U-II BLOCK CIPHER ALGORITHMS

IDEA:

- ✓ Idea is block cipher similar to DES
- ✓ Works on 64 bit plaintext block
- ✓ Key is longer and consist of 128 bits
- ✓ Idea is reversible like DES i.e. same algorithm can be used for encryption as well as decryption
- ✓ IDEA also uses diffusion as well as confusion techniques

Broad steps in IDEA:

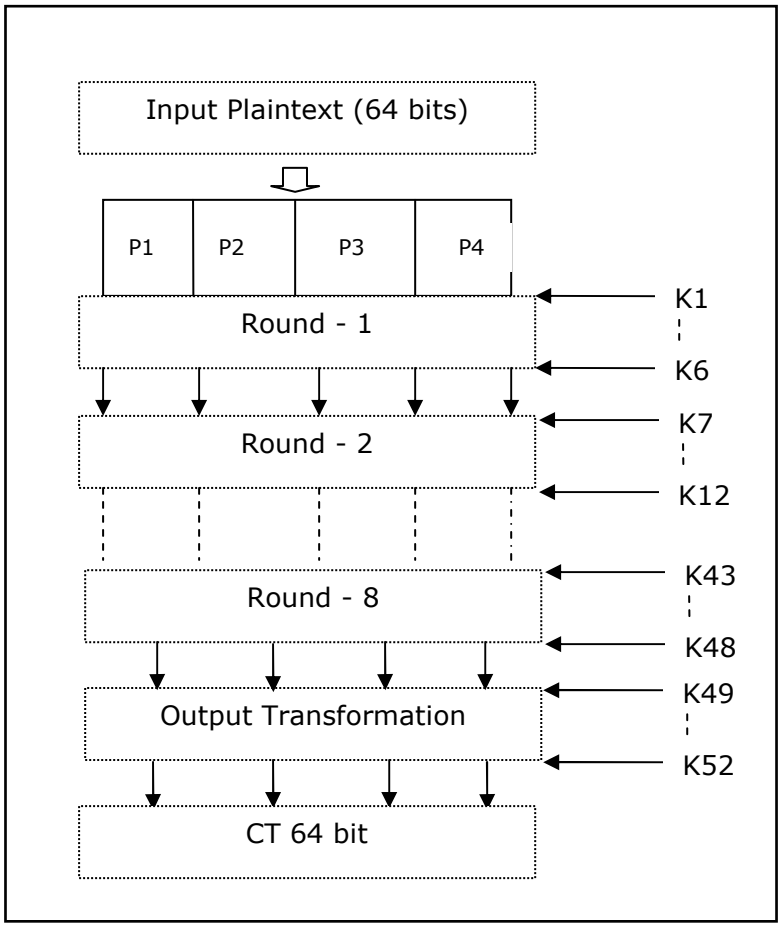


Fig: Broad level Steps in IDEA

- ✓ 64 –bit of Input PT block is divided into four parts (each of size 16 bit) Say p1 to p4 and taken as input in first round
- ✓ There are 8-such rounds and as we mentioned key consist of 128 bit
- ✓ In each round 6-subkeys are generated from the original
- ✓ Each sub key consists of 16-bit and are applied on four input blocks from p1 to p4
- ✓ Eight round consist of series of operation on the four data blocks using six sub keys
- ✓ Above specified broad steps perform lots of mathematical action in each step like Multiplication , Addition and XOR operations
- ✓ ADD* MULTIPLY* are not mere addition and multiplication instead they are addition modulo 2^{16} (Addition Modulo: 65536) and multiplication Modulo $2^{16}+1$ (Multiplication Modulo: 65537

