authentication using RSA

- RSA (Rivest- Shamir- Adleman) inventors
- Two keys are used (public key and private key)

Authentication using RSA

- m = message
- Public Key = (e,n)
- Private Key = (d,n)
- Encryption $c = m^e \bmod n$
- Decryption $m = c^d \bmod n$

Choice of n ,e,d

- Choose two large primes p and q.
- $n = p \times q$
- Choose e such that e and $(p-1) \times (q-1)$ are relatively prime.
- Calculate d so that
  $e \times d = 1 \bmod ((p-1) \times (q-1))$

Example

- p = 47,q = 71
- (p-1) ×(q-1) = 46 ×70=3220
- choose e = 79
- then d = 1019.
- m = 688 say
- c = 1570 and m = 688 after decryption

How to compute X^Y mod N

- X,Y and N are 1024 bit numbers typically.
- Repeated squaring and conditional multiplications
- 11^{23} mod 37 = ($11^{16}.11^4.11^2.11$) mod 37
- Basic operation is A.B mod N
- X^Y mod N needs 2047 such operations at most for 1024 bit numbers

Reducing squarings

- Steps can be reduced by precomputation: X^5, X^7
- Example X^{11749}
- 11749 = (2DE5)HEX = (101)(101)(111) 00101
- Whenever bit string with both sides 1 then raise previous result to power of 8 multiply by present value. If such 3 bit string does not occur keep squaring
- $((((2^5)^8x2^5)^8x2^7)^8)^2)^2x2^5$

Authentication algorithms

Basic Algorithms needed

- Modulo addition/subtraction (A+B) mod N or (A-B) mod N
- Modulo Multiplication (AB) modN
- Exponentiation XY mod N
- Finding Multiplicative Inverse
- Primality Testing
- Random number generation

Modulo Addition/Subtraction

- Addition Z = A+B
- Find Z - M = (A+B)-M
- If Z-M is positive Result = Z-M; else Result = Z
- E.g. (10+13) mod 17; Z= 23, Z-M = 6; Hence Ans = 6.
- Subtraction Z = (A-B)
- Find Z+M =(A-B)+M
- If Z is negative, Result =Z+M else Result = Z
- Example (10-13) mod 17; Z=-3, Z+M = 14, Ans = 14.

Table 4.4 Finding the Multiplicative Inverse of 550 in GF(1759)

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

How to compute X^Y mod N

- X,Y and n are 1024 bit numbers typically.
- Repeated squaring and conditional multiplications
- 11^{23} mod 37 = ($11^{16}.11^4.11^2.11$) mod 37
- Basic operation is A.B mod N
- X^Y mod N needs 2047 such operations at most for 1024 bit numbers
- Exponentiation can left to right or right to left scanning or faster methods like windowing

Exponentiation

- A^{13} is desired say.
- 13 is 1101 = $b_3b_2b_1b_0$
- Left to right. Start with b_3 =1.
- A
- Square old value multiply with A if b_2 =1 else new value = old value.
- We get $(A)^2.A = A^3$.
- b_1 = 0. Square and do not multiply = A^6
- b_0 = 1. Square and multiply = $(A^6)^2 \times A = A^{13}$
- We need three squarings and three multiplications in the worst case.

Window Method of RSA exponentiation

- Exponent 11749 = (101)(101)(111)00(101)
- Precompute A^5, A^7
- Scan from left three bit words with 1 at LSB and MSB
- (A^5) , $(A^5)^8 = A^{40}$
- $A^{40} x A^5 = A^{45}$
- $(A^{45})^8 = A^{360}$
- $A^{360} x A^7 = A^{367}$
- $(A^{367})^2 = A^{734}$ since no string bounded by ones exists
- $(A^{734})^2 = A^{1468}$ since no string bounded by ones exists
- $(A^{1468})^8 x A^5 = A^{11749}$.
- 11 squarings and 3 Multiplications vis a vis 13squarings and 8 multiplications

How to compute (A.B) mod N

- Example: 13.15 mod 23
- We do not want to do in a straight forward manner .
- Write b = 13 in binary form : 1101
- Do repeatedly starting from msb: (2.Old + b_i.A) mod 23

Example modulo multiplication

- **(2.0 + 1.15) mod 23 = 15**
- **(2.15 + 1.15) mod 23 = 22**
- **(2.22 + 0.15) mod 23 = 21**
- **(2.21+ 1.15) mod 23 = 11**
- **Maximum size of p <3.23 i.E 3N**
- **Modulo subtraction is by two comparisons: p<n?,N<p<2n?P>2n?**
- **Answer is p,p-n,p-2n respectively.**
- **Multiple precision arithmetic to be used in pc based implementaions**

Montgomery's Modular Multiplication

- Easier and faster to implement than conventional exponentiation needed for RSA
- Uses scaled inputs and gives scaled intermediate outputs
- Pre-scaling and post-scaling only once in the beginning and once in the end are needed.

Montgomery's method

- X = A.B mod M desired
- First define P = (A.2^n) mod M and Q = (B.2^n)mod M
- Find Using Montgomery technique, Z = (P.Q) 2^{-n} mod M
- i.e. we found Z = (A.B)2^{-n} mod M
- X can therefore be found by performing post-scaling X = Z.2^n mod M

Example (A.B) mod M

- A = 11, B = 13 and M = 19 and n = 4
- *Step1:* Scale A and B by 2^n mod M = 2^4 mod 19.
- Hence, X = A.2^n mod M = 11.2^4 mod 19 = 5, and Y = B.2^n mod M = 13.24 mod 19 = 18.
- *Step2:* Find Z = (X.Y 2^{-n}) mod M = (5.18.2^{-4}) mod 19 = 8
- *Step3:* (A.B) mod 19 = (Z.2^{-n}) mod M = (8.2^{-4}) mod19 = 10. which can be verified to be true.

Advantages

- No division or modulo reduction needed at intermediate steps unlike in conventional modulo multiplication.
- Comprises of n steps
- Each step after adding the partial product, division by two is done to ultimately realize the multiplication by 2^{-n} mod M
- e.g (11/2) mod 19 is estimated as (11+19)/2 mod 19 = 15 since LSB of 11 is 1.If LSB is zero no addition of 19 is needed.

Advantage

- In each step a conditional addition is done and deleting the LSB of the result in contrast with modulo reduction needed in conventional method.

Example of $(A.B.2^{-n})$ modM

- A = 1011, B = 1101 = $b_3b_2b_1b_0$, we need to find $(A.B.2^{-4})$ mod 19.
- 11 = 1011 b_o= 1, Add 11 to the accumulator
- 1010x LSB is 1, hence add 10 = (19+1)/2 to the MSBs
- 1111x ignore the LSB
- 0000 b1 = 0, Add 0 to the accumulator
- 1111 LSB is 1, hence add 10 = (19+1)/2 to the MSBs
- 1010x
- 10001
- 1011 b2 = 1, Add 11 to the accumulator
- 11100 LSB is 0, Need not add any number
- 1110 Ignore the LSB
- 1011 b3 = 1, Add 11 to the accumulator
- 11001
- 1010x LSB is 1, Hence add 10 = (19+1)/2
- 10110 result;
- 0011 Take mod 19 to obtain the result as 3.

Front-end Overhead

- Pre Evaluation of $A.2^n \bmod M$
- Example $(11.2^4) \bmod 19$
- 11.2^4 = binary 10110000
- Start from LSB find the residue mod 19 of each power of two
- (0+0+0+0+16+13+0+14)mod 19 = 5
- Verify
- Needs successive doubling of current residue of power of two and modulo reduction (subtraction of 19)

High Radix Montgomery Multiplication

- Take more bits at a time.
- Example (56x73)/100 mod 97
- Montgomery constant =100.
- In two steps we can do. Each step we divide by 10 mod 97.
- Find (-1/97) mod 10 first = (-1/7) mod 10 = 7
- First step Multiply with 3
- 3x56=168, Find (168x7) mod 10 = 6.
- Find 168+ 6X97 = 750, Divide by 10 = 75.
- Seco step multiply with 7 and accumulate : 7X 56+75 = 467
- Find (467x7) mod 10= 9
- Add 467+9X97 = 1340. Divide by 10 = 134 mod 97 = 37.
- Verify (56X73)/100 mod 97 = 37.
- Today we do word bssed Montgomery mutiplication. (16 bit words).

Primality testing

- How to find primes?
- Choose an odd number of required number of bits and test for primality.
- Use Miller-Rabin method:
- Express p-1= $2^k.q$
- Choose a base a.
- Find z = a^q mod p.
- If z = 1 or (p-1), this may be a prime.
- Find z^2 mod p ,z^4 mod p,..$z^{2\wedge k}$ mod p.
- if any of these is (p-1), test is inclusive (may be p is prime)
- Repeat for other bases.

- Before Rabin's test, divide by all primes less than 256 so as not to waste time.

Example1

- $p = 29$; $p-1=2^2.7$ i.e. $k=2$, $q = 7$
- Choose base 2. Find $2^7 \bmod 29 = 12$
- $(12)^2 \bmod 29 = 28$
- Inconclusive may be prime.
- Repeat for other bases 3,5 etc.

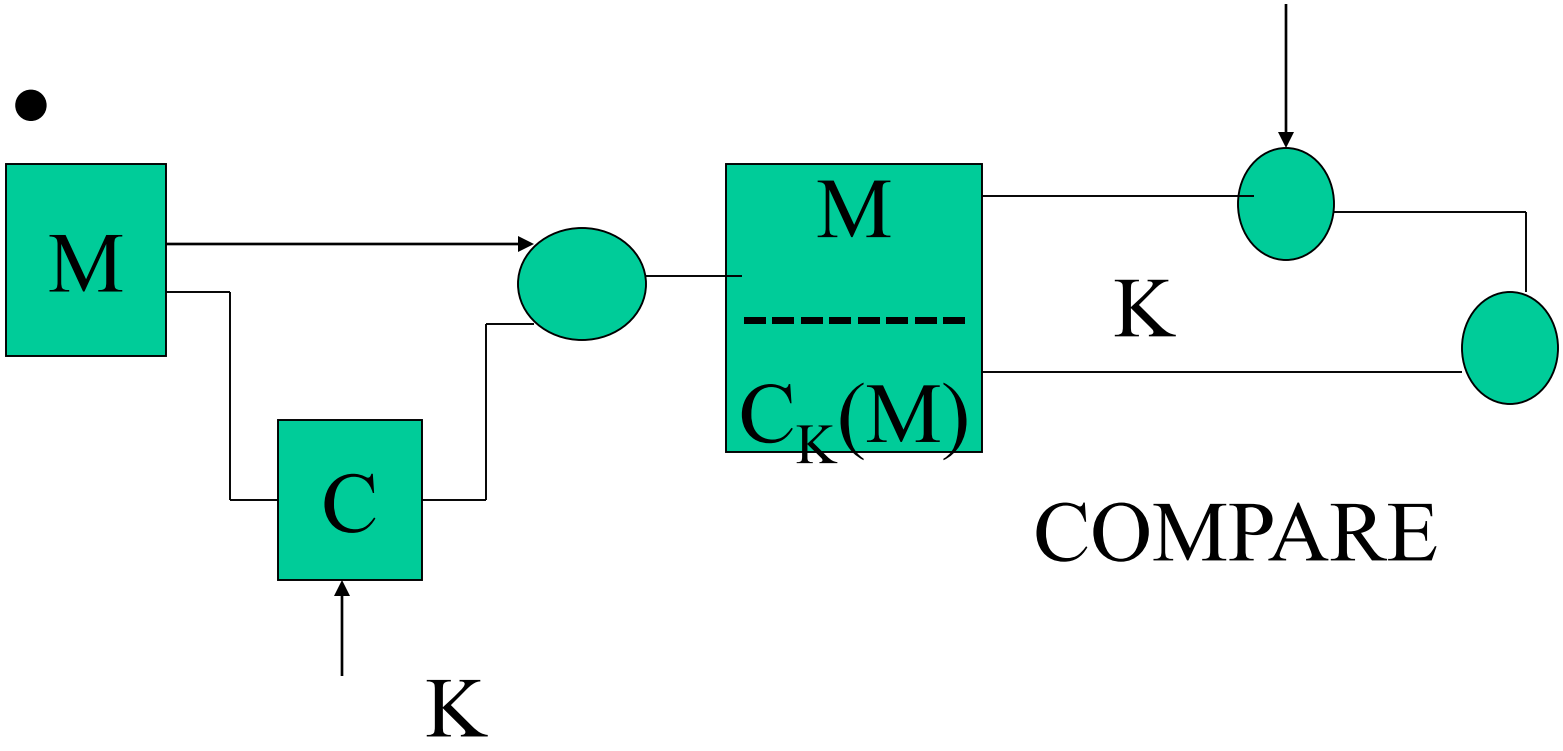
Example2

- p=composite= 221= 13.17
- p-1 = $55x2^2$; q=55, k=2
- Choose base 5
- $5^{55} \bmod 221 = 112$
- $(5^{55})^2 \bmod 221 = 168$
- We have exhausted k=0,1
- Hence p is composite.

Hashing Algorithms

Digital signature algorithms

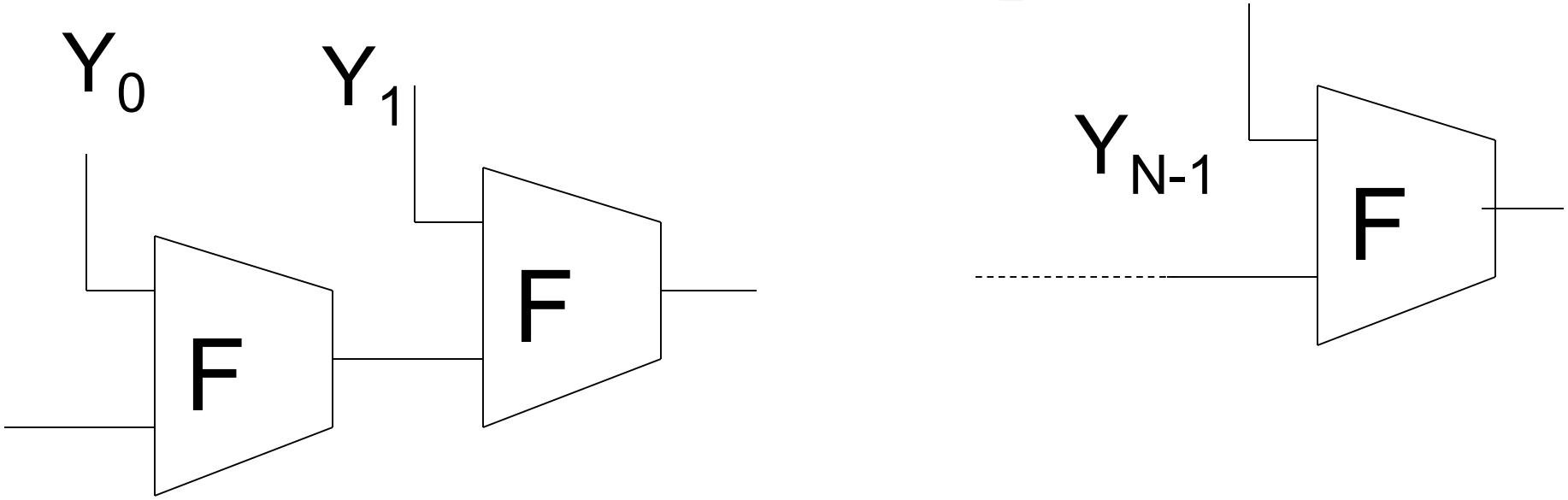
Authentication by digital signatures



Hashing

- one way functions are used:given x easy to compute f(x) but reverse is difficult to compute
- Shall be easy to compute hash but difficult to find an image that hashes to that desired value
- Checksum is not a good hash function since message can be easily forged which hashes
- Hash function shall be collision free (no two images hash to the same value)

General Principle



IV

- F is a compression function
- Y_i are successive blocks in the input
- If F is collision resistant, so is the Hash algorithm.

