SECRET KEY CRYPTOGRAPHY

DR. O.I. ADELAIYE

WHY IS IT IMPORTANT

- Secret key cryptography: Common uses
  - Confidentiality: sender and receiver share a secret and use it to communicate
  - Secure storage: encrypt content before storing it
  - Authentication: share a key and prove that you know it when being authenticated
  - Integrity: instead of having a simple cleartext checksum, encrypt it !
    - becomes Message Integrity Check or Code

BLOCK VS STREAM CIPHER

- Two different techniques (algorithms)
  - Stream ciphers: can encrypt any size
    - Are based on random number generators
    - The generated random number string has the same size as the data
  - Block ciphers: encrypt fixed size blocks
    - The data to be encrypted is split into blocks of a predefined size
    - If the block is smaller than that size, then padding bits are added before encryption (e.g. add zeros at the end)
    - Example: DES and IDEA use 64 bit blocks, AES uses 128 bit block

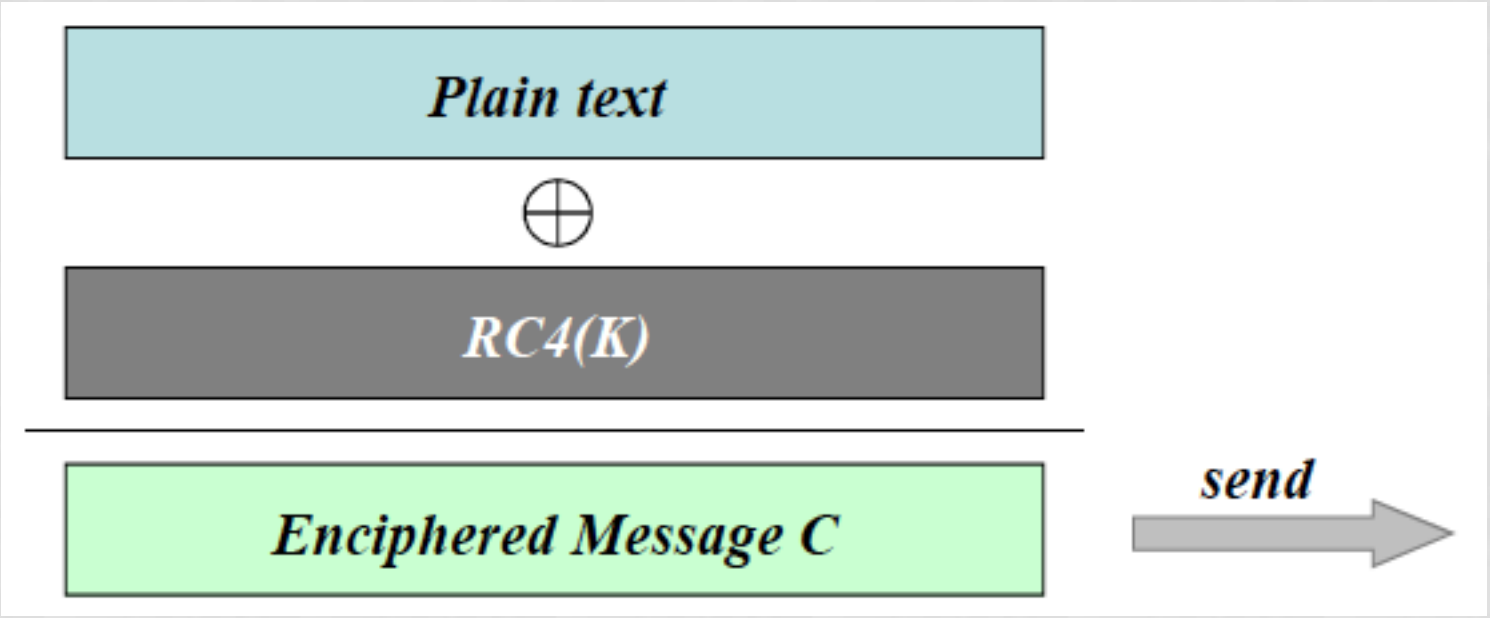
STREAM CIPHER

STREAM CIPHER

- Stream cipher
  - Generates a random string: Keystream
    - Not to be confused with the shared Key
  - XORs it with the message to be encrypted
  - The random string is not really random
    - Pseudo-random
  - The pseudo-random string is used only once
    - One-time pad
  - RC4 is one of the most widely known stream ciphers
    - Used in WEP and in one version of WPA (TKIP)
    - Designed in 1987 and leaked out in 1994
    - Ron's (Rivest) Code?
    - Rivest Cipher 4?