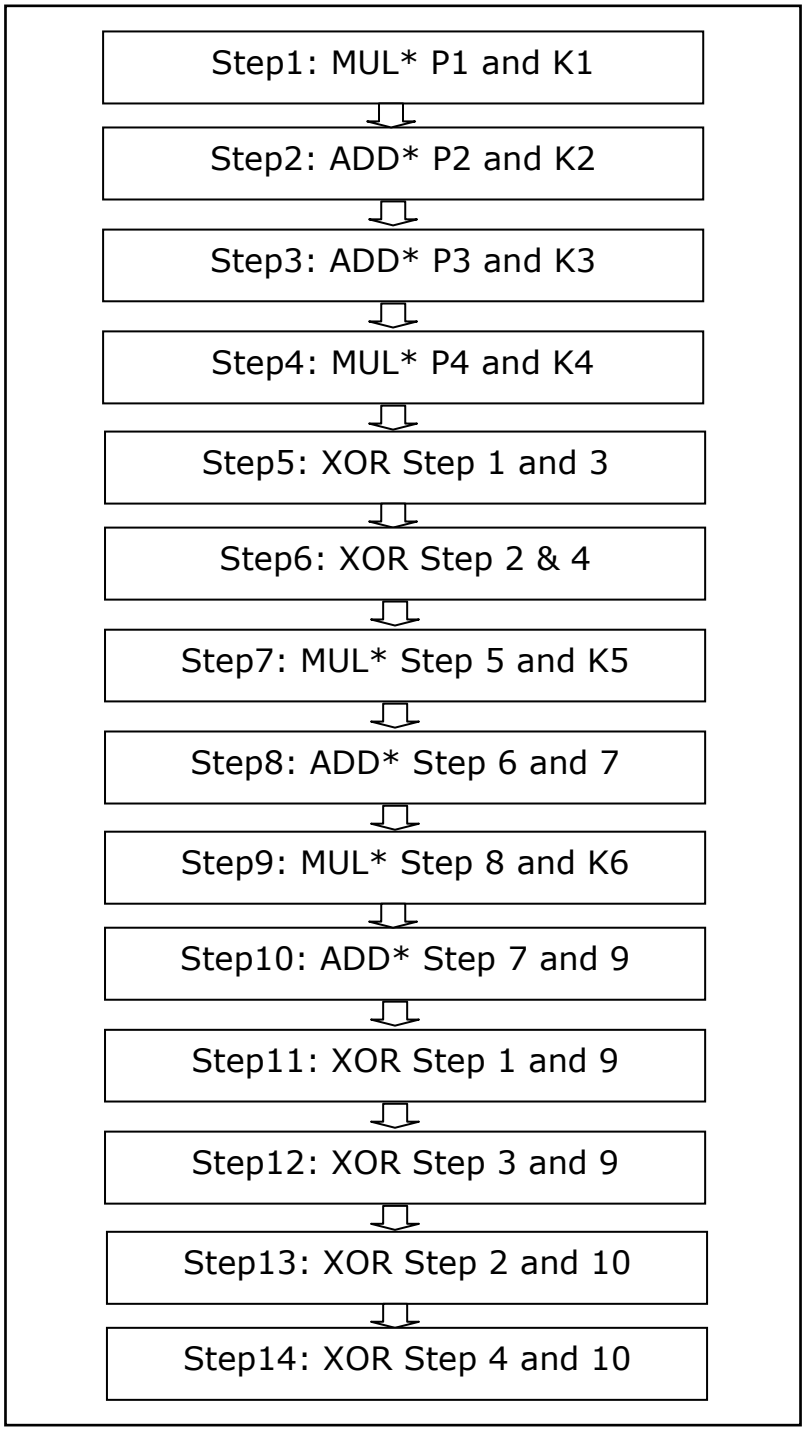


Fig: Details of One Round in IDEA

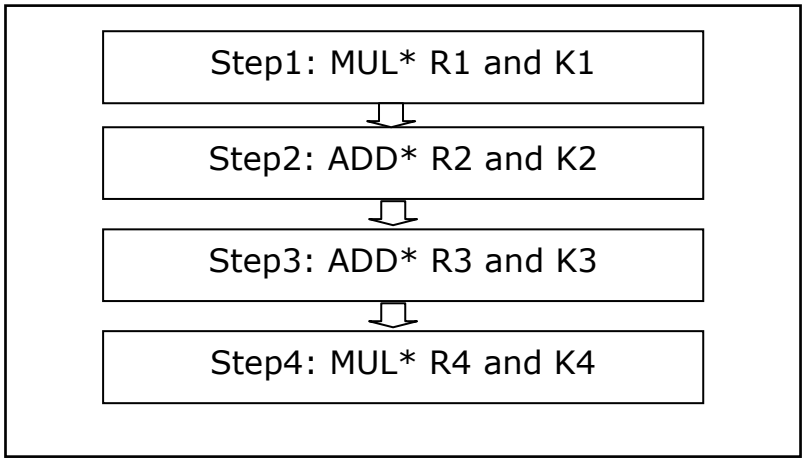


Fig: Details of Output Transform in IDEA

Summary:

- ✓ IDEA is a strong encryption algorithm
- ✓ Simple binary operations (XOR, ADD, MULT)
- ✓ Difficult to crypt analyze (because of 3 operations, use of 16 bits in the manipulations, and number of rounds)
- ✓ Efficient in software (simple operations, no convoluted permutations, same algorithm for encryption and decryption)
- ✓ Efficient in hardware (simple modules, simple operations on 16-bit registers)
- ✓ Used in commercial products (PGP and some standards)

BLOWFISH:

- ✓ Developed by Bruce schnier
- ✓ It has got the reputation of being very strong symmetric key cryptographic algorithm

Features:

- ✓ **Fast:** Encryption rate on 32 bit microprocessor is 26 clock cycles/sec
- ✓ **Compact:** Can execute in less than 5 kb of memory
- ✓ **Simple:** Uses only primitive operations i.e. Addition XOR and table lookup
- ✓ **Secure:** Having Variable key length up to maximum of 448 bits

- ✓ Suitable for the applications where the key remains constant for a long interval of time

Working:

Encrypts 64-bit block with a variable length key it contains major two operation

1. **Key Expansion:** This process expands key up to 448 bit long to sub key totaling 4168 bits

2. **Data Encapsulation:** This process involves iteration simple function 16 –times. Each round contains key- dependant permutation and key dependant substitution

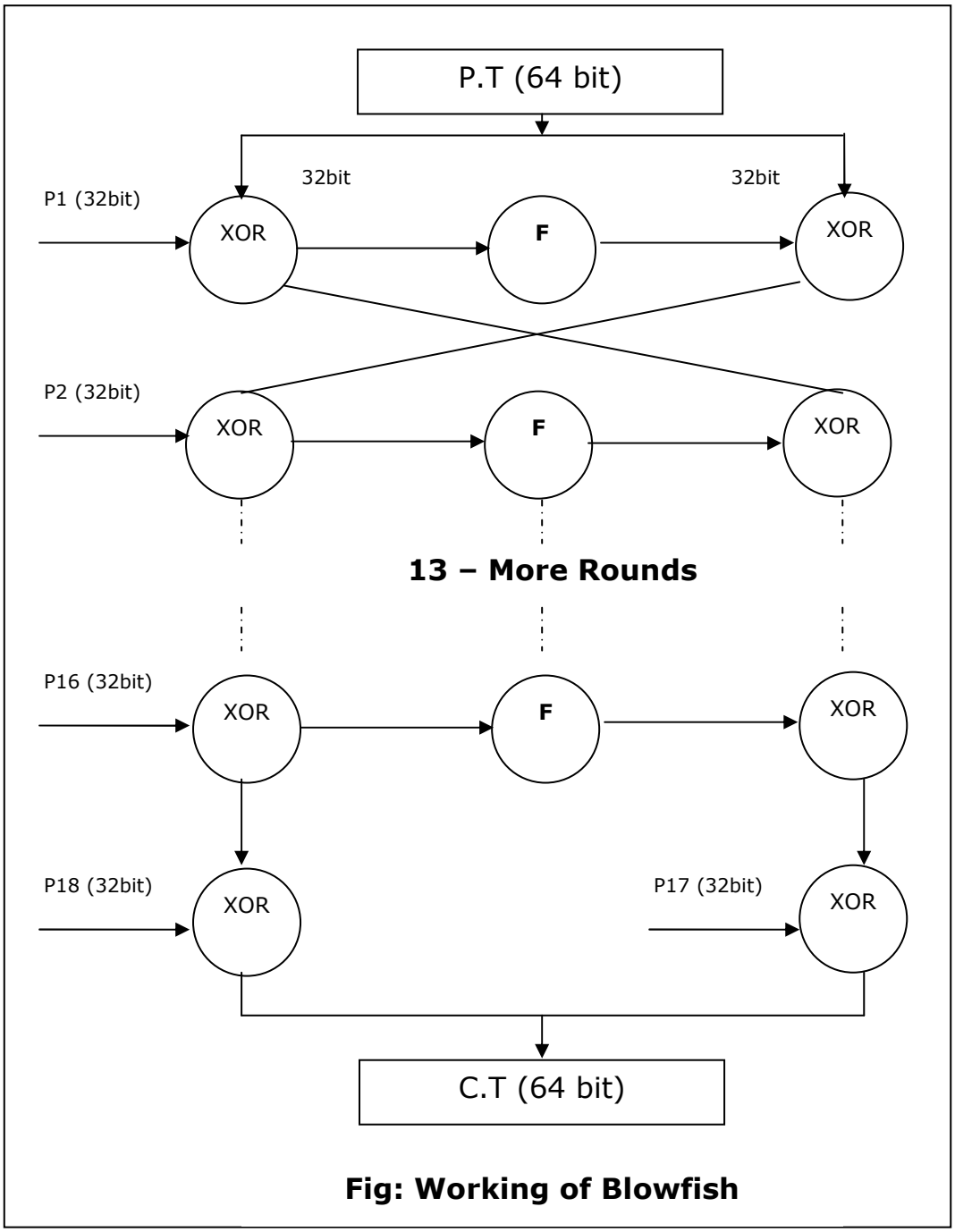


Fig: Working of Blowfish

Encryption of 64 bit input PT block is as follows

1. Divide X in two parts XL and XR of equal size i.e. 32 bit each
2. for I = 1 to 16

   XL = XL XOR P(i)