- *Pre-image resistance* Given a hash h it should be difficult to find any message m such that $h = hash(m)$. This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks.
- *Second pre-image resistance* Given an input m_1 it should be difficult to find another input m_2 such that $m_1 \neq m_2$ and $hash(m_1) = hash(m_2)$. Functions that lack this property are vulnerable to second-preimage attacks.
- *Collision resistance* It should be difficult to find two different messages m_1 and m_2 such that $hash(m_1) = hash(m_2)$. Such a pair is called a cryptographic hash collision. This property is sometimes referred to as *strong collision resistance.* It requires a hash value at least twice as long as that required for preimage-resistance; otherwise collisions may be found by a birthday attack.

MD5

- **128 bit hash is generated**
- **Processes input text of blocks of 512 bits.**
- **Preprocess text such that last block has only 448 bits. If less append a string of 1 followed by zeroes.**
- **The last 64 bits in the last block is the length of the message.**
- **Every bit of hash code is a function of all input bits**
- **Complex repetition of logic functions F,G,H,I produces results that are well mixed.**

MD5

- **Uses four 32 bit constants A,B,C,D called chaining variables: A=0x01234567,B=0x89abcdef, C=0xfedcba98 and D=0x76543210**
- **Four rounds each uses different operation16 times.**
- **Each operation is a nonlinear function on three of a,b,c d and adds the result to fourth and a constant. Result replaces a,b,c,d.**

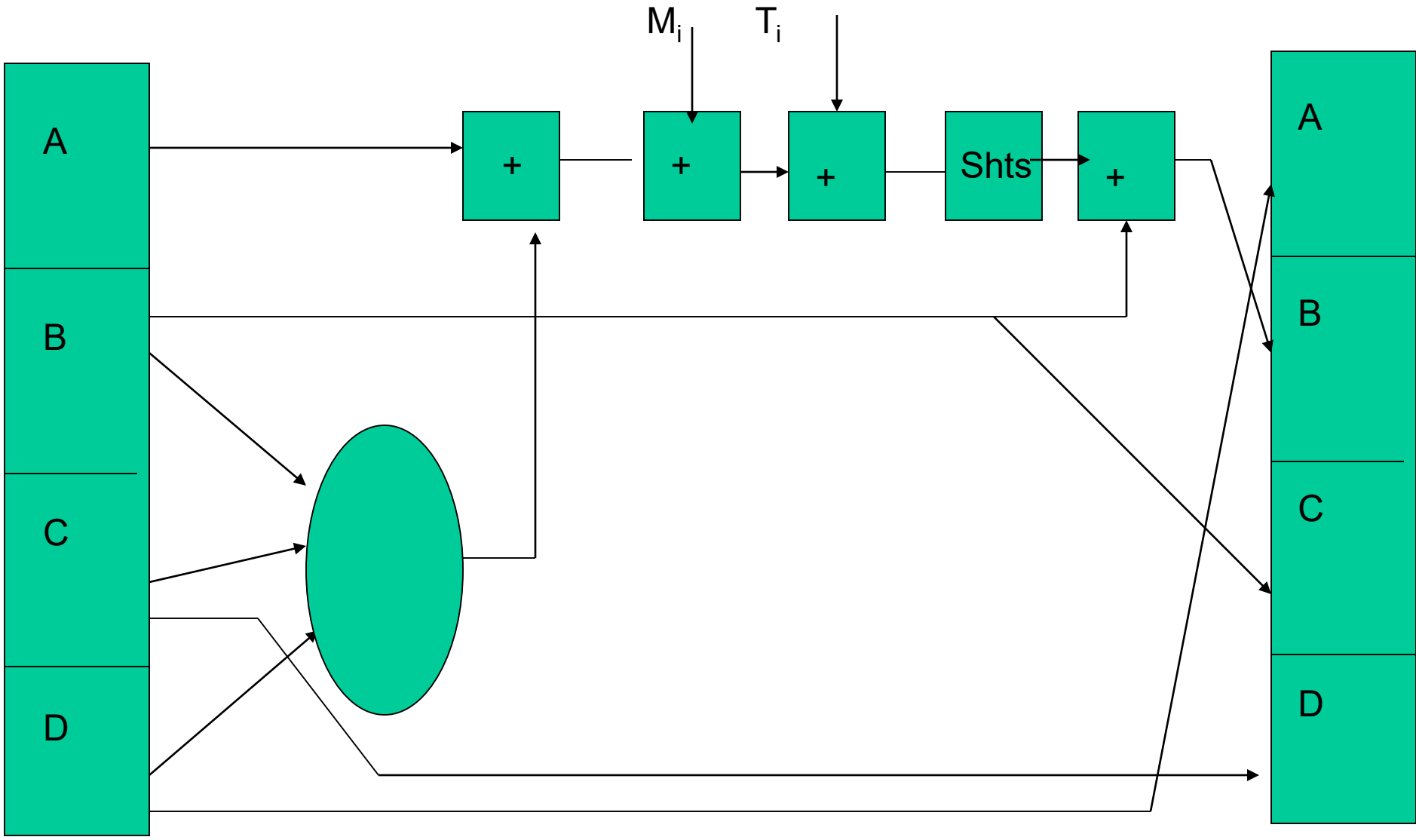
MD5

- Non-linear functions
- F: *If X then Y, else Z (Round 1)*
- G: *If X then Z else Y (Round 2)*
- H: Parity of X,Y,Z (exclusive or bitwise) *(Round 3)*
- I: *Y exor (X or not Z) (Round 4)*
- 64 constants t_i used one in each of the sixteen steps in four rounds
- $T_i = int\ (2^{32}.abs(sini))$. i is in radians.

MD5

- Typical operation
- FF(a, b, c, d, M_i, s, t_i) is
  $a = b+((a+F(b,c,d)+M_i+t_i)<<<s)$
- Each step adds to the result of previous step (Avalanche effect)
- Shifts are different in each round.
- Shifts also add to avalanche effect.

MD5



MD5

- Order of Using input words is different in each round.
- First Round is normal 1 to 16.
- order in 2nd round $\rho_i(t) = (1+5i) \bmod 16$
- order in 3rd round $\rho_i(t) = (5+3i) \bmod 16$
- order in 4th round $\rho_i(t) = 7i \bmod 16$

Example of Hash Collision MD5

```
d131dd02c5e6eec4 693d9a0698aff95c 2fcab58712467eab 4004583eb8fb7f89
55ad340609f4b302 83e488832571415a 085125e8f7cdc99f d91dbdf280373c5b
d8823e3156348f5b ae6dacd436c919c6 dd53e2b487da03fd 02396306d248cda0
e99f33420f577ee8 ce54b67080a80d1e c69821bcb6a88393 96f9652b6ff72a70

d131dd02c5e6eec4 693d9a0698aff95c 2fcab50712467eab 4004583eb8fb7f89
55ad340609f4b302 83e4888325f1415a 085125e8f7cdc99f d91dbd7280373c5b
d8823e3156348f5b ae6dacd436c919c6 dd53e23487da03fd 02396306d248cda0
e99f33420f577ee8 ce54b67080280d1e c69821bcb6a88393 96f965ab6ff72a70
```

Both produce the MD5 hash 79054025255fb1a26e4bc422aef54eb4.

Secure Hash Algorithm

- Treats messages as 512 bit blocks
- Four rounds of 20 operations each
- Five Constants 32 bit A, B, C, D, E
- Uses nonlinear operations involving AND, OR, EXCLUSIVE-OR
- Uses circular shifts
- Generates a hash of 160 bits. Improvement over MD5

SHA

- Uses 5 Initial values for A,B,C,D and E
- A = 67452301
- B = EFCDAB89
- C = 98BADCFE
- D = 10325476
- E = C3D2E1F0

SHA

- Preprocessing same as in MD5 to make the input 512 bit words.
- Four rounds
- 20 steps in each round as against 16 steps in MD5
- 4 constants one for 20 steps as against one in each step
- K_1= 5A827999 $(2^{30}/\sqrt{2})$
- K_2= 6ED9EBA1 $(2^{30}/\sqrt{3})$
- K_3= 8F1BBCDC $(2^{30}/\sqrt{5})$
- K_4= CA62C1D6 $(2^{30}/\sqrt{10})$

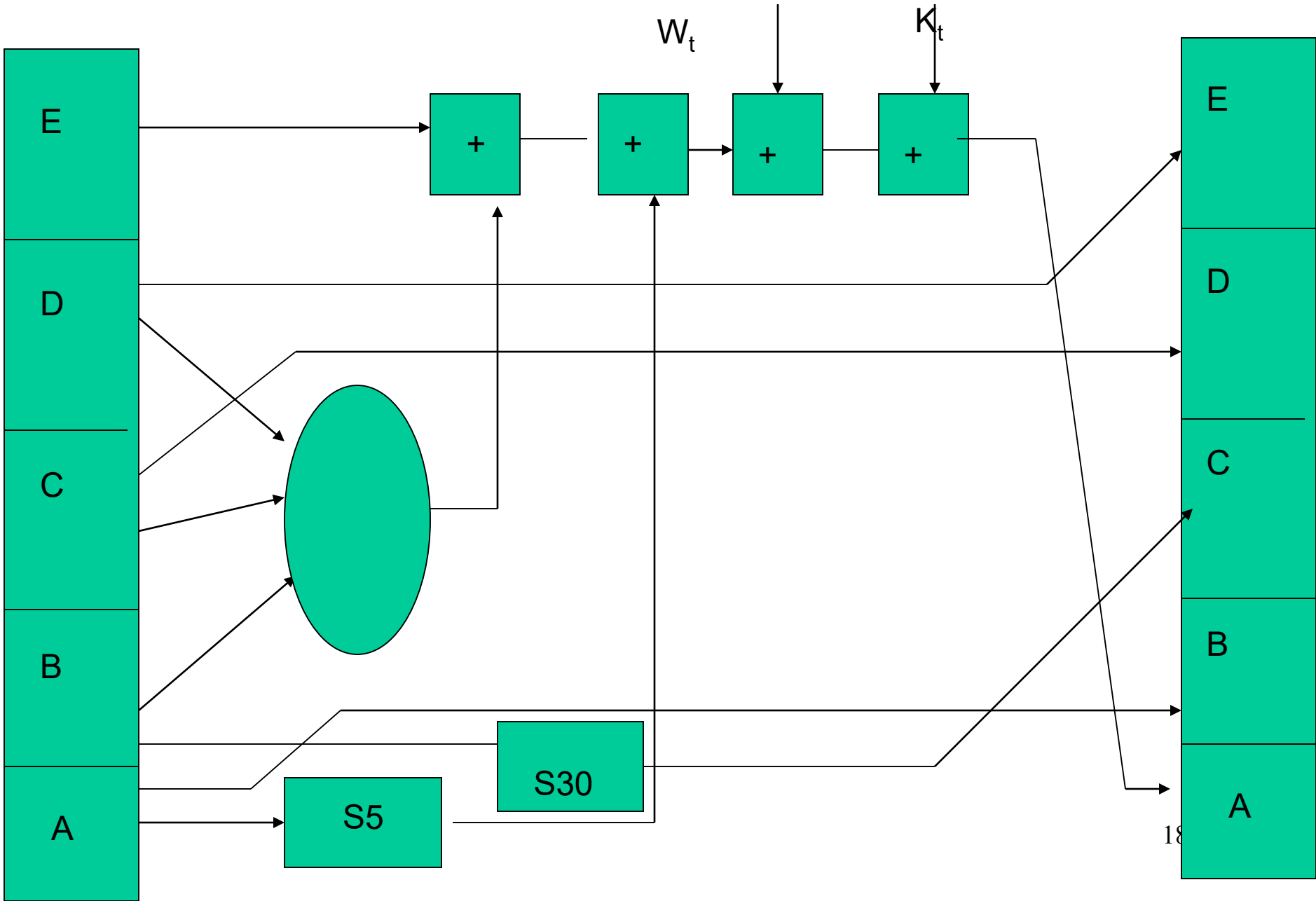
SHA

- 4 functions used one for each 20 steps as against four in MD5
- (B.C) OR (B'.D) $0 \leq t \leq 19$
- B exor C exor D $20 \leq t \leq 39$
- (B.C) or (B.D) or (C.D) $40 \leq t \leq 59$
- B exor C exor D $60 \leq t \leq 79$

SHA Basic Operation

- A,B,C,D.E will become
  $E+f(t,B,C,D)+s^5(A) + W_t+K_t)$,
  $A,S^{30}(B),C,D$
- S stands for left circular shift
- $W_t = S^1(W_{t-16}+W_{t-14}+W_{t-8}+W_{t-3})$
- This yields all the remaining 64 words.
- Original 16 words are used as it is.

SHA Hashing step



SHA-3 *Keccak*

- State 5x5 array of 64 bit words = 25x64 = 1600bits
- Sponge Construction
- W = 2^l bits where $l = 6$.

SHA-3 Continued

- Each round 5 steps R = ι. χ. π. ρ. θ.
- Five operations: θ *theta* (compute parity of each column and EXOR into neighbours) , ρ *rho* (bitwise rotate the word by a triangular number a permutation) , π *Pi* a permutation of 25 words , χ *chi* (bit wise combining along rows) and ι *iota* (Exor a round constant)
- rc[t] = (x^t mod $x^8 + x^6 + x^5 + x^4 + 1$)mod x in GF(2)[x].

SHA-3 Continued

- Block size is 1152 bits Hash is 224 bits or 188 to 256 or 832 to 384 or 576 to 512.
- C = capacity = 2n where n is the hash length.
- Padding is needed to make 1152 bits as 1600 bits.
- 24 rounds
- R+c determines width of the permutation r+c = 1600, if n =224, c = 448 and hence r = 1152.

Theta

- Find parity of all words in column 0 namely 00, 01,02,03,04 in a column, Circular left shift by 1 bit and Exor the result with parity of column 3 and EXOR with all words in column 4 and put back in column 4.
- Repeat for all columns.