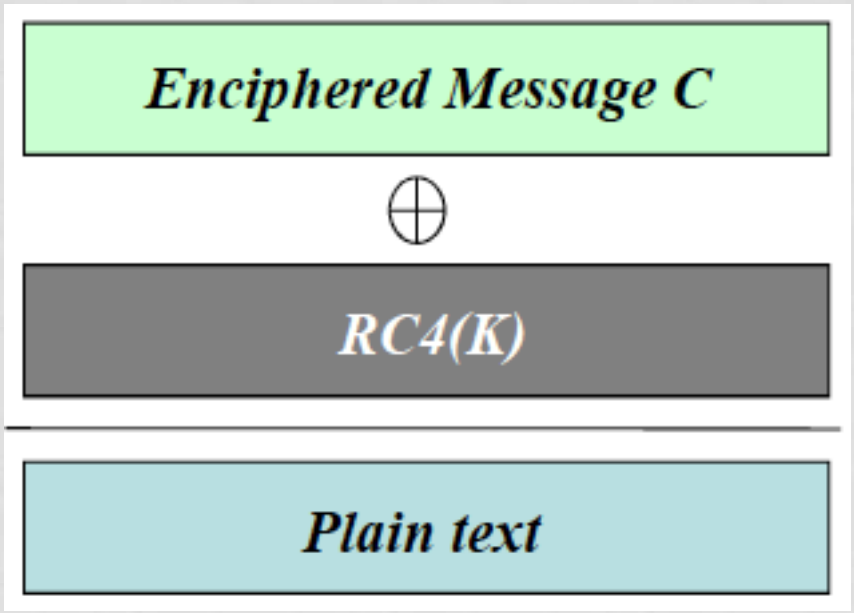
STREAM CIPHER

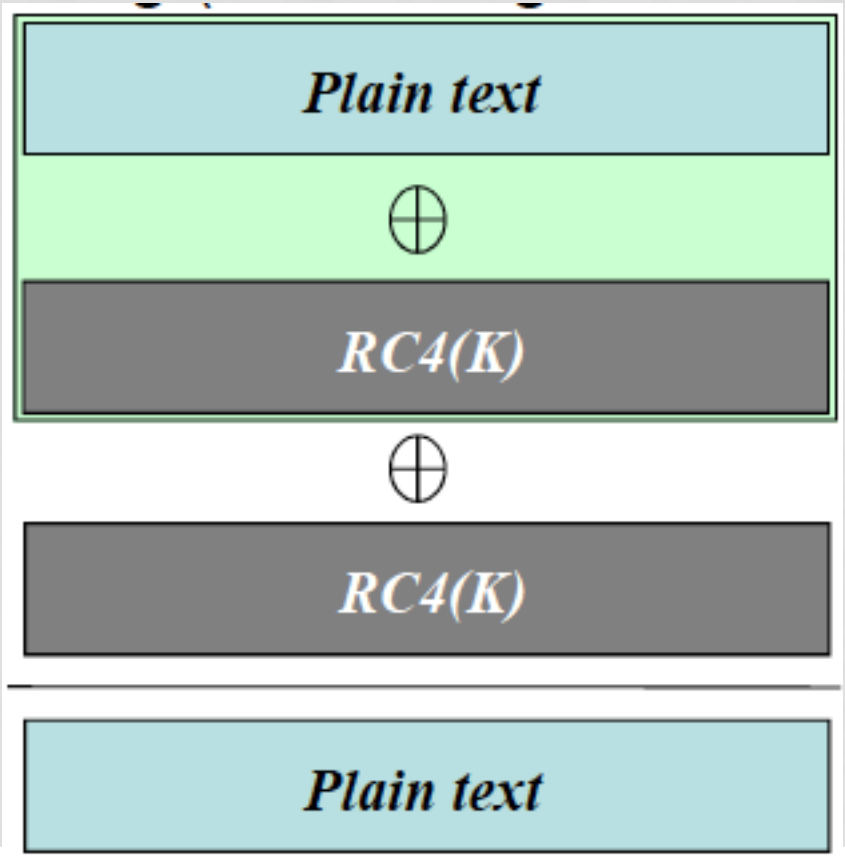
- RC4
  - Simple to deploy
  - Off-line phase
  - 1 to 256 bytes key
    - 8 bytes key is used in WEP(Wired Equivalent Privacy)