   XR = f(XL) XOR XR

   SWAP XL, XR

   Next i
3. SWAP XL, XR
4. XL = XL XOR P18
5. XR = XR XOR P17
6. Combine XL and XR back into X to produce CT as shown in above fig

TWO FISH:

- ✓ Developed by Bruce Schnier and teammates (in 1993)
- ✓ Symmetric key block cipher
- ✓ PT Block size is of 128 bits
- ✓ Key Size up to 256 bits i.e. 128, 192 and 256
- ✓ No. of rounds -16 uses basic Fiestel Network
- ✓ It was one among the Five Finalist for AES

AES (Advanced Encryption Standard):

- ✓ In 1990 US government wanted to standardized algorithm which was universally accepted
- ✓ Many proposals submitted after a long debate Rijndael was accepted
- ✓ Rijndael was developed by Joan Daeman and Vincent Rijmen (From Belgium)
- ✓ Out of 15 proposals only 5 are sort listed in August 1999
- ✓ In October 2000 Rijndael was accepted as final selection for AES

Features of AES:

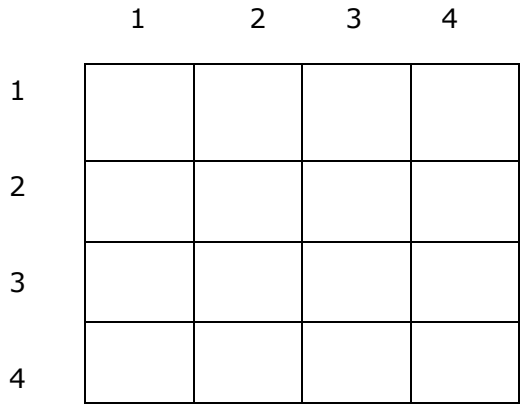
- ✓ Symmetric and parallel structure- Gives implementer flexibility and stand up well against cryptanalysis attack
- ✓ Suitable for modern RISC processors
- ✓ Suited for smart cards

Working:

- ✓ PT Block size : 128,192,256
- ✓ Key length: Independent of selected PT block and are organized in variable sizes(16,24 and 32 bytes)
- ✓ Rijndael /AES consist of 10,12 or 14 rounds and each round consist of 4 steps
- ✓ Does not have the structure of a classical *feistel*cipher
- ✓ treats data in 4 groups of 4 bytes
  operates an entire block in every round

Designed to be:

- ✓ resistant against known attacks
- ✓ speed and code compactness on many platforms
- ✓ Decryption algorithm different than the encryption



(a) Initial PT/Key block

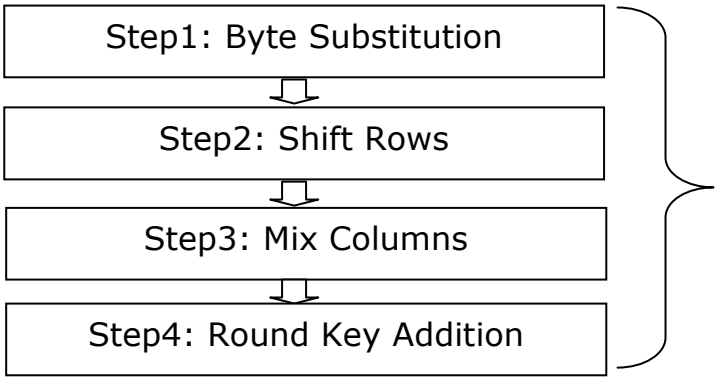
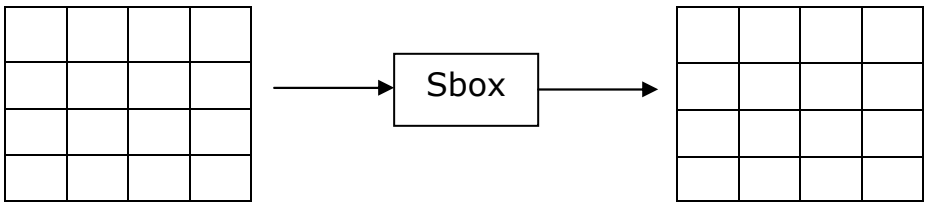


Fig: Broad Level Steps

Step1: Byte Substitution:

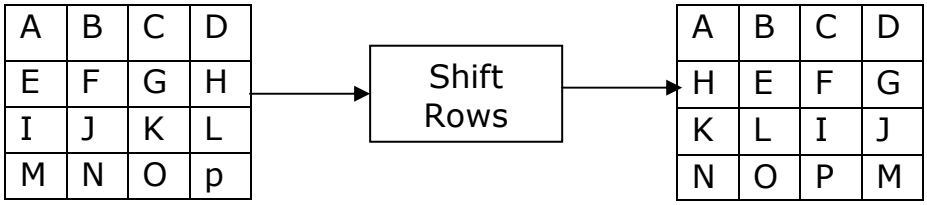
S – Box technique is used similar to Des input text passes through S-box and corresponding text is generated



Step1: Byte Substitution:

Step2: Shift Rows:

In this step first row is untouched the other three rows are shifted by a variable amount as shown below



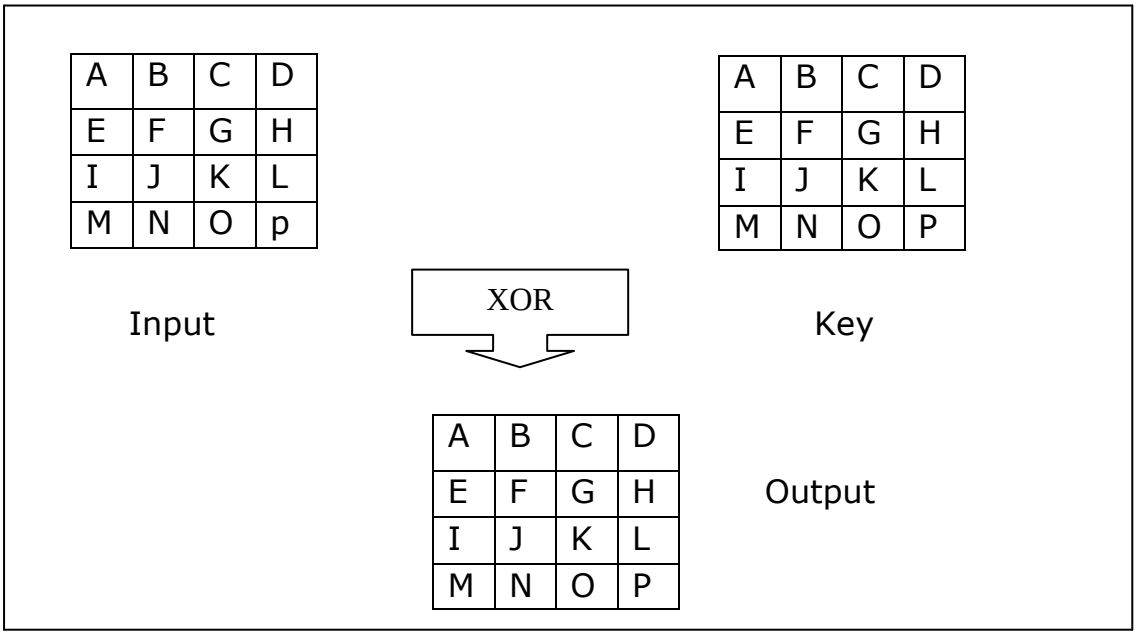
Step2: Shift Rows

Step3: Mix Columns

In this step 4 bits of every column is mixed in linear fashion, its not possible to depict