Pi

- Permutation rule a[j][2i+3j] = a[i][j]
- i<5, j<5
- Consider i=1, j=1;
- a[1][1] goes to a[1][5] or a[1][0]
- Consider i=3, j=4;
- a[3][4] goes to a[4][18] = a[4][3]

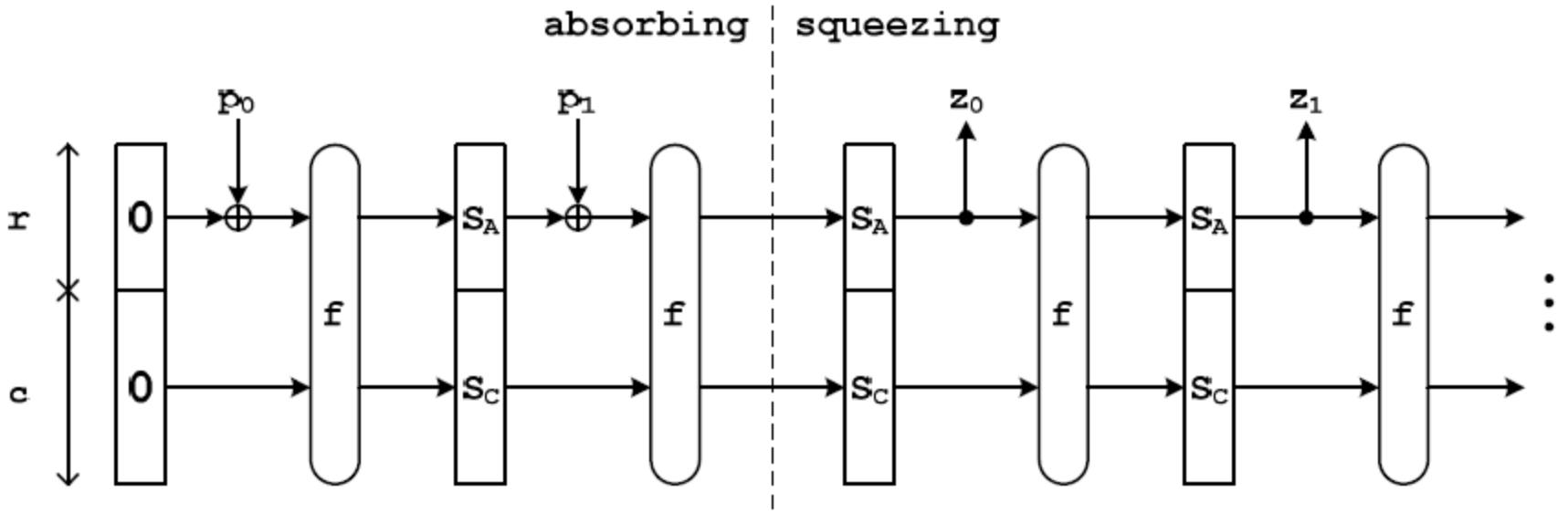
Rotation rho

- Complicated formula bit
  k becomes bit k-(t+1)(t+2)/2 for block [i][j]
- where $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 1 & 0 \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- Example t=1, $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ and k = k-3
- Hence [0][2] is left circular shifted by 3

Chi

- Nonlinearity is only in *Chi*
- $a = a\ exor(\neg b \& c)$
- where b is two words and c one word right of a.
- example words a =44 , b =14 and c = 34 yield new 44.

Keccak



Discoverers

- Bertoni, Daemen, Peeters and Assche

- Set seven permutations Keccak-f (b) where b = 25,50,100,200,400,800,1600
- Arranged as an array 5x5 lanes each of length w bits w =1,2,4,8,16,32,64
- b=25w
- Sponge function Keccak [r, c, d] obtained by construction Keccak-f [r.c]
- d is diversifier

- Number of rounds = 12+2xl
- Where $w = 2^l$
- *Thus, number of rounds is 12,14,16,18,20,22,24 for Keccak f[25].f[50],f[100],f[200],f[400],f[800],f[1600]*

Algorithm 1. Pseudo-code of *Keccak-f*

$Keccak-f[b](A)\{$

for i *in* $0 \ldots n_r - 1$

$\quad A = Round[b](A, RC[i])$

return A

$\}$

$Round[b](A, RC)\ \{$

θ *step*:

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4], \quad \forall x \text{ in } 0...4$$

$$D[x] = C[x-1] \oplus ROT(C[x+1],1), \quad \forall x \text{ in } 0...4$$

$$A[x,y] = A[x,y] \oplus D[x], \quad \forall (x, y) \text{ in } (0...4, 0...4)$$

ρ *and* π *steps*:

$$B[y, 2x+3y] = ROT(A[x,y], r[x,y]), \quad \forall (x, y) \text{ in } (0...4, 0...4)$$

χ *step*:

$$A[x,y] = B[x,y] \oplus ((NOT\ B[x+1,y])\ AND\ B[x+2,y]), \quad \forall (x, y) \text{ in } (0...4, 0...4)$$

ι *step*:

$$A[0,0] = A[0,0] \oplus RC$$

return A

$\}$

Algorithm 2. Pseudo-code of the sponge function *Keccak*[*r*,*c*,*d*]

$Keccak[r,c,d](M)\{$

Initialization and padding:

$S[x,y] = 0, \qquad \forall (x,y) \text{ in } (0...4, 0...4)$

$P = M \,||\, 0x01 \,||\, byte(d) \,||\, byte(r/8) \,||\, 0x01 \,||\, 0x00 \,||\, ... \,||\, 0x00$

Absorbing phase:

$\forall$ block P_i in P

$S[x,y] = S[x,y] \oplus P[x+5y], \qquad \forall (x,y) \text{ such that } (x+5y) < (r/w)$

$S = Keccak - f[r+c](S)$

Squeezing phase:

$Z = empty\ string$

while output is requested

$Z = Z \,||\, S[x,y], \qquad \forall (x,y) \text{ such that } (x+5y) < (r/w)$

$S = Keccak - f[r+c](S)$

return Z

}

In Algorithm 1, all of the operations on the indices are done in modulo 5. A denotes the complete permutation state array, and $A[x,y]$ denotes a particular lane in that state. $B[x,y]$, $C[x]$, $D[x]$ are intermediate variables, the constants $r[x,y]$ are the rotation offsets and $RC[i]$ are the round constants. $rot(w,r)$ is the bitwise cyclic shift operation which moves the bit from position i into position i+r, in the modulo lane size.

The pseudo-code of the sponge function $Keccak[r,c,d]$ is given in Algorithm 2, again with parameters capacity c, bit rate r and diversifier d. This description is restricted to the case of messages that span a whole number of bytes. For messages with a number of bits not dividable by 8, the details are given in [12]. In the algorithm, S denotes the state as an array of lanes. The padded message P is organized as an array of blocks P_i. The operator || denotes the byte string concatenation.

Input Padding

mode	output length	P	10*1
SHA-2 replacement	224	11001	10*1
SHA-2 replacement	256	11101	10*1
SHA-2 replacement	384	11001	10*1
SHA-2 replacement	512	11101	10*1
variable-length output	arbitrary	1111	10*1

- Basic operations XOR, AND, NOT, Rotate
- For 26 bit Keccak, Input message block is 1088 bits
- Extended input, exclusive ored with a chaining value
- Chaining value
- 24 rounds

Keccak–256 : r = 1088

Keccak–512 : r = 576

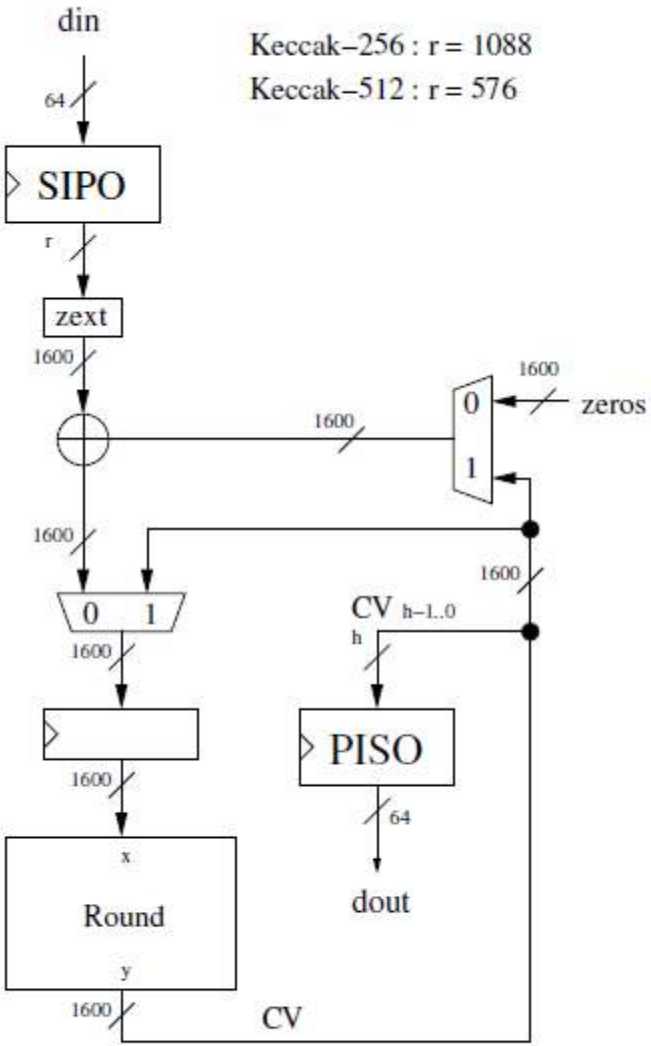
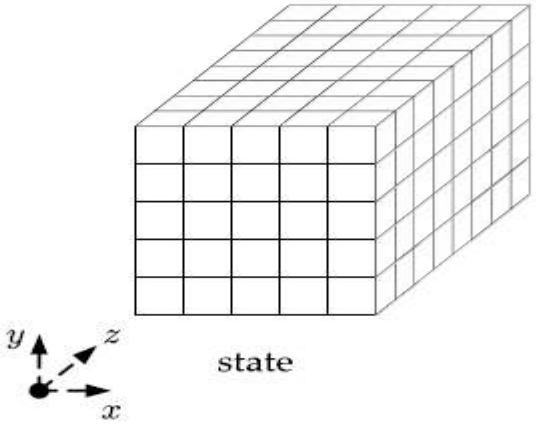
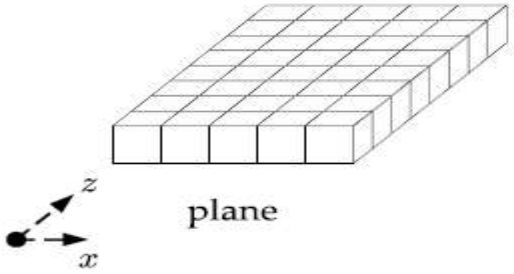


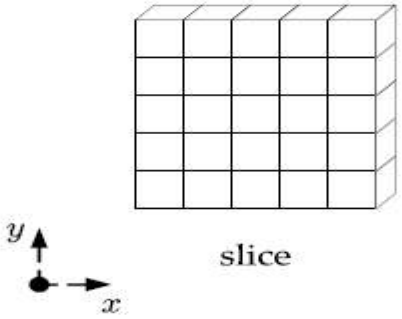
Figure 3.29: Keccak : Datapath



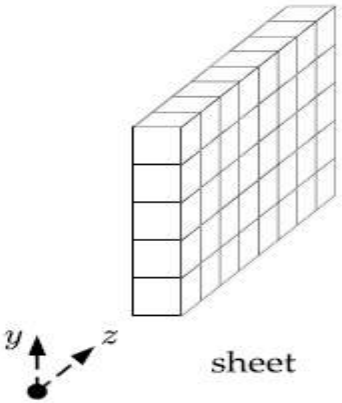
state



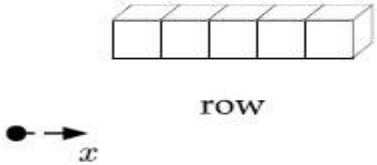
plane



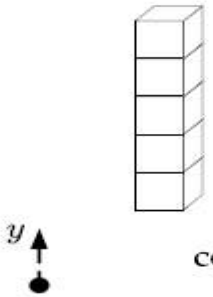
slice



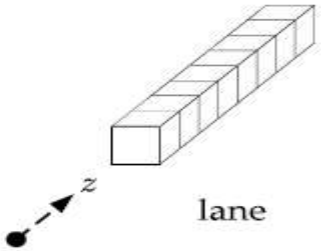
sheet



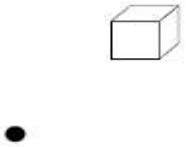
row



column



lane



bit

For all triples (x, y, z) such that $0 \le x < 5$, $0 \le y < 5$, and $0 \le z < w$,

$$\mathbf{A}[x, y, z] = S[w(5y + x) + z].$$

For example, if $b = 1600$, so that $w = 64$, then

$$\begin{array}{llll}
\mathbf{A}[0,0,0] = S[0] & \mathbf{A}[1,0,0] = S[64] & \mathbf{A}[2,0,0] = S[128] & \mathbf{A}[3,0,0] = S[192] \\
\mathbf{A}[0,0,1] = S[1] & \mathbf{A}[1,0,1] = S[65] & \mathbf{A}[2,0,1] = S[129] & \mathbf{A}[3,0,1] = S[193] \\
\mathbf{A}[0,0,2] = S[2] & \mathbf{A}[1,0,2] = S[66] & \mathbf{A}[2,0,2] = S[130] & \mathbf{A}[3,0,2] = S[194] \\
\vdots & \vdots & \vdots & \vdots \\
\mathbf{A}[0,0,62] = S[62] & \mathbf{A}[1,0,62] = S[126] & \mathbf{A}[2,0,62] = S[190] & \mathbf{A}[3,0,62] = S[254] \\
\mathbf{A}[0,0,63] = S[63] & \mathbf{A}[1,0,63] = S[127] & \mathbf{A}[2,0,63] = S[191] & \mathbf{A}[3,0,63] = S[255]
\end{array}$$

and

$$\begin{array}{llll}
\mathbf{A}[4,0,0] = S[256] & \mathbf{A}[0,1,0] = S[320] & \mathbf{A}[1,1,0] = S[384] & \mathbf{A}[2,1,0] = S[448] \\
\mathbf{A}[4,0,1] = S[257] & \mathbf{A}[0,1,1] = S[321] & \mathbf{A}[1,1,1] = S[385] & \mathbf{A}[2,1,1] = S[449] \\
\mathbf{A}[4,0,2] = S[258] & \mathbf{A}[0,1,2] = S[322] & \mathbf{A}[1,1,2] = S[386] & \mathbf{A}[2,1,2] = S[450] \\
\vdots & \vdots & \vdots & \vdots \\
\mathbf{A}[4,0,62] = S[318] & \mathbf{A}[0,1,62] = S[382] & \mathbf{A}[1,1,62] = S[446] & \mathbf{A}[2,1,62] = S[510] \\
\mathbf{A}[4,0,63] = S[319] & \mathbf{A}[0,1,63] = S[383] & \mathbf{A}[1,1,63] = S[447] & \mathbf{A}[2,1,63] = S[511]
\end{array}$$

etc.