STREAM CIPHER

- RC4
  - Decipher by XORing the enciphered message with the same pseudo-random string (can be regenerated using the same key K)





STREAM CIPHER

- RC4
  - Very easy to implement
  - Light-weight
  - But light security !
  - The first few generated bytes give insight on the used key
    - Avoid using the first 256 bytes
  - Not used anymore in new " secure" systems
  - Still a good pseudo-numbers generator though

BLOCK CIPHER

BLOCK CIPHER

- Encryption of a fixed length block
  - Defined by the encryption algorithm
- The size of a block should be small enough to avoid wasting resources
  - Imagine a block size of 1GB !
- The size of a block should be big enough to prevent crackers from building dictionaries
  - Imagine a block size of 1 bit !
    - A message 1100 will be deciphered as either 0011 or 1100
    - The number of possibilities is 2^{block_size}
    - The cracker does not need to know the key
- 64 bits was considered to be a reasonable block size in 1970s: DES and later on IDEA
  - The more recent AES uses 128 bits

HOW ENCRYPTION WORKS?

- Needs a technique that maps inputs (cleartext) to outputs (ciphertext) and vice versa
  - Has to be one-to-one, otherwise a ciphertext could be decrypted to many possible cleartexts
- Two techniques
  - Permutation: changes positions of bits
    - 1^{st} becomes 7^{th}, 2^{nd} becomes 5^{th} , etc.
    - n! permutations (where n is the number of bits), n! = n.(n-1).(n-2)… 1
  - Substitution: replaces n bits by n other bits
    - Works like a dictionary
    - E.g. 0000 is replaced by 1010; 0001 by 1100; 0010 by 1001, etc.
    - 2^n possible outputs for each input
- Can be combined as many times as needed

LETS CREATE A BLOCK CIPHER

- Gordon's cipher: block size is 16 bits, key size is 1 bit
  - Combines both substitution and permutation to create a cipher
  - Start with substitution
    - Divide the block (say 16 bits) into smaller chunks (2 bits each)
    - Apply the substitution on each chunk

      00 > 10

      01 > 11

      10 > 00

      11 > 01
  - Then, apply permutation: reorder bits
    - 1,2,3,4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16 becomes
    - 2,4,6,8,10,12,14,16,15,13,11,9 , 7 , 5, 3 ,1

LETS CREATE A BLOCK CIPHER

- Gordon's cipher: block size is 16 bits, key size is 1 bit
  - Oups ! the key has not been used !
  - XOR the key with each chunk (two bits) after the permutation
- Example: encrypt 11 00 10 01 00 11 00 11 with key " 1
  - After substitution: 01 10 00 11 10 01 10 01
  - After permutation:
  - After XOR with the key:
- Note that modifying one bit in the input > modify one or two bits in the output

LETS CREATE A BLOCK CIPHER

- This process is repeated many times (rounds)
  - Otherwise, one input bit impacts only two output bits
- Is this process reversible?
  - Is it important to be reversible?

DATA ENCRYPTION STANDARD (DES)

DES

- Published in 1977
  - By the National Bureau of Standards
    - Became the National Institute of Standards and Technology (NIST)
  - Encrypts 64 bit blocks into 64 bit blocks
  - Uses a 64 bit key
    - Not really! 8 bits are odd parity bits: a parity bit per octet: 64 - 8 = 56 bit key
  - Designed to be implemented in hardware
  - Based on IBM's Lucifer Cipher

DES

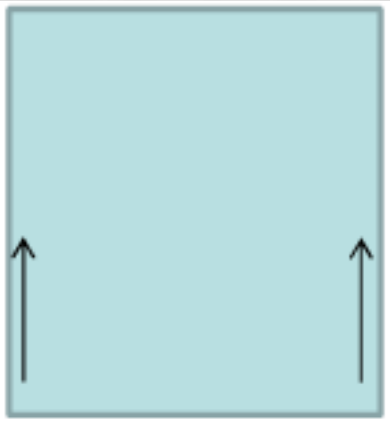
- Initial permutation, then 16 rounds, then final swap and final permutation
- In each round
  - The 56 bit key is used to generate a new 48 bit round key
  - A 64 bit input is converted into 64 bit output
- Decryption is done by running the rounds backwards
  - Initial permutation is still done first in decryption (cancels the final permutation done in encryption)
  - Final swap and permutation still done at the end