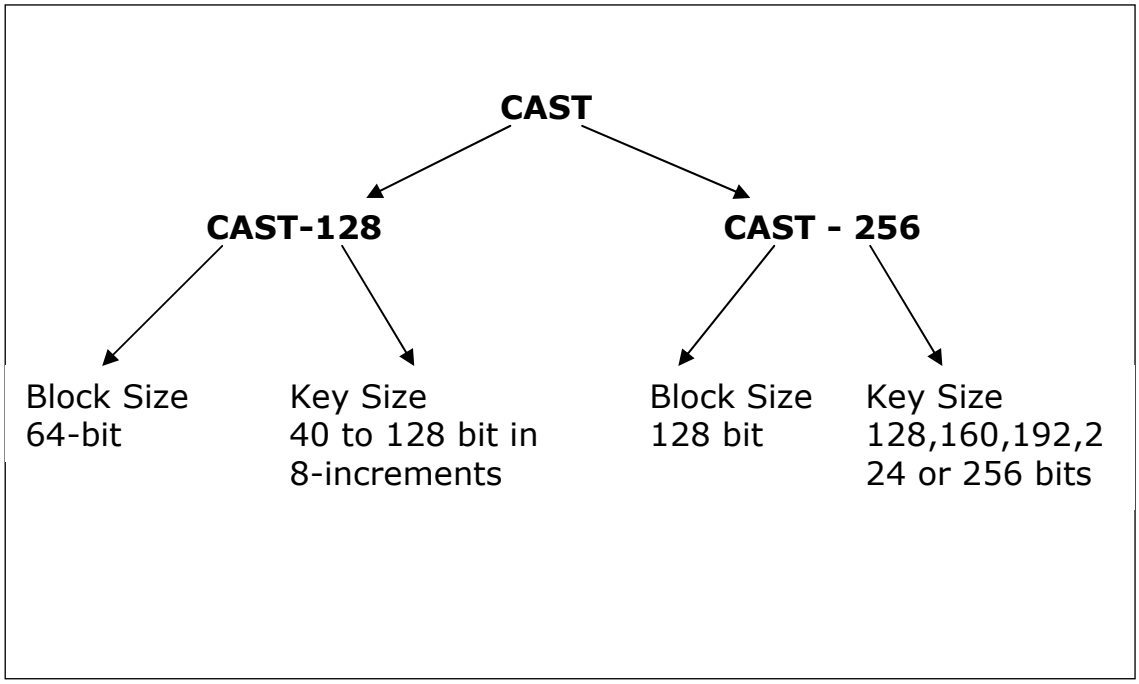
Step4: Key Addition:

Each Key byte is XORed with the corresponding input byte and result become CT for this round



CAST:

- ✓ CAST takes its name from the initials of its designers- **C**arlisle **A**dams and **S**tafford **T**avares
- ✓ CAST cipher is patented by entrust Technologies
- ✓ They allow the royalty free uses of the cipher to any one
- ✓ CAST uses DES like substitution permutation Network(SPN)
- ✓ CAST also posses number of other desirable cryptographic properties including avalanche and strict avalanche(SAC)



Stream Cipher Structure:

- ✓ Process the message byte by byte (as a stream)
- ✓ Typically have a (pseudo) random **stream key** that is XORed with plaintext bit by bit
- ✓ Randomness of **stream key** completely destroys any statistically properties in the message

- ✓ Ci= Mi XOR Stream Key i

- ✓ The simplest encryption/decryption algorithm possible!

- ✓ A stream cipher is similar to the one-time pad discussed a few lectures back

- ✓ The difference is that a one-time pad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream generated based on a secret key

- ✓ One must never reuse stream key

- ✓ Otherwise can remove effect and recover messages

- ✓ XOR two cipher texts obtained with the same key stream to obtain the XOR of the plaintexts –enough to know about the structure of the files to effectively attack them

Pseudo Random Number generation Standard:

- ✓ As we know random numbers are extremely crucial in cryptography
- ✓ A series of numbers is said to be random if a given number n of series then we can not predict what would be the $n+1^{th}$ number in concerned series
- ✓ We feel that computers can generate random number even some programming language provide facility to generate random numbers
- ✓ However this is not quite correct method, random number generated by computers are not truly random over a period of time as we can predict them

- ✓ Prediction of computer generated random number is possible because computers are rule based machines and they have a finite range of generating random number

- ✓ So to tackle the problem of generating truly random number by computers we use some external means with computers

- ✓ The process of generating truly random number by using some external means is called as pseudo random number generation

There are following ways for generating pseudo random number

1. Monitor hardware that generate random Data:

Best but costliest approach the generator is generally an electronic circuit which is sensitive to some random physical event such as diode, noise or atmospheric changes. This unpredictable sequence of event can be transformed into a random number

2. Collect Random Data from user Interaction:

In this approach user interaction such as keyboards, key press, mouse movements are used as input to generate random number

3. Collect Random Data from inside the computers:

This approach involves the collection of data from inside the computers which is hard to predict, this data can be systems clock, total no. of files on disk, the number of disk block the amount of free and unused memory etc

Netscape navigator is using the system clock and some other attributes to generate random number which form the basis of SSL protocol

Pseudo Random Number generator (PRNG)

Linear Congruencies generator:

- ✓ This is the first algorithm proposed by Lehmer (Lehm-51), it consist of following parameters

M	-	Modulus	m>0
a	-	the number	0<a<m
c	-	The increment	0<=c<m
X0	-	Seed / Starting value	0<= X0 < m

- ✓ The sequence of random no. {Xn} is obtains with the following iterative equation

$$X_{n+1} = (aX_n + C) \bmod m$$

- ✓ If m, a, c and X0 are integers then this technique will produce a sequence of integers with each integer in the range of 0<=Xn < m

- ✓ We would like to keep m to be very large so that there is the potential to produce long series of distinct random numbers

- ✓ The strength of linear congruential algo depends on the selection of multiplies and modulus

- ✓ If opponent knows linear congruential method is used then there is possibility of discovering all subsequent numbers