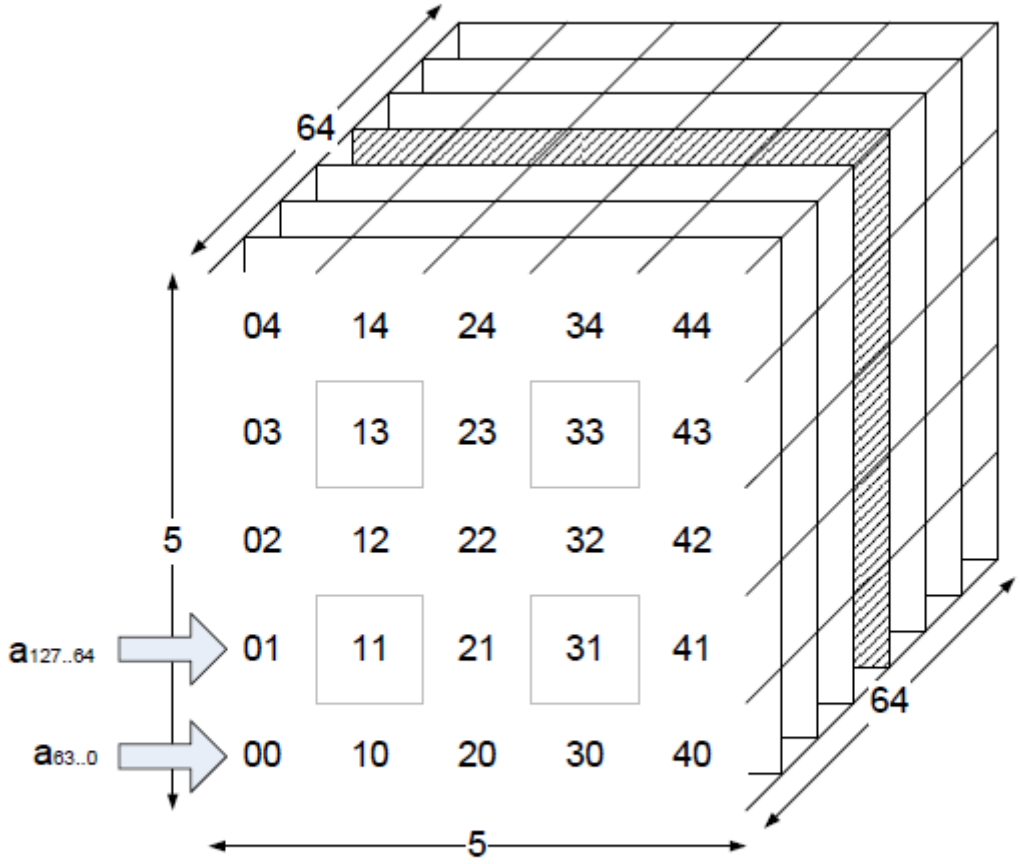
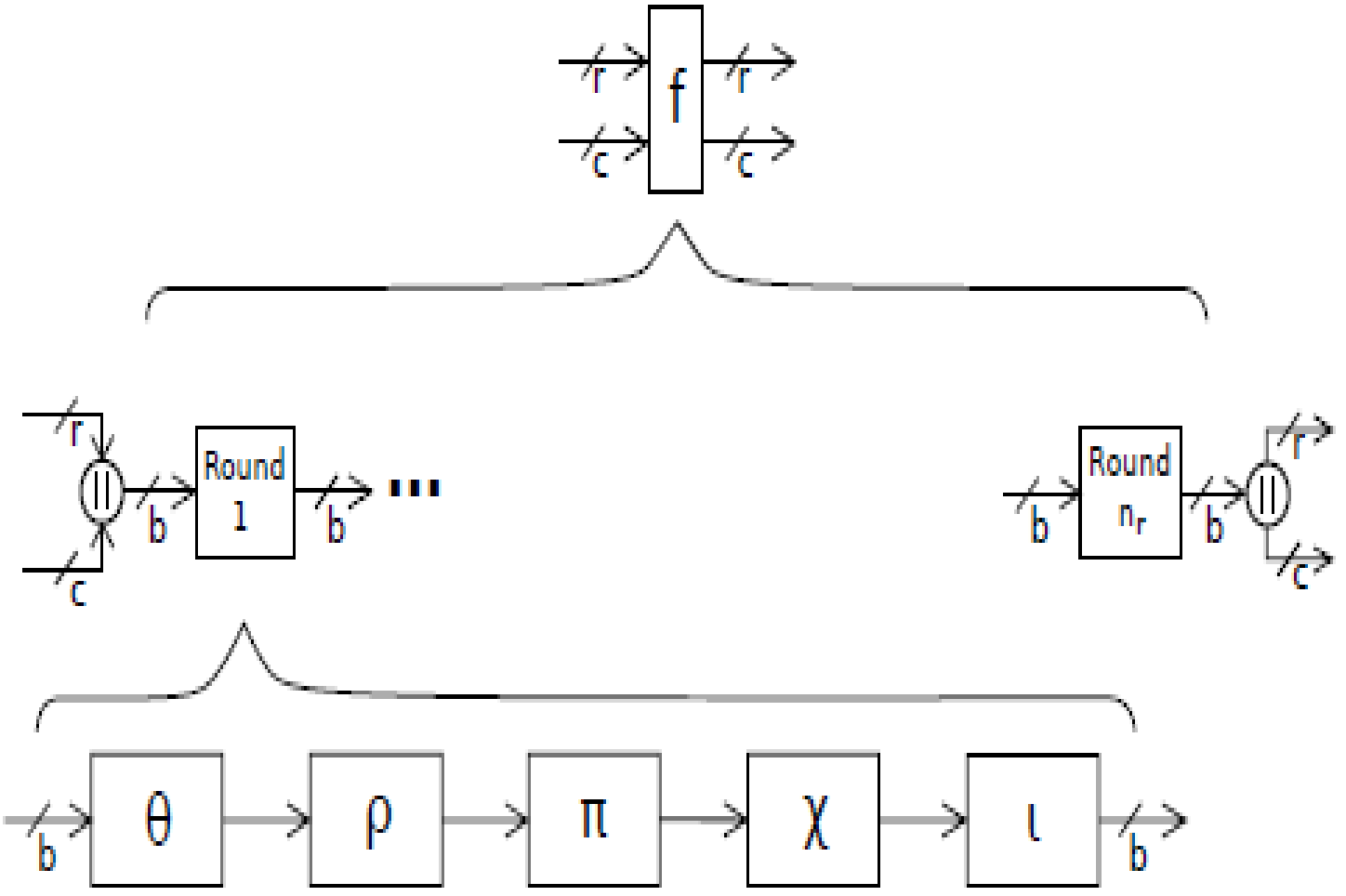


Figure 3.30: Keccak : State

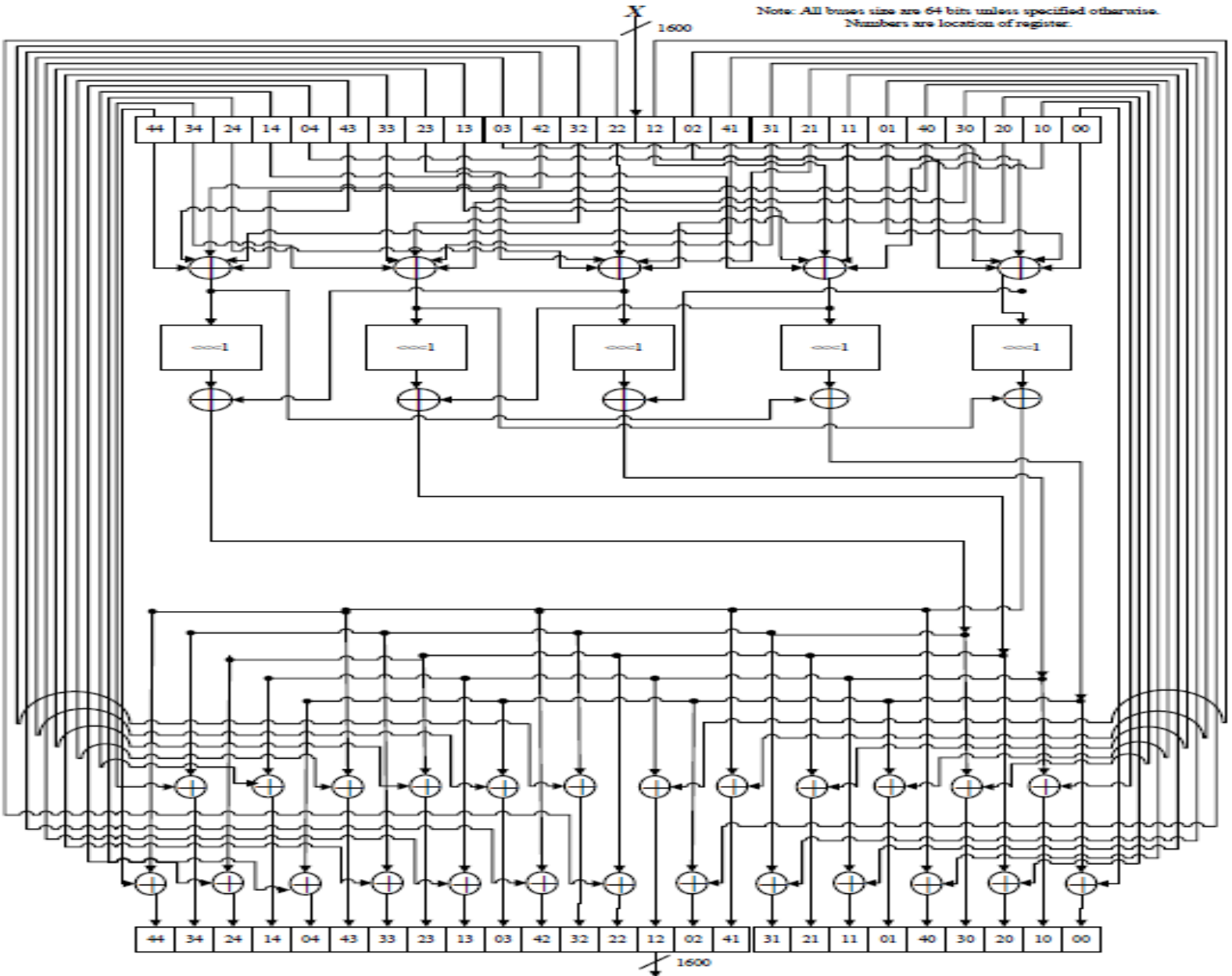

f
r
c
r
c
Round
1
b
Round
n_r
θ
ρ
π
χ
ι

$$
\begin{aligned}
C[x] &= A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4] \quad , x = 0,1,2,3,4 \\
D[x] &= C[x-1] \oplus \mathrm{rot}(C[x+1],1) \quad , x = 0,1,2,3,4 \\
A[x,y] &= A[x,y] \oplus D[x] \quad , x,y = 0,1,2,3,4
\end{aligned}
$$

$$
B[y, 2x+3y] = \mathrm{rot}(A[x,y], r[x,y]) \quad , \quad x,y = 0,1,2,3,4
$$

$$
A[x,y] = B[x,y] \oplus ((\bar{B}[x+1,y]) \wedge B[x+2,y]) \quad , \quad x,y = 0,1,2,3,4
$$

Round operation Part I



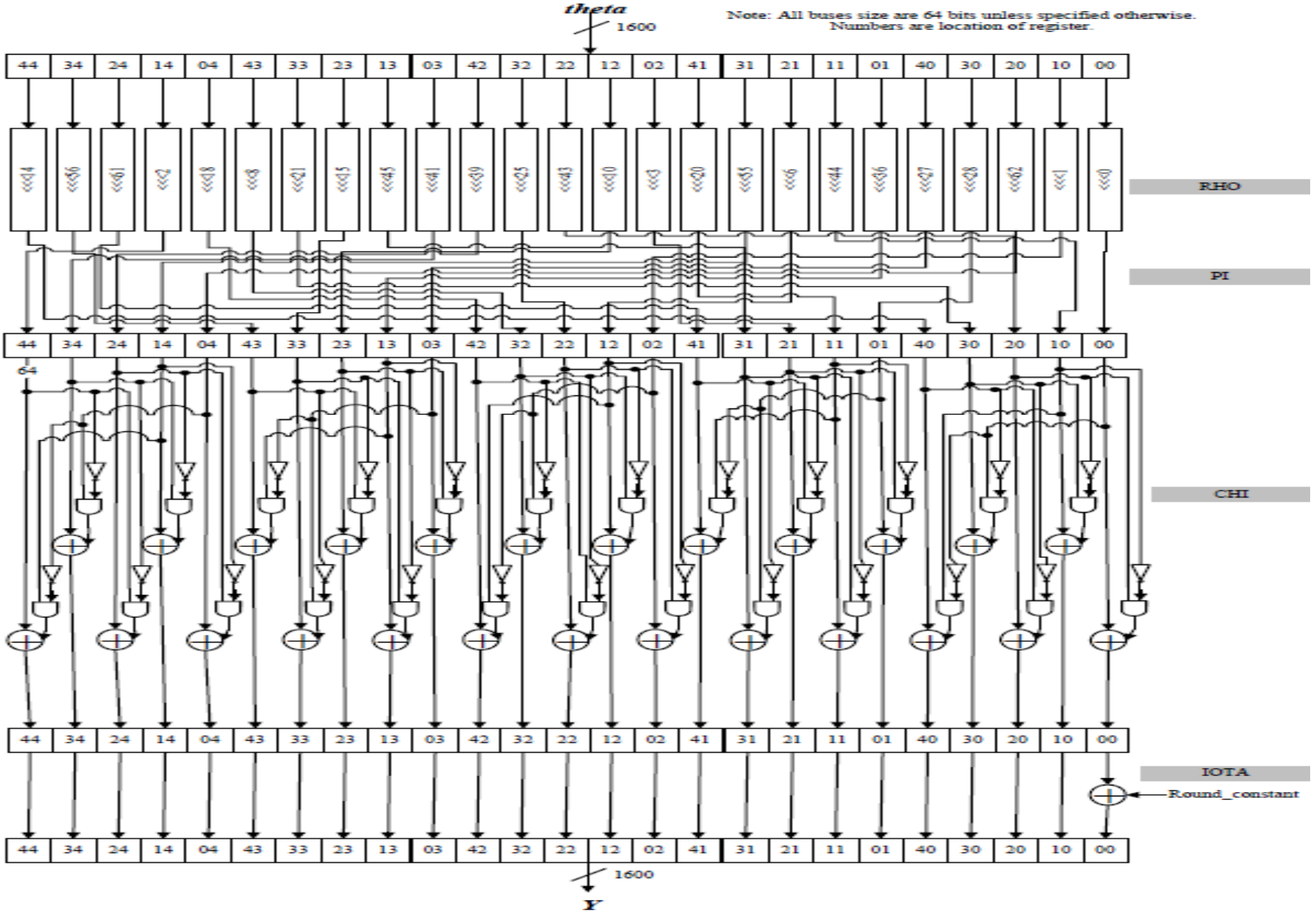


Figure 3.32: Keccak : Round Part 2

- Five functions are used in each round called “step mappings”.