PERMUTATIONS

- Initial permutation
  - Place even columns before odd ones
    - 2,4,6,8,1,3,5,7
  - Rotate clockwise
- Final permutation: cancels the initial permutation

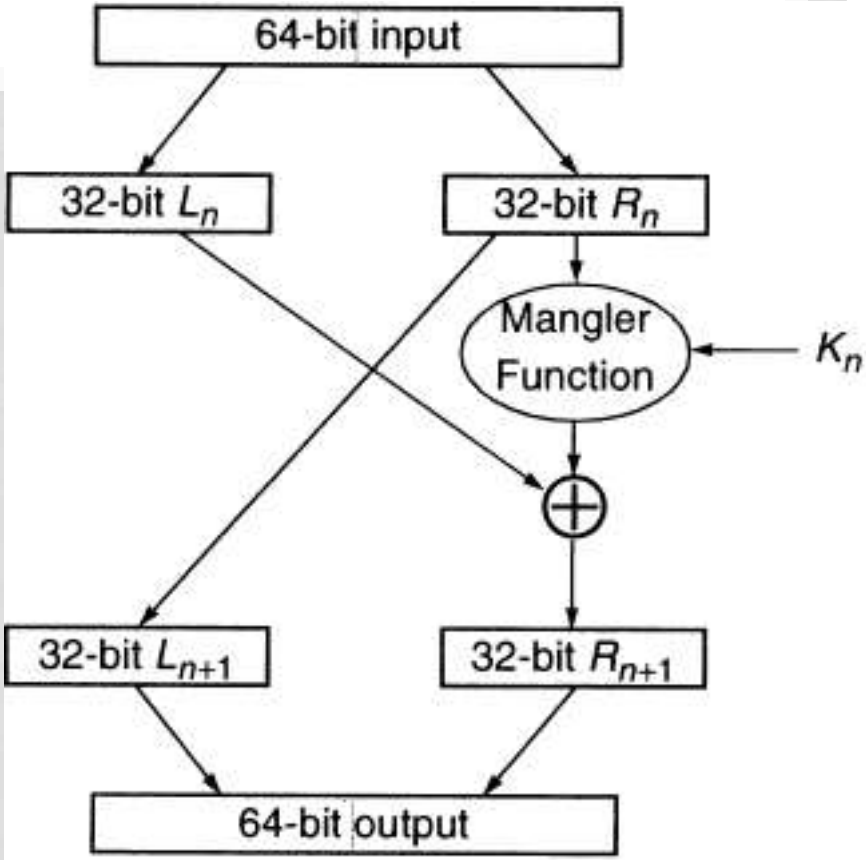
ROUNDS

- Reminder: the key has 8 parity bits
  - 56 useful bits: 1-7;9-15;17-23; … ;57-63
- Initialisation
  - Permutation
  - Split the key into two halves: C_0 and D_0
- Round *i*
  - Shift C_i and D_i by one or two bits to the left
  - Permute (again!) each of the two halves, the result of the permutation is the round key
    - The permutation on each half discards 4 bits of it > the round key is 56 But 48 bits only are selected (Left Choice Shift)