Cryptographic Generators

For cryptographic applications it makes advantage of encryption algorithm available to produce random number

Cyclic Encryption:

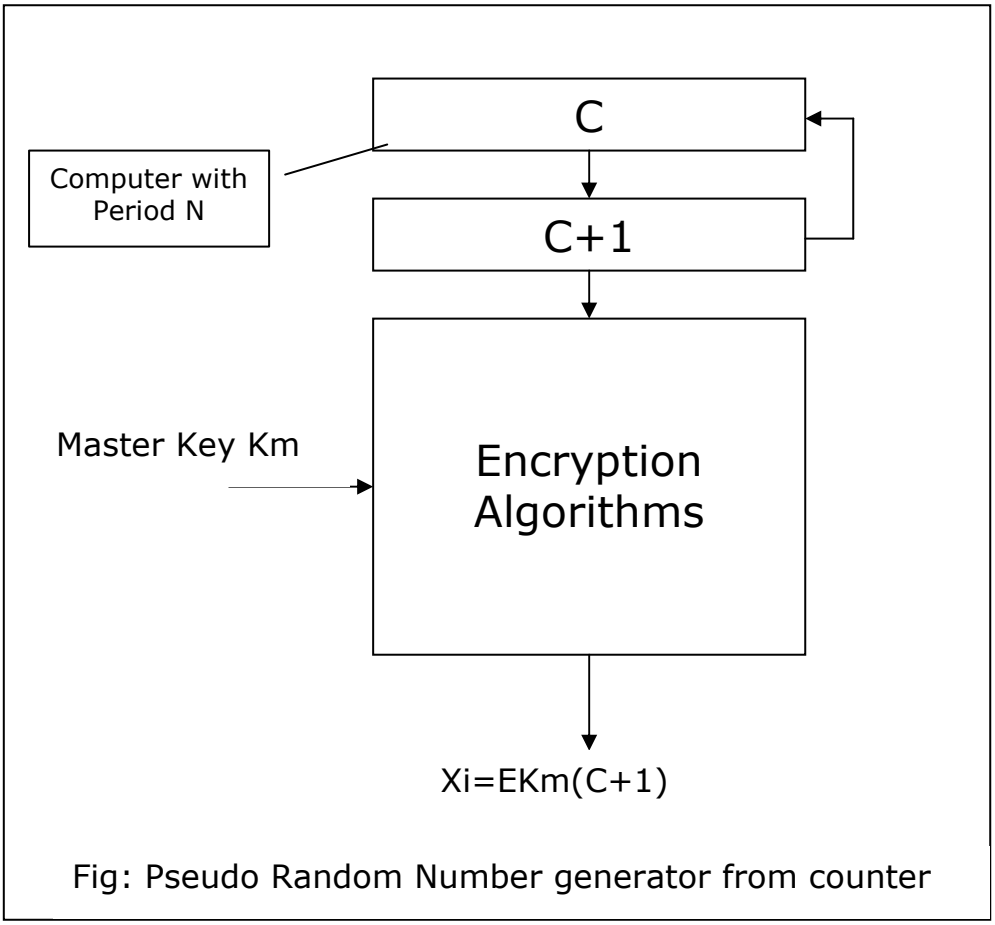


Fig: Pseudo Random Number generator from counter

- ✓ In this case procedure is used to generate session keys from master key.
- ✓ A counter with period N provides encryption logic
- ✓ After key is produced counter is incremented
- ✓ So pseudo random number produced by this scheme cycles through a full period $X_0, X_1 . . . . X_{n-1}$

ANSI X9.17 PRNG:

Cyclic Encryption:

- ✓ IS one of the strongest PRNG method shown below which makes use of triple DES (DES-III)

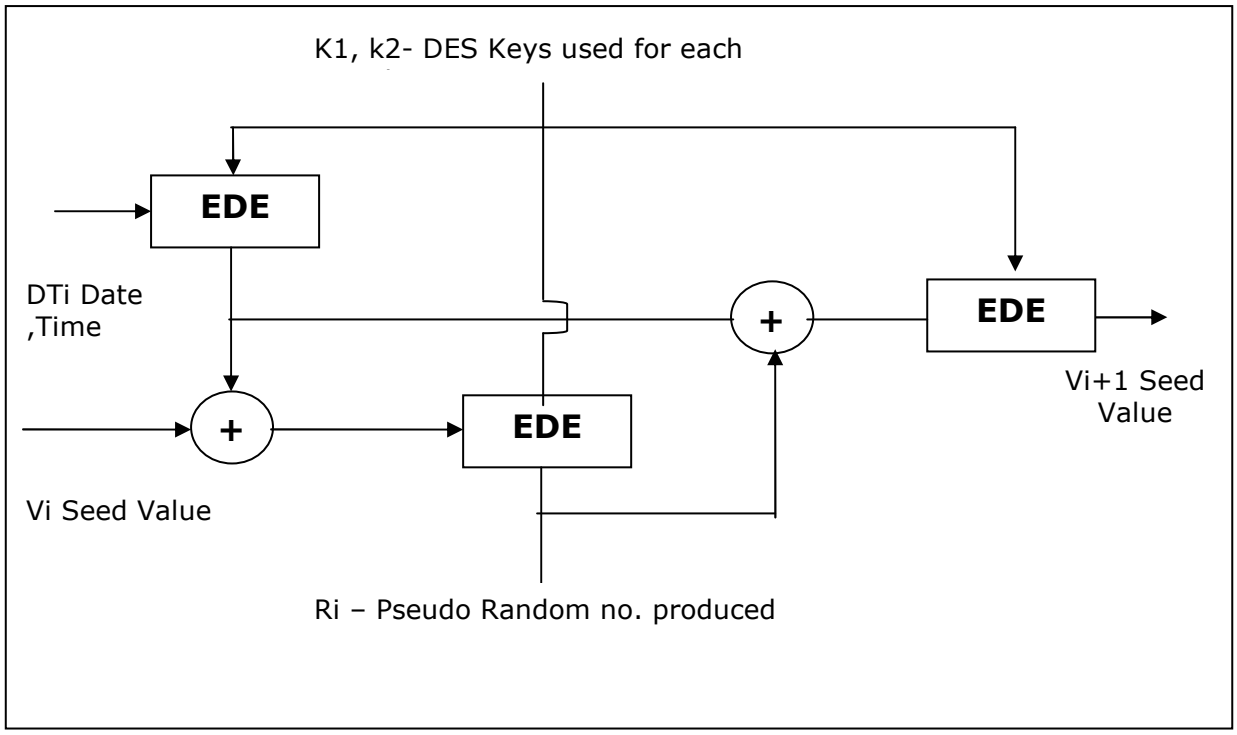


Fig: ANSI X9.17 PRNG

Inputs: Two inputs drive the generator and they are

1. 64 – bit representation of current data and time which is updated after each number generation
2. Other is 64 –bit seed value initialized to some arbitrary value