Round constants

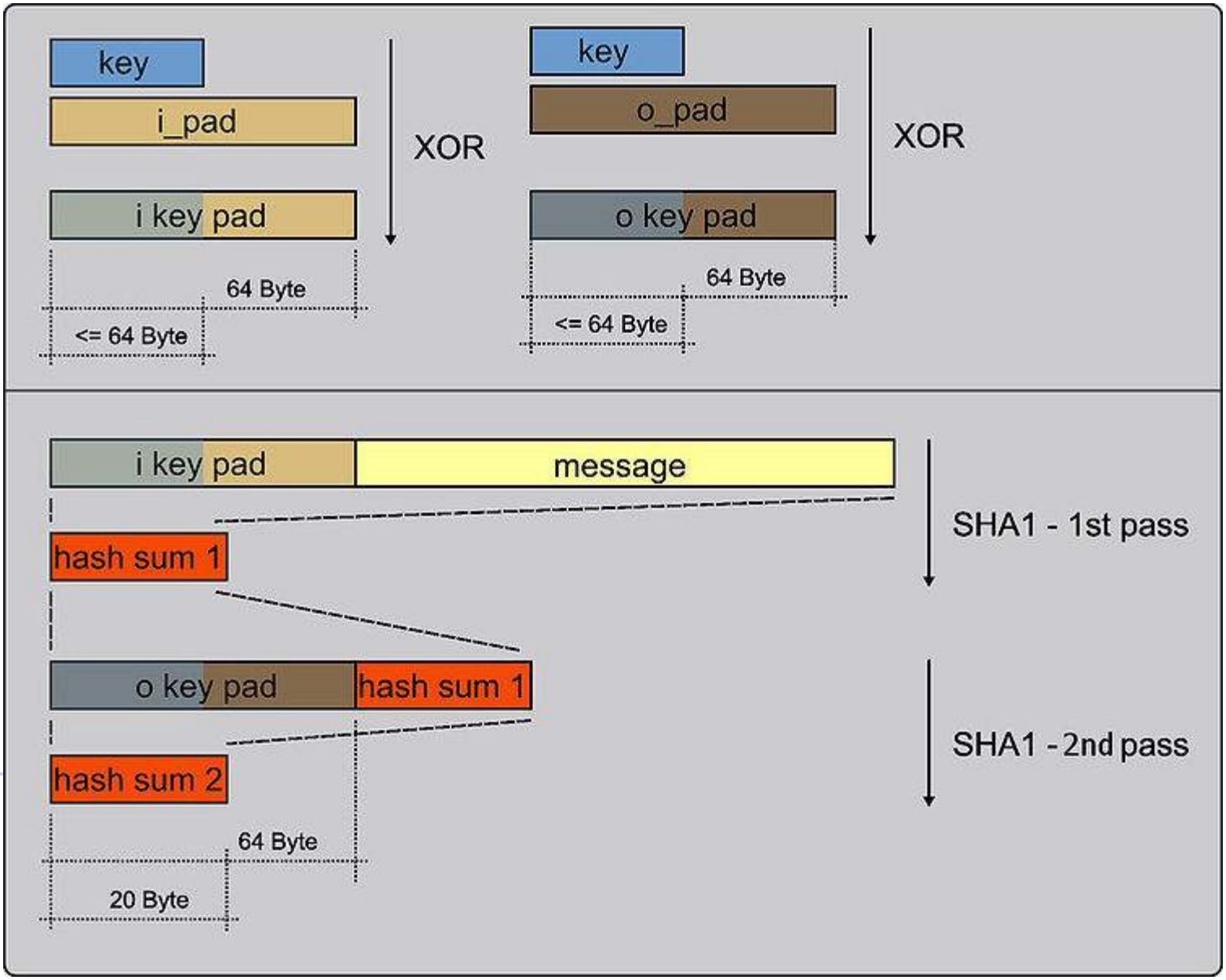
RC[0] = 0x0000000000000001
RC[1] = 0x0000000000008082
RC[2] = 0x800000000000808A
RC[3] = 0x8000000080008000
RC[4] = 0x000000000000808B
RC[5] = 0x0000000080000001
RC[6] = 0x8000000080008081
RC[7] = 0x8000000000008009
RC[8] = 0x000000000000008A
RC[9] = 0x0000000000000088
RC[10] = 0x0000000080008009
RC[11] = 0x000000008000000A
RC[12] = 0x000000008000808B
RC[13] = 0x800000000000008B
RC[14] = 0x8000000000008089
RC[15] = 0x8000000000008003
RC[16] = 0x8000000000008002
RC[17] = 0x8000000000000080
RC[18] = 0x000000000000800A
RC[19] = 0x800000008000000A
RC[20] = 0x8000000080008081
RC[21] = 0x8000000000008080
RC[22] = 0x0000000080000001
RC[23] = 0x8000000080008008

Rotation offsets

	x = 3	x = 4	x = 0	x = 1	x = 2
y = 2	25	39	3	10	43
y = 1	55	20	36	44	6
y = 0	28	27	0	1	62
y = 4	56	14	18	2	61
y = 3	21	8	41	45	15

HMAC

- **H**(·) be a cryptographic hash function
- *K* be a secret key padded to the left with zeros such that the block is 512bits.
- *m* be the message to be authenticated
- *opad* be the outer padding (0x5c5c5c…5c5c, block-long constant)
- *ipad* be the inner padding (0x363636…3636, one-block-long constant)
- Then **HMAC**(*K*,*m*) is mathematically defined by
  - **HMAC**(*K*,*m*) = **H**((*K* +*opad*) // **H**((*K* + *ipad*)// m)).
  - // means concatenation


key
i_pad
XOR
i key pad
64 Byte
<= 64 Byte
key
o_pad
XOR
o key pad
64 Byte
<= 64 Byte
i key pad
message
SHA1 - 1st pass
hash sum 1
o key pad
hash sum 1
SHA1 - 2nd pass
hash sum 2
64 Byte
20 Byte

DSS

- DSS versus RSA
- DSS uses a random number k, a private key and a global public key
- Signature comprises of two components.
- Test interestingly is on *r* which does not depend on the message.
- Hence demands calculation of *r* and k^{-1}

Digital signature algorithm

- p 1024 bit prime
- q 160 bit prime factor of (p-1)
- $g = h^{(p-1)/q} \bmod p$ where $h < p$-1
- *Public key* $y = g^x \bmod p$
- *Private key* $x < q$ (a 160 bit number)

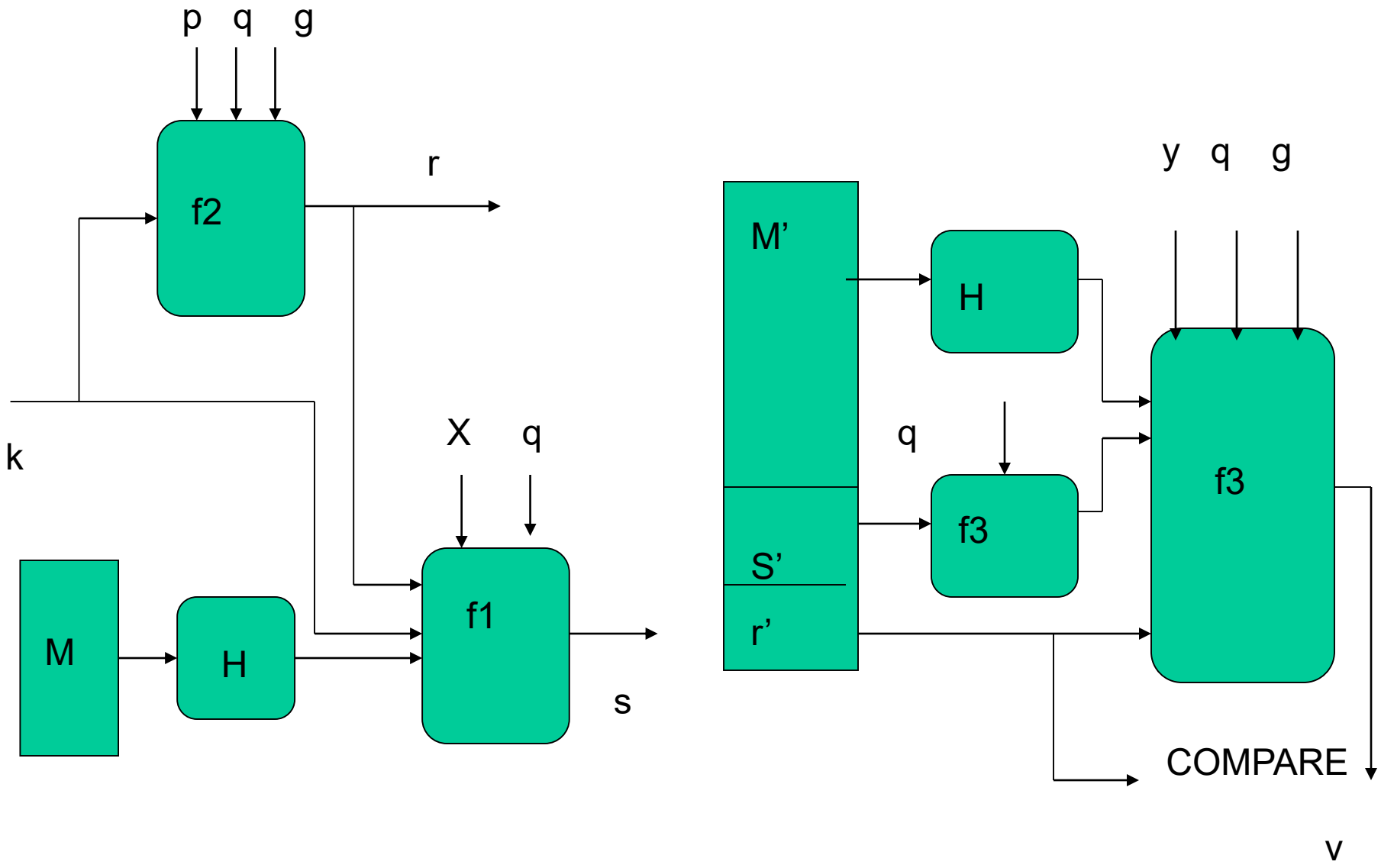
DIGITAL SIGNATURE ALGORITHM

- *Signing:*
- choose a random $k < q$
- $r = (g^k \bmod p) \bmod q$
- $s = (k^{-1}(H(m) + xr) \bmod q$
- send r and s
- Note that $H(m)$ can be obtained by SHA

DIGITAL SIGNATURE ALGORITHM

- Verifying:
- $w=s^{-1} \bmod q$
- $u_1 = (H(m).w) \bmod q$
- $u_2 = (r.w) \bmod q$
- $v = ((g^{u1}.y^{u2}) \bmod p\ ).\bmod q$
- If $v = r$, the signature is verified.

DSS Signing and Verifying



Elliptic curve cryptography

Elliptic curve Cryptography

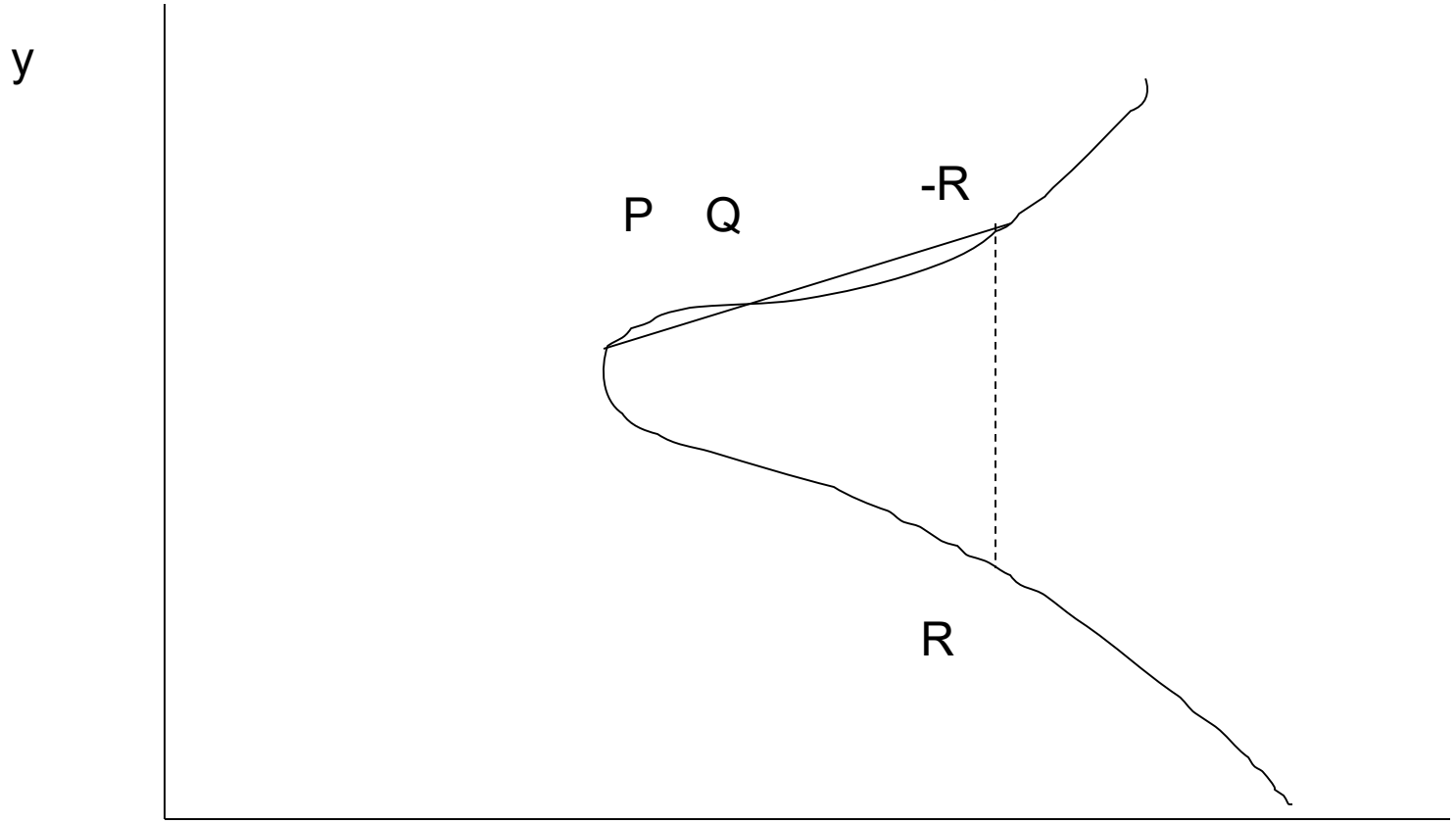
- Elliptic curves have nothing to do with ellipses.
- Elliptic curves are of the form
- $y^2 = x^3+ax+b \bmod p$
- p is a prime, $4a^3 + 27b^2 \neq 0 (\bmod p)$.
- a and b need to be chosen.
- There are other types of Elliptic curves also
- $y^2+xy = x^3+ax+b$
-

- There are points on the elliptic curves e.g (x,y) where x and y are integers
- Points on the elliptic curve can be added
- Multiplication in this domain is addition.
- $2P = P+P$, $3P = 2P+P$, $4P = 2P+2P$, $P^{10} = 10P$

About Elliptic Curves

- We define a point O as point at infinity.
- For any point on the curve P: P+O = P
- O = -O
- If a point P is (x,y), -P is (-x,y) i.e. they are on a vertical line.
- To add two points, draw a line between them which intersects the curve at point R. Then, P+Q = -R.
- To find 4P, or P^4, you have to find P+P+P+P

How to Find (P+Q)?



- To find 2P draw a tangent at P. Find intersection with curve and mirror of that point.

Other computations

- $2P = P+P$
- Can be in finite field of integers or $GF(2^m)$ (polynomial rings)
- We consider Integer field only.
- e.g. $p = 23$
- i.e $y^2 = x^3+ax+b \bmod 23$
- e.g for curve parameters $a = 1, b = 1, x = 9, y = 7$. Therefore (9,7) is a point on the elliptic curve chosen. Check $49 = (729+9+1) \bmod 23$!!!