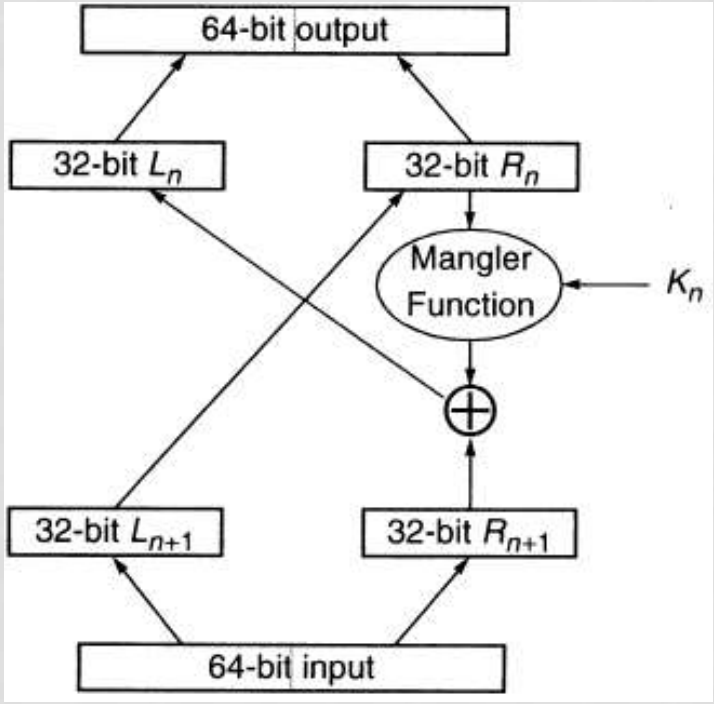
ROUNDS

- Round I
  - Input: L_i and R_i
  - Output: L_{i+1} and R_{i+1}
- $L_{i+1} = R_i$
- $R_{i+1} = f(R_i, K_i) \oplus Li$
- f is the mangler function
  - Takes a 32 bit R and 48 bit K
  - Split R into blocks of 4 bits
  - Expand each 4 bit block by adding the surrounding bits (i.e. +1 bit from each side)
    - Becomes 6 bits
  - Split K into blocks of 6 bits and XOR them with R's blocks
  - The outputs are eight 6 bit long blocks and each will be mapped using a S-box to 4 bits
    - Final output is 8x4=32 bits long and is the new R



DECRYPTION

- Decryption
  - Use exactly the same round algorithm
  - Input: R_{i+1} , L_{i+1} , K_i (note that the inputs are swapped)
  - Output: R_i , L_i



OBSERVATIONS

- Key size
  - The 8 parity bits are useless with not important use in practice
  - Lucifer's key was 128 bits and got reduced to 56 bits in DES
    - Justification: so DES could be implemented on a single chip
- Initial and final permutations useless
  - If DES is insecure without permutations then it is insecure with permutations
- Key permutations are useless
  - Same argument
- The rational behind the S-boxes is unknown
  - Justification: some secret attacks were considered during the design and they should not be made public
  - NSA contributed to them !

OBSERVATIONS

- In 1999, NIST advised not to use it in any new systems
- 3DES could be used
  - Three successive DES: encryption, decryption, then encryption
  - This makes a 3DES chip able to perform DES by setting all the keys to the same value
  - Two keys (1st and 3rd DES use the same), or three keys
- Similar concepts (rounds, per-round keys, etc.) are used in IDEA and AES

MODES OF OPERATION

WHY?

- DES and all other block encryption algorithms can encrypt a fixed size
  - What happens if you need to encrypt a message longer than one block?
- Modes of operation describe how to encrypt a message larger than one block
- The first four modes were introduced in 1981 for DES
  - ECB, CBC, OFB, and CFB
- CTR added in 2001 by NIST

ELECTRONIC CODE BLOCK (ECB)

- Simply encrypt each 64 bit block separately
  - Last block is padded to get 64 bits length
  - Encryption can be parallelized (same for decryption)
- Problems
  - Two identical blocks produce the same ciphertext
  - Blocks could be rearranged by the attacker
    - Swap a clerk's salary with the CEO's

OUTPUT FEEDBACK (OFB)

- Works as a stream cipher (computes a keystream)
  - An IV, b_0 is encrypted to generate b_1
  - Then, b_1 is encrypted to generate b_2 and so on
  - $b_0 \mid b_1 \mid b_2 \mid \ldots$ is the one-time pad
  - Has the advantages of stream ciphers
    - Can prepare the keystream in advance (offline)
    - As a byte arrives it can be sent (no need to wait for a complete 64 bit block to arrive)
    - If some bits are flipped en route in a given block, following block still can be decrypted correctly
- Problems
  - Known cleartext, ciphertext attack is fatal
    - XORing them gives the keystream
  - XORing two ciphertexts whose keystreams are equal could reveal one cleartext if the other is known

CIPHER FEEDBACK

- Similar to OFB but
  - Instead of having $b_{i+1} = E_{DES}\{b_i\}$, now $b_{i+1} = E_{DES}\{b_i \oplus Block_i\}$
    - $Block_i$ is the i^{th} block of plaintext
  - Avoids OFB's problems
- Problems
  - Keystream cannot be computed offline

COUNTER (CTR)

- Computes a keystream offline
- Generates a random iv then
  - Encrypts iv in order to generate b_0
  - Encrypts iv+1 in order to generate b_1 and so on
  - Can start decryption at any block
- Problems
  - Known cleartext, ciphertext attack
  - XORing two ciphertexts whose keystreams are equal

END