Keys: Generators makes use of triple DES encryption module and they make use of the same pair of 56 bit key which must be kept secret and used only for random number generation

Output: Consist of 64-bit PRN and 64 bit seed value

DTi – Date and time of ith round

Vi - Seed value of beginning of ith generation

Ri- Pseudo random number produces by ith gen stage

K1, k2- Des keys used for each stage

Random No is Given by:

$$Ri=EDE_{k1k2}[vi \ \ XOR \ \ EDE_{k1k2}\ [DTi]]$$

Seed value is given by:

$$V_{i+1}=EDE_{k1k2}[Ri \ \ XOR \ \ EDE_{k1k2}\ [DTi]]$$

Stream Cipher Design

- ✓ Key stream should have a large period –a pseudorandom number generator uses a function that produces a **deterministic** stream of bits that eventually repeats
- ✓ Key stream should approximate the properties of a true random number generator
- ✓ Same frequency of 0 and 1
- ✓ If treated as a stream of bytes, all 255 values should occur with the same frequency
- ✓ Key should be long enough to protect against brute-force attack
- ✓ At least 128 bits
- ✓ **Advantage** over block ciphers: generating the stream key is much **faster** than encrypting and decrypting and less code is needed

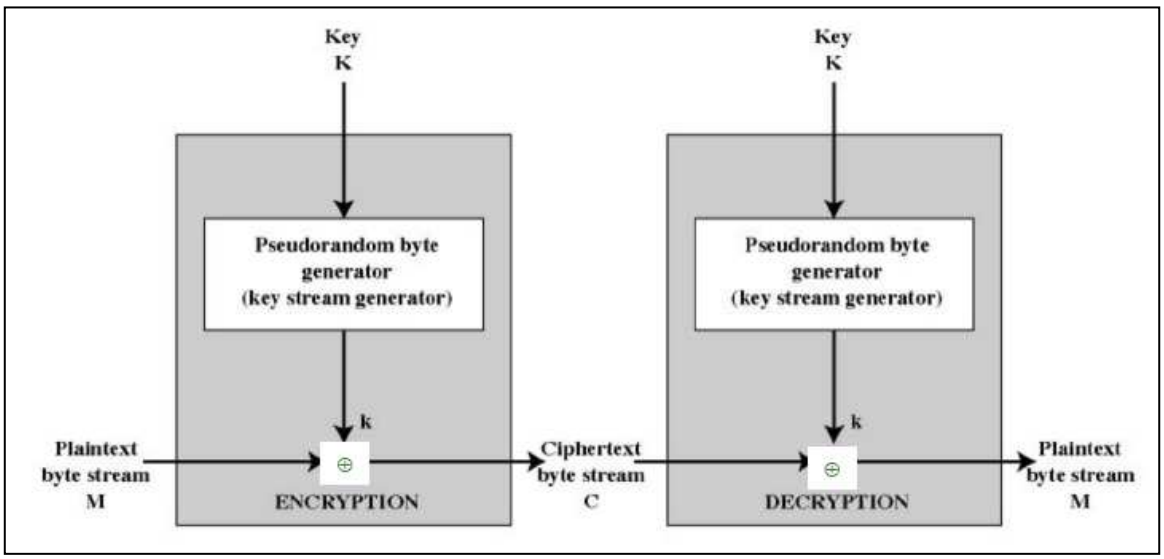


Fig: Stream Cipher Diagram

RC4 Stream Cipher

- ✓ This is the most popular symmetric stream cipher
- ✓ Designed by Rivest for RSA Security
- ✓ Used in SSL/TLS (Secure Sockets Layer/Transport Layer Security) standards for secure communication between Web browsers and servers
- ✓ Used in WEP, part of the IEEE 802.11 wireless LAN standard
- ✓ RC4 was kept as a trade secret by RSA Inc but got anonymously posted on the Internet in 1994

RC4 Algorithm:

- ✓ Key length is variable: from 1 to 256 bytes
- ✓ Based on the key initialize a 256-byte state vector **S**: S[0…255]
- ✓ At all times S contains a permutation of the numbers 0, 1, …, 255
- ✓ For encryption and decryption a byte **k** is selected from S and the entries in S are permuted

RC4 Initialization of S:

- ✓ Initially S[i]=i, i=0,1,…,255 and create a temporary vector T of length 256 –the key K is copied to T (if K has less than 256 bytes, repeat K as many times as necessary to fill T)

```
For i=0 to 255 do
    S[i]=i;
    T[i]=K[i mod keylen]
```

- ✓ Use T to produce the initial permutation of S

```
j=0;
For i=0 to 255 do
    j=(j+S[i]+T[i]) mod 256;
    Swap(S[i],S[j]);
```

- ✓ Input key is never used after this initialization

- ✓ RC4 Key stream Generation:

```
i,j=0;
While(true)
    i=(i+1) mod 256;
    j=(j+S[i]) mod 256;
    Swap(S[i],S[j]);
    t=(S[i]+S[j]) mod 256;
    k=S[t];
```

- ✓ *Encryption*: XOR k with the next byte of the plaintext
- ✓ *Decryption*: XOR k with the next byte of the ciphertext
- ✓ There is no practical attack against RC4 with reasonable key length such as 128 bits
- ✓ **Strength of RC4**: there has been a report of a problem in the WEP protocol (for 802.11 wirelesses LAN) –the problem is not with RC4 but rather with the way in which keys are generated to use as input to RC4

RC5 Algorithm:

- ✓ Symmetric key encryption algorithm developed by Ronald Rivest

Features of RC5 are

- ✓ Fast : Since uses primitive operation such as addition XOR and shift
- ✓ It allows variable no of rounds and variable length of keybits
- ✓ Requires less memory
- ✓ Suitable for modern processors, smart cards as well as devices with less memory

Working:

1. **Basic Principles:**

   - ✓ Word size (PT block size) in bits: RC5 encrypts two word blocks at a time and they are of 16, 32, 64 bits
   - ✓ No. of rounds: 0 to 255
   - ✓ No. of 8 bit bytes (octets) in the key are 0-255
   - ✓ i.e. PT block size can be of 32,64 or 128 bit (since 2-word blocks are used)
   - ✓ Output resulting from the RC5 i.e. CT has same size as the input plaintext
   - ✓ RC5 allows variable values in three parameters i.e. PT length, No. of rounds , Key length so at any instance RC5 algo is denoted as RC5- w/r/b or $RC5_{wrb}$
   - ✓ Where w- Word size

     r- No. of rounds

     b- No. of 8 bit bytes in key

- ✓ RC5-32/16/16 means

  RC5- with block size of 64 bit as it uses two word blocks at a time
  RC5- with 16 rounds