Example 5: An elliptic curve over F_{23}.

Let $y^2 = x^3 + x + 1$ be an equation over F_{23}. Here $a = 1$ and $b = 1$. Then the solutions over F_{23} to the equation of the elliptic curve are:

(0,1)	(0,22)	(1,7)	(1,16)	(3,10)	(3,13)	(4,0)	(5,4)	(5,19)
(6,4)	(6,19)	(7,11)	(7,12)	(9,7)	(9,16)	(11,3)	(11,20)	(12,4)
(12,19)	(13,7)	(13,16)	(17,3)	(17,20)	(18,3)	(18,20)	(19,5)	(19,18).

Points on Elliptic curve

- Points can be more than p
- Point counting algorithms: Schoof's algorithm, Satohs technique
- G base point or Generating point
- Use standard curves with known number of points on the curve $\#E(F_p)$ including point at infinity.
- Order *n*: maximum power (exponent) to which base point can be raised
- Shall be very large to make search space for key very large
- Co-factor $h = \#E(F_p)/n$ is an integer shall be e.g. 1-6.

Typical Curve parameters Example

P-192: $p = 2^{192} - 2^{64} - 1$, $a = -3$, $h = 1$,

$b =$ 0x 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1

$n =$ 0x FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831

P-224: $p = 2^{224} - 2^{96} + 1$, $a = -3$, $h = 1$,

$b =$ 0x B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4

$n =$ 0x FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D

P-256: $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, $a = -3$, $h = 1$,

$b =$ 0x 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E
27D2604B

$n =$ 0x FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2
FC632551

P-384: $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$, $a = -3$, $h = 1$,

$b =$ 0x B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F
5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF

$n =$ 0x FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81
F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973

P-521: $p = 2^{521} - 1$, $a = -3$, $h = 1$,

$b =$ 0x 00000051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3
B8B48991 8EF109E1 56193951 EC7E937B 1652C0BD 3BB1BF07 3573DF88
3D2C34F1 EF451FD4 6B503F00

$n =$ 0x 000001FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFA 51868783 BF2F966B 7FCC0148 F709A5D0 3BB5C9B8
899C47AE BB6FB71E 91386409

2.3.2 Recommended Parameters secp128r2

The verifiably random elliptic curve domain parameters over $\mathbb{F}_p$ `secp128r2` are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$$p = \texttt{FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFF}$$
$$= 2^{128} - 2^{97} - 1$$

The curve E: $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$$a = \texttt{D6031998 D1B3BBFE BF59CC9B BFF9AEE1}$$
$$b = \texttt{5EEEFCA3 80D02919 DC2C6558 BB6D8A5D}$$

E was chosen verifiably at random as specified in ANSI X9.62 [1] from the seed:

$$S = \texttt{004D696E 67687561 517512D8 F03431FC E63B88F4}$$

The base point G in compressed form is:

$$G = \texttt{02 7B6AA5D8 5E572983 E6FB32A7 CDEBC140}$$

and in uncompressed form is:

$$G = \texttt{04 7B6AA5D8 5E572983 E6FB32A7 CDEBC140 27B6916A 894D3AEE}$$
$$\texttt{7106FE80 5FC34B44}$$

Finally the order n of G and the cofactor are:

$$n = \texttt{3FFFFFFF 7FFFFFFF BE002472 0613B5A3}$$
$$h = \texttt{04}$$

The basic computation

- P =(x_P,y_P) and Q=(x_Q,y_Q), to find R =(x_R,y_R)
- Use the formulas
- Define $\lambda = ((y_Q - y_P)/(X_Q - X_P)) \bmod p$ if P is not equal to Q and $\lambda = ((3X^2_P + a)/2Y_P) \bmod p$ if P = Q.
- If P not equalling Q, then
- $X_R = (\lambda^2 - X_P - X_Q) \bmod p$
- $Y_R = (\lambda(X_P - X_R) - y_P) \bmod p$
- If P = Q,
- $X_R = (\lambda^2 - 2X_P) \bmod p$
- $Y_R = (\lambda(X_P - X_R) - y_P) \bmod p$
- G is base point such that n.G = O.
- n is called the “order”.

Example

- P(3,10) Q=(9,7)
- a = 1, b = 1 for the curve chosen.
- p = 23.
- Find R.
- $\lambda = ((y_Q-y_P)/(x_Q-x_P)) \bmod p = ((7-10)/(9-3)) \bmod 23$
- $= (-3/6) \bmod 23 = (-1/2) \bmod 23 = 11$
- $X_R = (\lambda^2 - X_P - X_Q) \bmod p = (11^2-3-9) \bmod 23$
- $= 109 \bmod 23 = 17$
- $Y_R = (\lambda(X_P-X_R)-y_P) \bmod p$
- $= (11(3-17)-10) \bmod 23 = 20$

EC point subtraction

- Add the point –Q to P
- -Q is (x,-y) if Q = (x,y)

How to avoid division?

- Division is multiplication with multiplicative inverse.
- Use projective coordinates. Trade off division with multiplications and additions.
-

$$x = \frac{X}{Z^2}, y = \frac{Y}{Z^3},$$

$$E(F_p): Y^2 = X^3 + aXZ^4 + bZ^6$$

The addition of two points $P_o = (X_o, Y_o, Z_o)$ and $P_1 = (X_1, Y_1, Z_1)$ thus will yield he sum $P_2 = P_0 + P_1 = (X_2, Y_2, Z_2)$ given by

$$P_2 = P_0 + P_1 = \begin{cases} X_2 = R^2 - T W^2 \\ 2 Y_2 = VR - M W^3 \\ Z_2 = Z_0 Z_1 W \end{cases}$$

$$W = X_0 Z_1^2 - X_1 Z_0^2, R = Y_0 Z_1^3 - Y_1 Z_0^3, T = X_0 Z_1^2 + X_1 Z_0^2,$$
$$M = Y_0 Z_1^3 + Y_1 Z_0^3, V = T W^2 - 2 X_2$$

The doubling of point P_1 is given as

$$P_2 = 2 P_1 = \begin{cases} X_2 = M^2 - 2S \\ Y_2 = M(S - X_2) - T \\ Z_2 = 2 Z_1 Y_1 \end{cases}$$

$$M = 3 X_1^2 + a Z_1^4, S = 4 X_1 Y_1^2, T = 8 Y_1^4$$

- Intensive in additions and multiplications but avoids division/finding Multiplicative inverses)

Three Algorithms

- ECDH
- ECDSA
- ECMQV

Diffie- Hellman key exchange General method and for ECC

- Users A and B know *p* and *n*.
- A and B select two numbers *x* and *y*
- A sends $k_1 = p^x \bmod n$ to b
- B sends $k_2 = p^y \bmod n$ to a
- A and b compute $k_2^x \bmod n$ and $k_1^y \bmod n$ and use them as their keys
- Exponentiation algorithms needed
- Both the keys are same.
- Applicable or ECC also.
- x and y are numbers chosen.
- But you find xG, yG where G is the base point on the Elliptic curve. Send Xg to B , yG to A.
- Both compute xyG = yxG.

EC Diffie-Hellman

- *A* chooses Private Key d_A. Calculates $Q_A = d_A G$ and sends to *A*
- *B* chooses Private Key d_B , Calculates $Q_B = \mathrm{d_B} G$ and sends to *A*
- *A* calculates $d_A Q_B$ and *B* calculates $d_B Q_A$ to get session key *SK*:
- $SK = d_A Q_B = d_A(d_B G) = d_B(d_A G) = d_B Q_A$.
- *x* coordinate of the output point can be chosen as key.

ECDSA Signature Generation

ign a message m, A does the following:

Select a random integer k, $1 \leq k \leq n-1$.

Compute $R = kP$ and $r = x(R) \bmod n$.
If $r = 0$ then go to step 1.

Compute $k^{-1} \bmod n$.

Compute $e = H(m)$, where H is a hash function.

Compute $s = k^{-1}(e + dr) \bmod n$.
If $s = 0$ then go to step 1.

A's signature for the message m is (r, s).

ECDSA Signature Verification

To verify A's signature (r, s) on m, B should do the following

1. Verify that r and s are integers in the interval $[1, n-1]$.
2. Compute $e = H(m)$.
3. Compute $u_1 = es^{-1} \bmod n$ and $u_2 = rs^{-1} \bmod n$.
4. Compute $R = u_1 P + u_2 Q$ and $v = x(R) \bmod n$.
5. Accept the signature if and only if $v = r$.

ECMQV (Menezes-Qu-Vanstone)

- P is a point on the elliptic curve
- Alice generates a key pair (X,x) i.e $X=xP$
- Bob generates a key pair (Y,y) i.e $Y=yP$
- Alice calculates $S_a = (x + X'a) \bmod n$
- Bob calculates $S_b = (y + Y'b) \bmod n$
- S_a and S_b are called implicit signatures since they use their own private key and other party's Public Key
- Alice calculates $K = hS_a(Y+Y'B)$ and Bob calculates $K = hS_b(X+X'A)$ where X' and Y' are the first L bits of X and Y 's x component where $L = (\log_2 n+1)/2$
- K is the *shared secret key* which both have arrived at.

Curve Selection

Popular choices are the 15 elliptic curves recommended by NIST in FIPS 186-2:

- One randomly-selected curve (with $h = 2$) over each of the binary fields $\mathbb{F}_{2^{163}}$, $\mathbb{F}_{2^{233}}$, $\mathbb{F}_{2^{283}}$, $\mathbb{F}_{2^{409}}$, $\mathbb{F}_{2^{571}}$.
- One Koblitz curve (with $h = 2$ or 4) over each of the binary fields $\mathbb{F}_{2^{163}}$, $\mathbb{F}_{2^{233}}$, $\mathbb{F}_{2^{283}}$, $\mathbb{F}_{2^{409}}$, $\mathbb{F}_{2^{571}}$.
- One randomly-selected curve (with $h = 1$) over each of the prime fields $\mathbb{F}_p$ for the following p;
  - $p = 2^{192} - 2^{64} - 1$.
  - $p = 2^{224} - 2^{96} + 1$.
  - $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.
  - $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$.
  - $p = 2^{521} - 1$.

Standards for ECC

- P1363
- NIST Digital signature standard
- ANSI Public Key Cryptography for the financial services Industry
- OMA wireless transport layer security Sepcification
- SECG SE1 and SEC2 standards

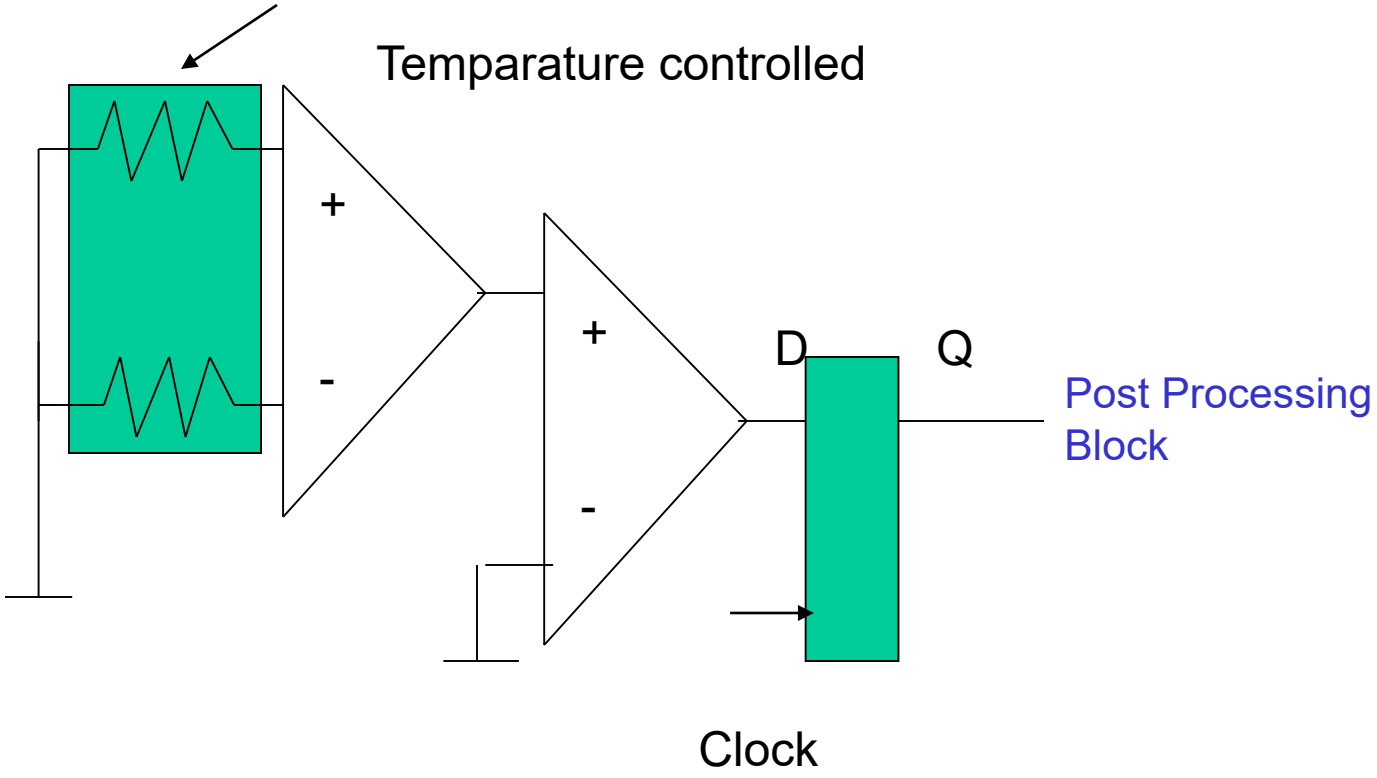
Random Number Generation

TRNG Techniques

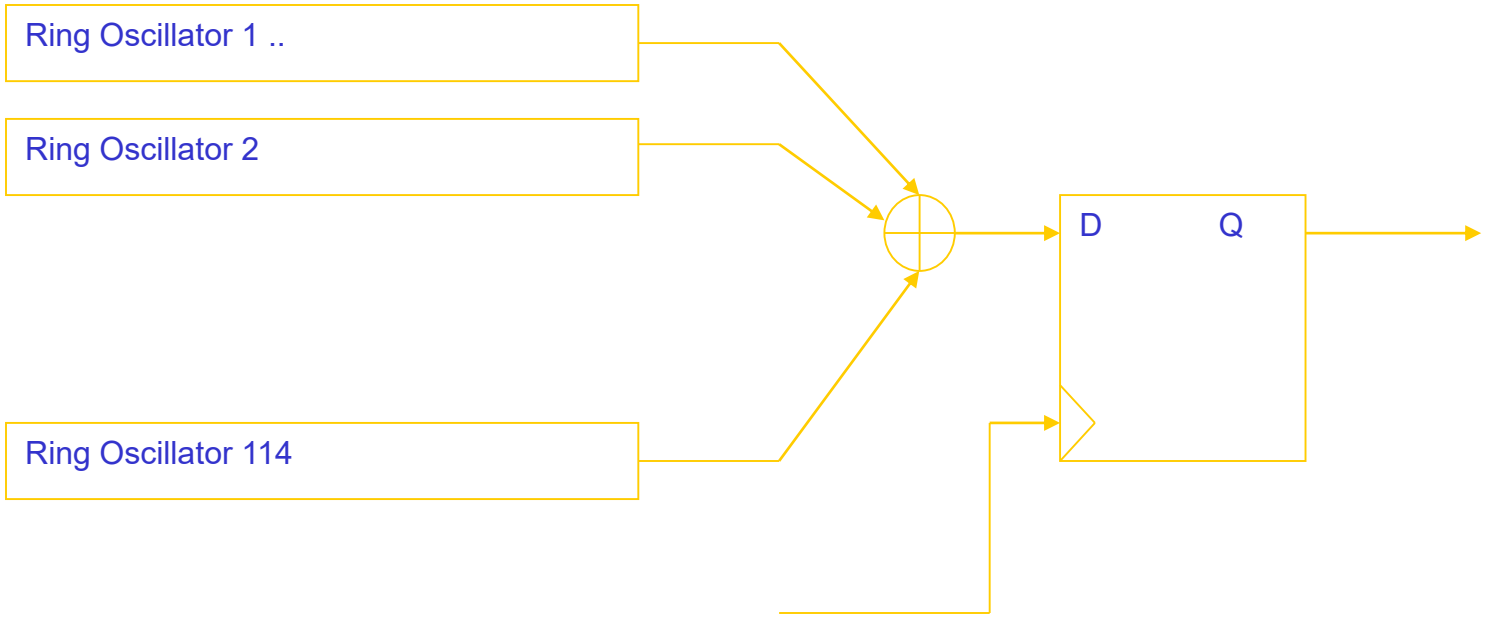
- ***ASIC*** *(Based on Analog Technique)*

- ***FPGA*** *(Based on Digital Technique)*

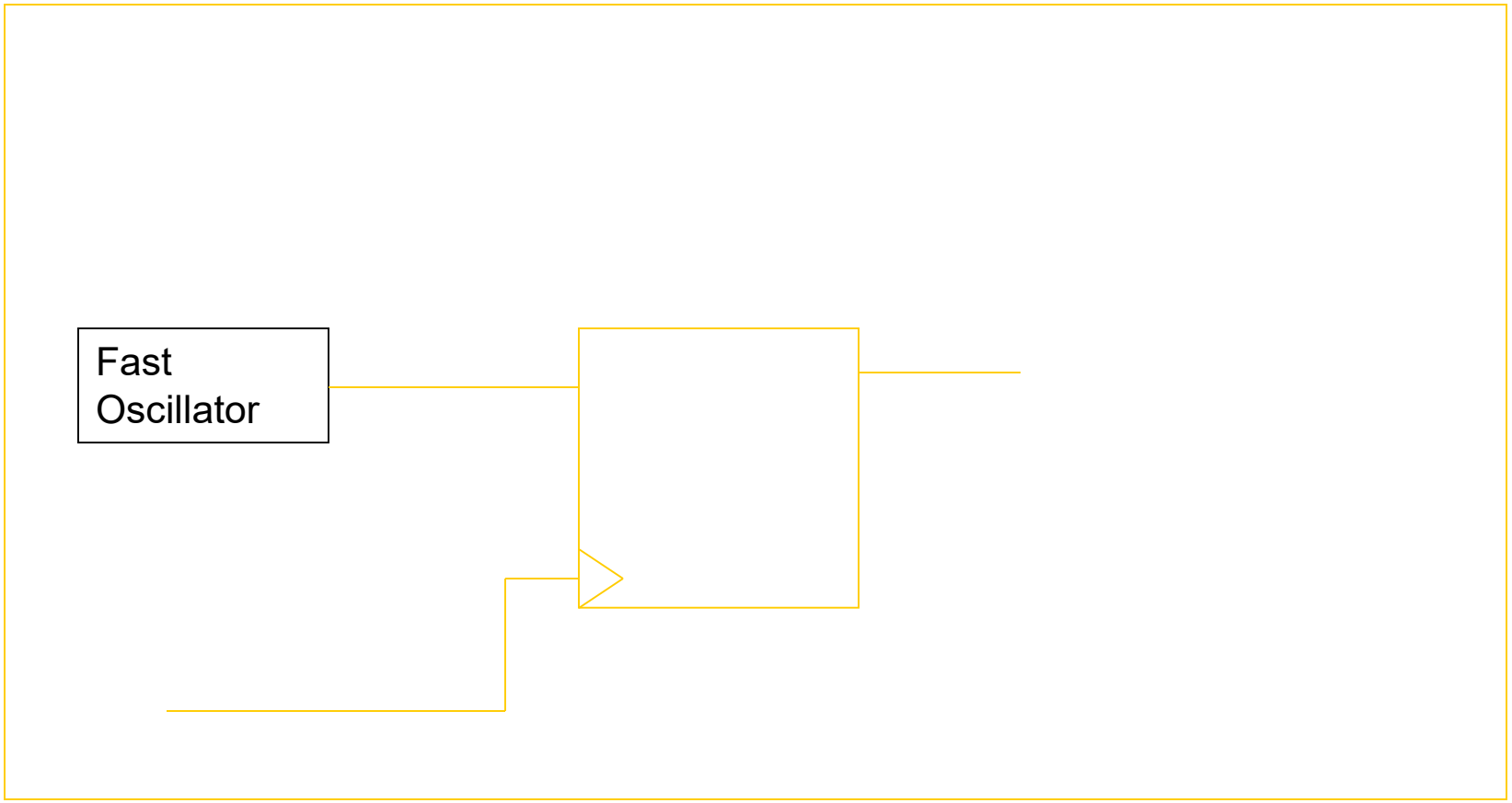
ASIC Method



FPGA Implementations of TRNG (1)

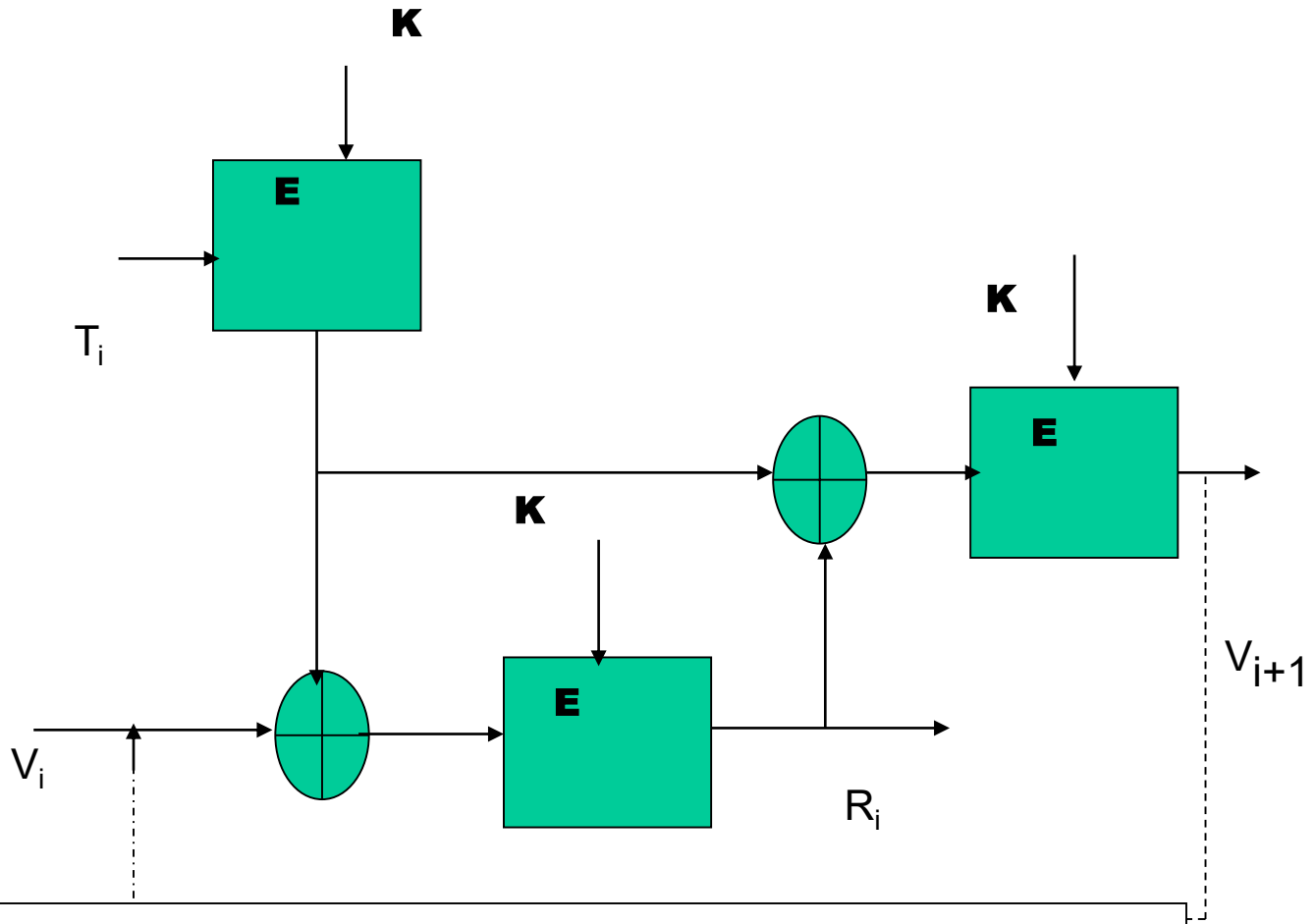


FPGA Implementations of TRNG (2)



Random Number Generation

ANSI X9.31 Random Data Gen.



- Useful for generating session keys
- AES is used T_i is time stamp, V_i is seed and R_i is the output random number

Randomness Tests

- NIST test suite: Many tests
- Five tests (Menezes et al):
- Frequency(1bit) test (count zeroes and ones)
- Serial (2bit) test (count 00,01,10,11 overlapping)
- Poker test (nonoverlapping parts)
- Runs test (runs of zeroes (gaps) and runs of ones (blocks))
- Autocorrelation test (agreement with shifted sequences)

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

$$X_2 = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1$$

$$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

$$X_5 = 2\left(A(d) - \frac{n-d}{2}\right) / \sqrt{n-d}$$

Example (*basic statistical tests*) Consider the (non-random) sequence s of length n =
160 obtained by replicating the following sequence four times:
11100 01100 01000 10100 11101 11100 10010 01001:
(i) (*frequency test*) n0 = 84, n1 = 76, and the value of the statistic X1 is 0.4.
(ii) (*serial test*) n00 = 44, n01 = 40, n10 = 40, n11 = 35, and the value of the statistic X2 is 0.6252.
(iii) (*poker test*) Here m = 3 and k = 53. The blocks 000, 001, 010, 011, 100, 101, 110, 111 appear 5, 10, 6, 4, 12, 3, 6, and 7 times, respectively, and the value of the statistic X3 is 9.6415.
All *m* bit segments are non overlapping.

$$\left\lfloor \frac{n}{m} \right\rfloor \geq 5.2^m, k = \left\lfloor \frac{n}{m} \right\rfloor$$

(iv) (*runs test*) Here e1 = 20.25, e2 = 10.0625, e3 = 5, and k = 3. There are 25, 4, 5 blocks of lengths 1, 2, 3, respectively, and 8, 20, 12 gaps of lengths 1, 2, 3, respectively.
The value of the statistic X4 is 31.7913. *k* is the largest integer for which $e_i > 5$.

$$e_i = (n-i+3)/2^{i+2}.$$

(v) (*autocorrelation test*) If d = 8, then A(8) = 100. The value of the statistic X5 is 3.8933.

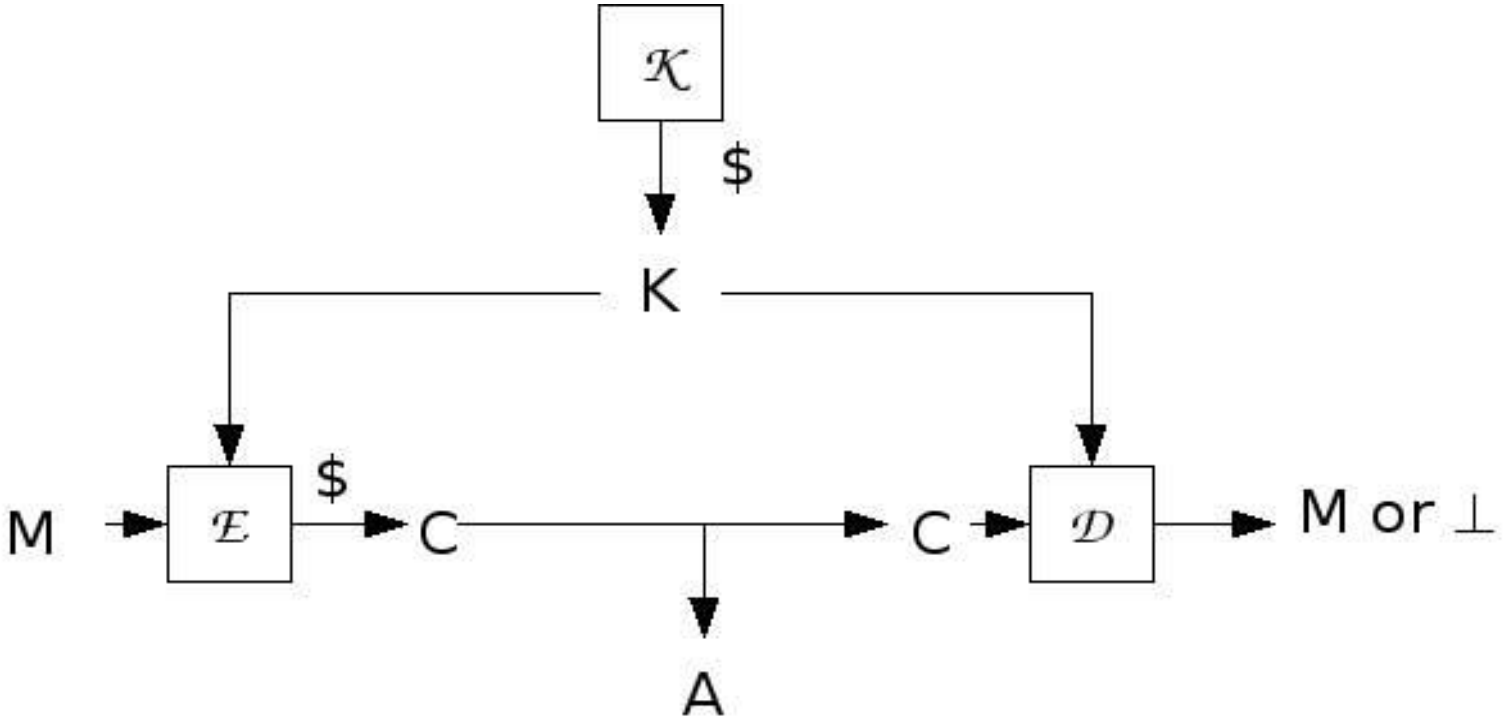
For a significance level of $\alpha = 0.05$, the threshold values for X1, X2, X3, X4, and X5 are
3.8415, 5.9915, 14.0671, 9.4877, and 1.96, respectively
Hence, our sequence passes frequency, serial and poker tests but fails the runs and autocorrelation tests.