  RC5 – with 16 bytes (i.e. 128 bits) in key values

- ✓ Rivest suggested RC5 – 32/12/16 as minimum safety version

2. Principle of operation:

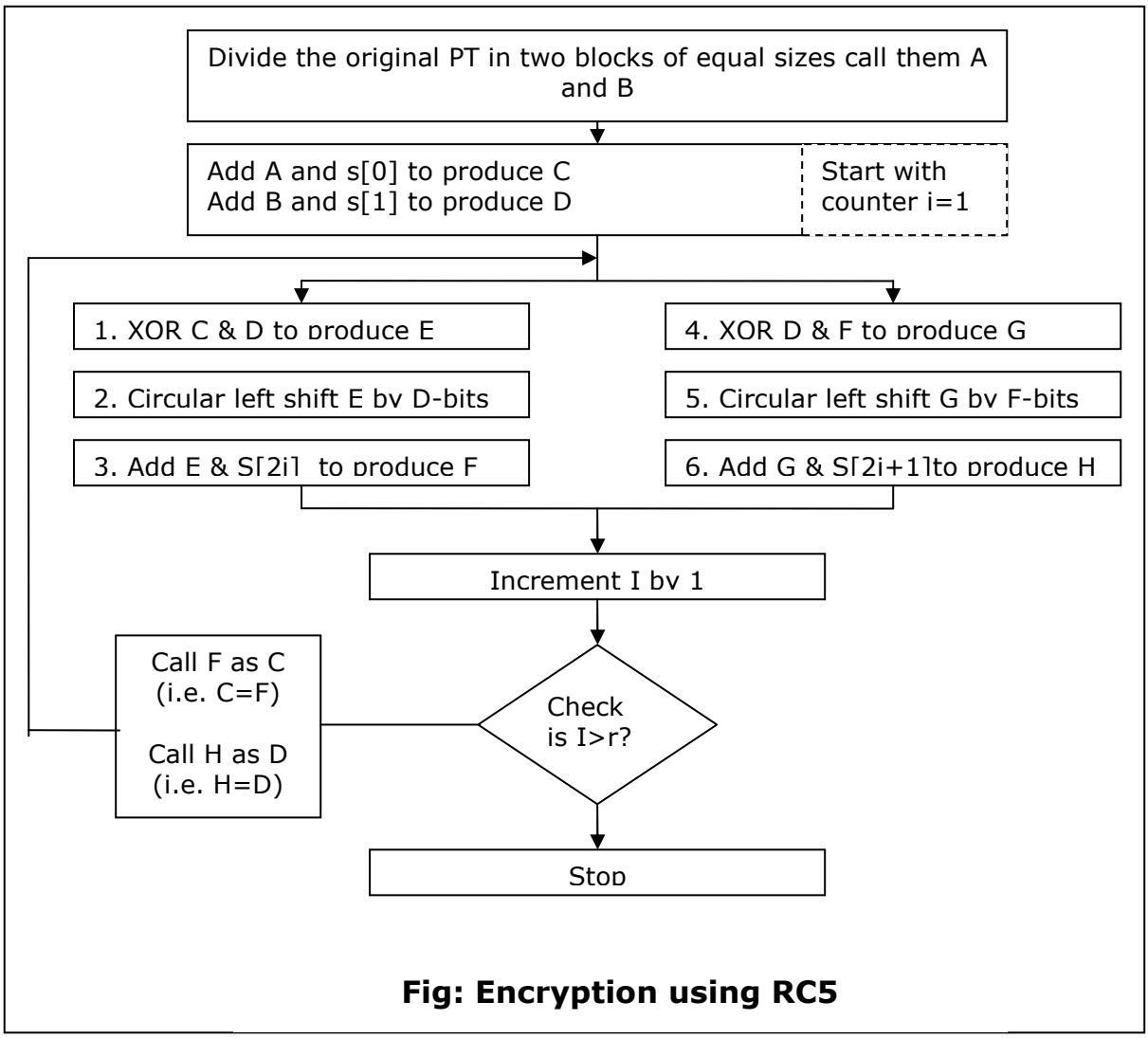


Fig: Encryption using RC5

- ✓ As shown in above fig. first two steps are of one-time initial operation. Input PT is divided in to 32 bit blocks A and B

- ✓ First two sub keys s[0] and s[1] are added to A and B and produces C & D
- ✓ Now the rounds will begin and in each round there are following operations
    1. Bitwise XOR
    2. Left Circular Shift
    3. Addition with the next sub key for both C and D

- ✓ As shown in fig. the output of one block is feedback to the input of next block which makes whole logic complicated for cryptanalyst to decipher

Step by step Algorithmic Details:

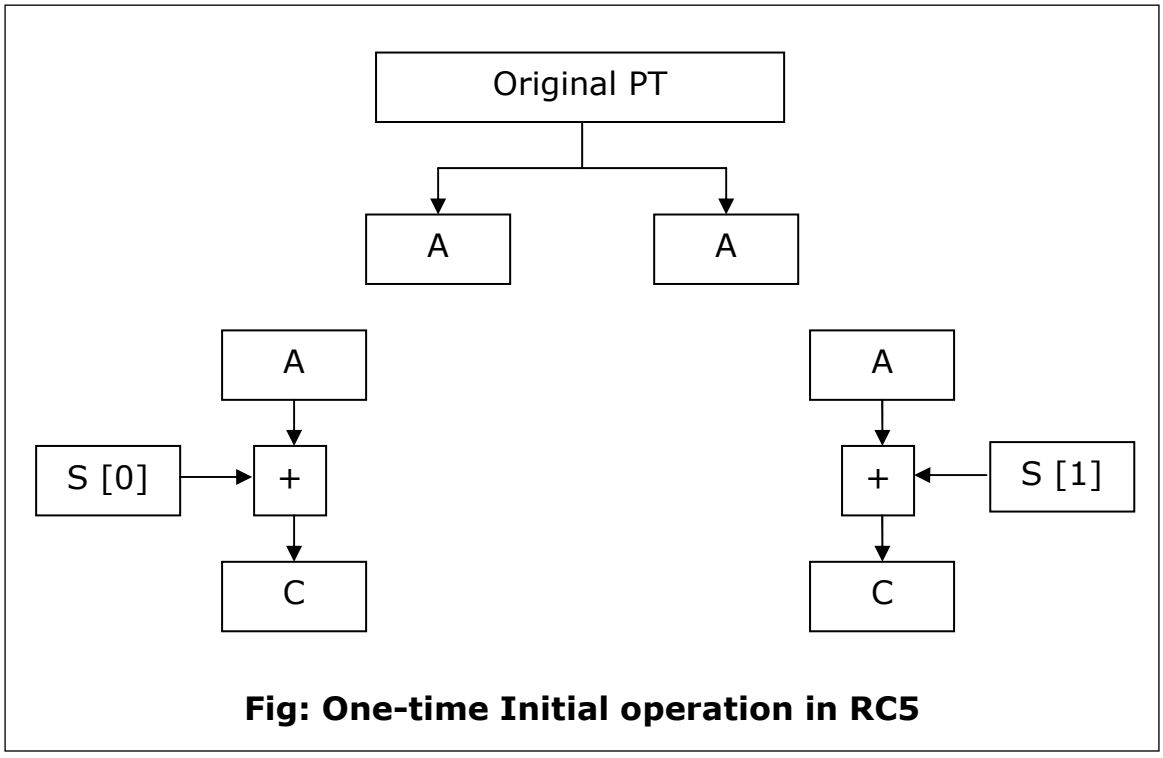
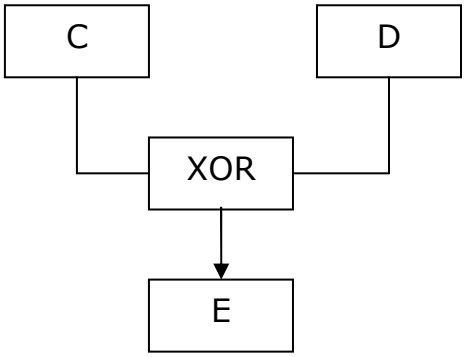
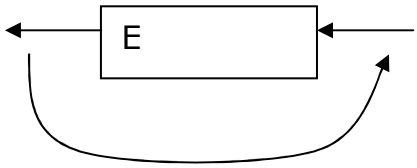