- Now on line randomness testing hardware is implemented on FPGA.
- See “On-Line Monitoring of Random Number Generators for Embedded Security” by Renaud Santoro, Olivier Sentieys and S´ebastien Roy.

Definition of AE

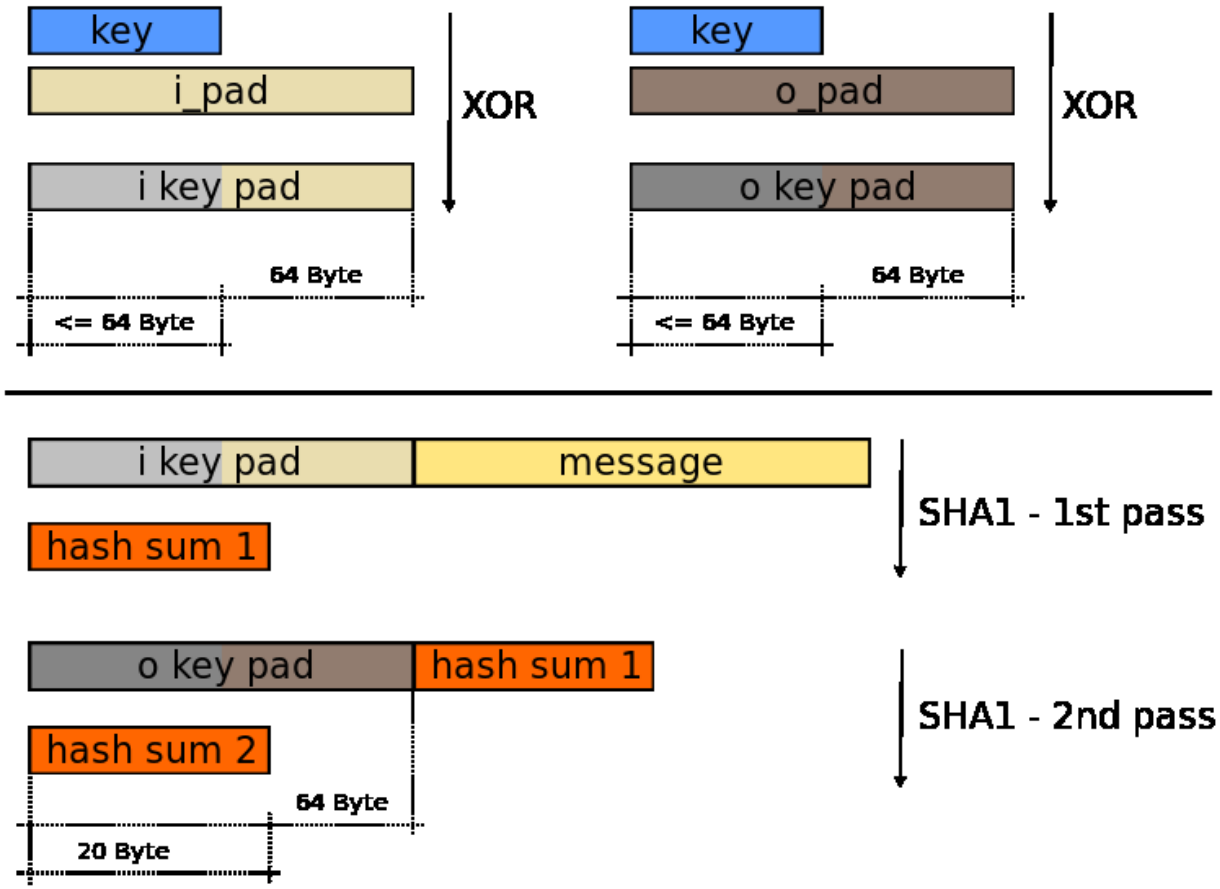
- In practice we often want both privacy and authenticity.
- Example: A doctor wishes to send medical information about Alice to the medical database. Then, we want data privacy to ensure Alice's medical records remain confidential.
- We want authenticity to ensure the person sending the information is really the doctor and the information was not modified in transit.
- We refer to this as **authenticated encryption**.

Simple method of AE



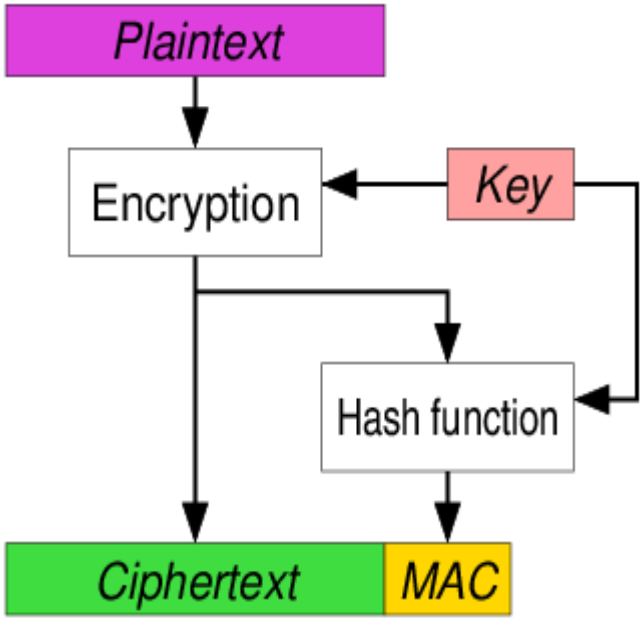
- E can be AES (advanced Encryption standard) and D can be HMAC (hash based message Authentication code)
- Advantage: proven primitives
- Disadvantage1: two keys, can be one key also
- Disdavantage 2: Sequential
- Purpose: Authenticates cipher text

HMAC

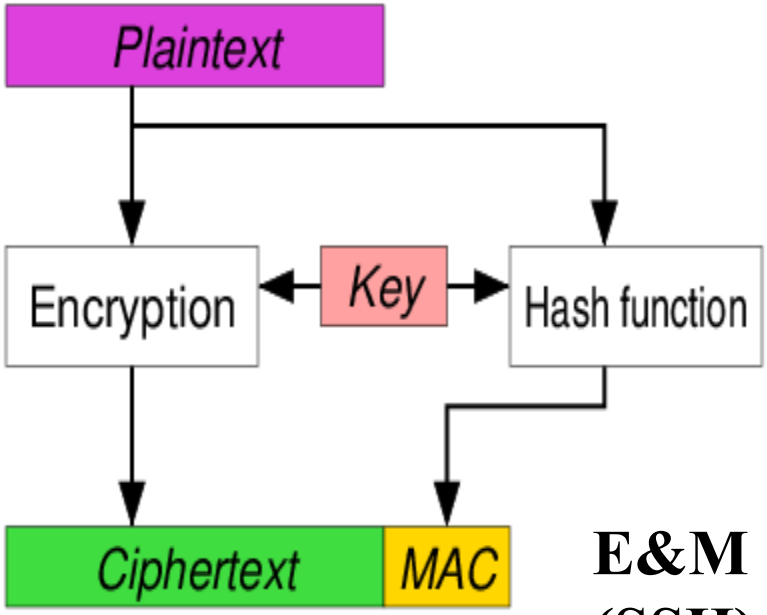


$$\mathrm{HMAC}(K, m) = H\Big((K' \oplus opad) \| H\big((K' \oplus ipad) \| m\big)\Big)$$

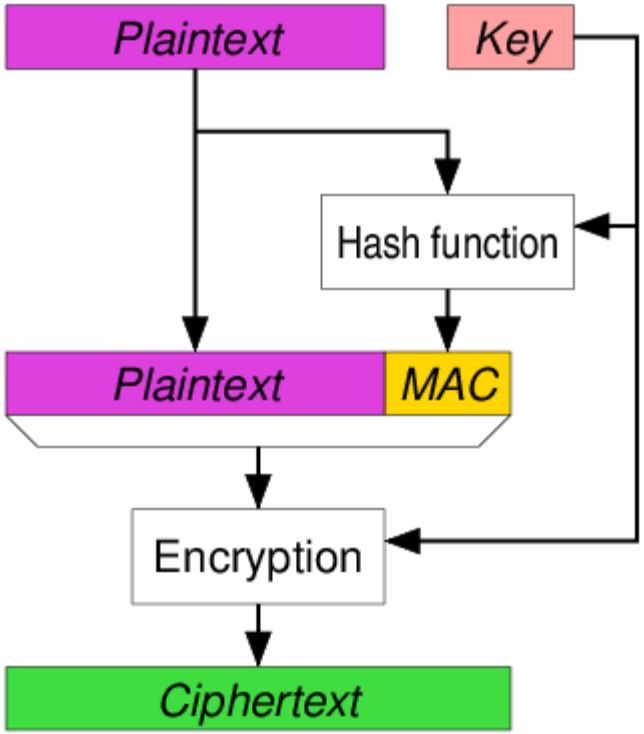
- Opad 5c5c5c…5c
- Ipad 363636…36



EtM (IpSec)



E&M (SSH)



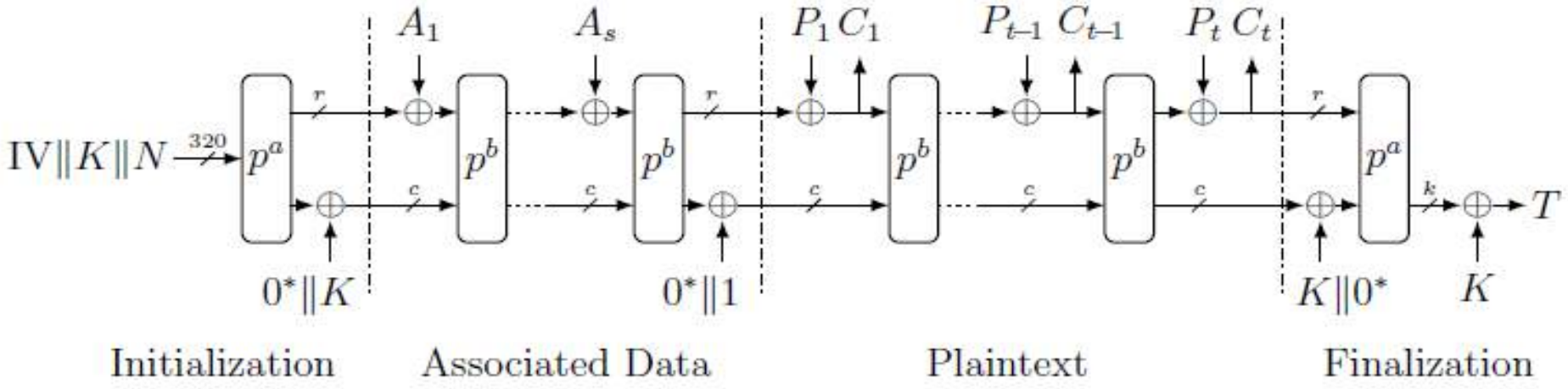
MtE

AEAD Three possible Methods

Method	Usage
Encrypt-and-MAC (E&M)	SSH
MAC-then-encrypt (MtE)	SSL/TLS
Encrypt-then-MAC (EtM)	IPSec

- EtM is believed to be better

ASCON



- Monkey Duplex construction
- IV = 64, K = 128 and N = 128bits
- Rounds $p^a = 12$, $p^b = 6$
- 320 bit state S = IV‖K‖N
- k is key size, r rate, and a , b are number of iterations of p

$$\mathrm{IV} = k \,\|\, r \,\|\, a \,\|\, b \,\|\, 0^{288-2k} = \begin{cases} 80400c0600000000 & \text{for ASCON-128} \\ 80800c0800000000 & \text{for ASCON-128a} \end{cases}$$

ASCON Processing steps

- Initialization

$$S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$$

- Processing of AD of *s* blocks in *s* steps

$$S \leftarrow p^b((S_r \oplus A_i) \| S_c), \qquad 1 \leq i \leq s$$

$$S \leftarrow S \oplus (0^{319} \| 1)$$

- Processing of Plain text

$$C_i \leftarrow S_r \oplus P_i$$

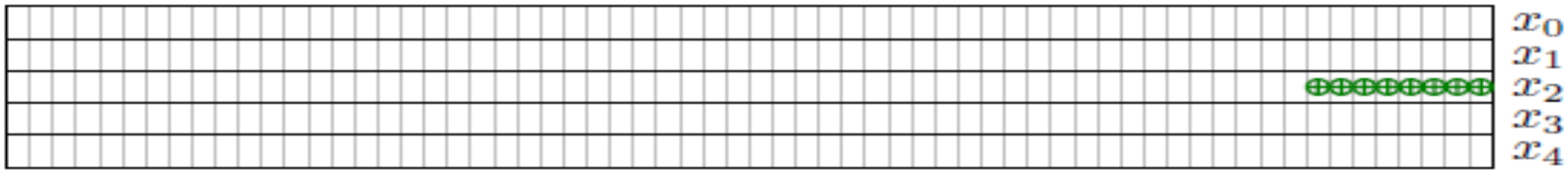
$$S \leftarrow \begin{cases} p^b(C_i \| S_c) & \text{if } 1 \leq i < t, \\ C_i \| S_c & \text{if } 1 \leq i = t. \end{cases}$$

ASCON Processing steps (contd)

- Finalization

$$S \leftarrow p^a(S \oplus (0^r \parallel K \parallel 0^{c-k}))$$
$$T \leftarrow \lceil S \rceil^k \oplus K$$

Permutation step



- Add round constants in last byte of x_2.
- Use 5- bit S-box on 64 Columns
- Use Linear diffusion on 64 bit words to get final 320 bits.

Permutation p

$$p = p_L \circ p_S \circ p_C$$

- Three operations
- p_c add round constant
- p_s S-Box 5-bit

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathcal{S}(x)$	4	11	31	20	26	21	9	2	27	5	8	18	29	3	6	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathcal{S}(x)$	30	19	7	14	0	13	17	24	16	12	1	25	22	10	15	23

Permutation p (contd)

- Third step: linear operation on the five 64 bit words.

$$\Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$
$$\Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$
$$\Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$
$$\Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$
$$\Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)$$

Round Constants

p^{12}	p^8	p^6	Constant	p^{12}	p^8	p^6	Constant
0			00000000000000000f0	6	2	0	0000000000000000096
1			00000000000000000e1	7	3	1	0000000000000000087
2			00000000000000000d2	8	4	2	0000000000000000078
3			00000000000000000c3	9	5	3	0000000000000000069
4	0		00000000000000000b4	10	6	4	000000000000000005a
5	1		00000000000000000a5	11	7	5	000000000000000004b

Benchmarking results

- ATHNEa (Automatic tool for hardware evaluation) of all AEAD schemes
- George Mason University
- Virtex-6, Virtex-7, Stratix IV, Stratix V
- Software Implementations See https://bench.cr.yp.to/supercop.html

Popular Books

- William Stallings
- Bruce Schneier
- Menezes et al (free downloadable)
- Perlman et al
- E-stream site
- Nessie site

Journals

- IEEE transactions on Computers
- IEEE Transactions on Selected Areas in Communications
- IEEE Computer
- IEEE Privacy and Security
- IET journal on Information Security
- Electronics Letters
- IEEE Journal of Solid-State Circuits
- iacr.archives
- Cryptologia
- Dr.Dobb's Journal

CONCLUSION

- Several other areas and applications
- Post-quantum cryptography
- Password hashing
- Attribute based encryption
- Secure Multiparty computation
- Homomorphic encryption