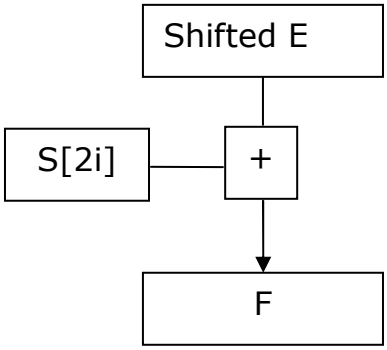
Fig: One-time Initial operation in RC5



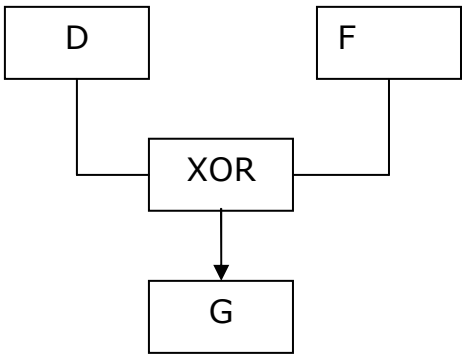
Step- I: XOR C and D
in first step of each round
to produce E



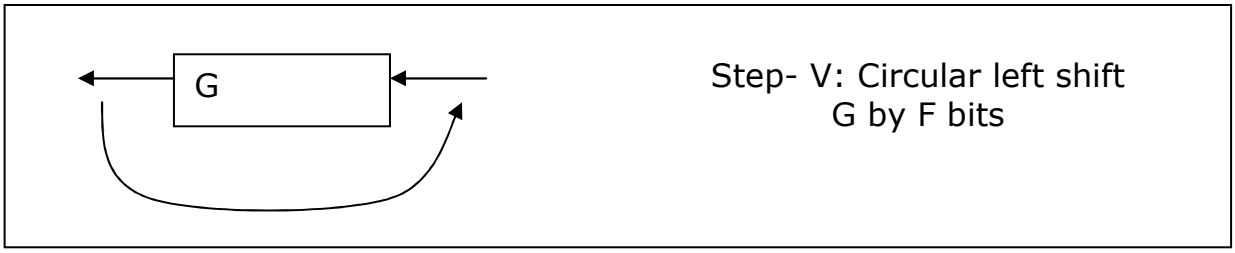
Step- II: Circular left shift
E by D bits

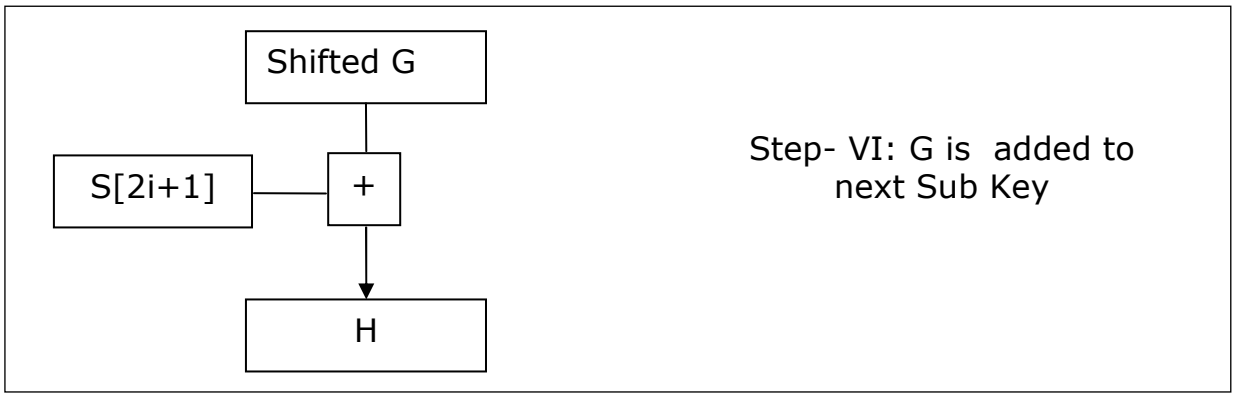


Step- III: E added to next
Sub Key



Step- IV: XOR D and F
To produce G





Step VII: Miscellaneous task

- ✓ In this step we check to see if all the rounds are over or not for this we perform the following steps
- ✓ Increment I by 1
- ✓ Check to see if i<r perform following

```
I=i+1
If i<r
Call F as C again
Call H as D again
Go back to step1
Else
Stop
End if
```

Mathematical representation of RC5 (Encryption):

A=A+S[0]

B=B+S[1]

For i=1 to r

A=((A XOR B) <<< B)+S[2i]

B=((B XOR A) <<< A)+S[2i+1]

Next i

Mathematical representation of RC5 (Decryption) :

For I = r to 1 step-1

A= ((B-S [2i+1] >>> A) XOR A

B= ((A-S [2i] >>> B) XOR B

Next I

B= B-S [1]

A= A-S [0]

Sub key creation:

✓ Sub key creation is two step process

1. First step the sub keys (denoted by s[0] ,s[1],_ _ _ _) are generated

2. Original key is called as L in second step the sub key (s[0], s[1],_ ___) are mixed with corresponding sub-portion or original key i.e. [0],L[1]_ _ _ _ as shown

Question:

1. Explain the working of IDEA by drawing block diagram and details of round steps
2. Explain output transformation process of IDEA
3. Explain the working of Blowfish by giving the details of bits and bytes utilized for block and key
4. What is difference between random no and pseudo random no ? also explain the role of random no in cryptography
5. Explain different methods used for generating pseudo random numbers
6. Explain Linear congruencies method for generating pseudo random number
7. What is cryptographic random no generator? How they are different from normal random number generators
8. Explain cyclic method and ANSI X9.17 for generating random numbers
9. Explain working of RC5 with details of each and every step also give mathematical representation of encryption and decryption
10. Explain working of RC4