Block Ciphers & DES

CS 411/507 - Cryptography

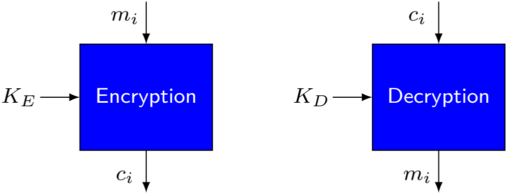
Erkay Savaş & Atıl Utku Ay

Department of Computer Science and Engineering
Sabancı University

October 30, 2023

Block Cipher: Definition

- A family of functions which maps n-bit plaintext blocks to n-bit ciphertext blocks;
  - n is called the <u>block-length</u>.
- The function is parameterized by a k-bit key K.
- It may be viewed as a simple substitution cipher with a large character size.



$$K_D = F(K_E)$$
$$K_E = F^{-1}(K_D)$$

Evaluating Block Ciphers

- Historical strength:
  - The longer it is exposed to public scrutiny, the higher the confidence level
- Key Size:
  - Effective key size defines an upper bound on the level of security of the cipher
  - While longer keys provides more security, they also impose additional implementation costs.
- Complexity:
  - Complexity of the mapping is good for the security
  - May be restrictive in terms of the efficiency.

Evaluating Block Ciphers

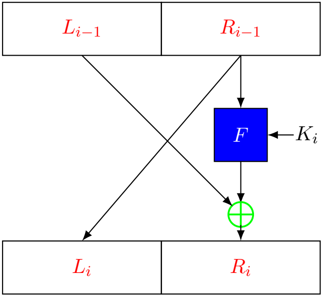
- Block Size:
  - The larger the block size the higher the security
  - Performance implications.
- Throughput:
  - Fast and easy to implement in hardware and software.
- Data Expansion:
  - Encryption should not increase the size of plaintext data.
- Error propagation:
  - Decrypting the ciphertext containing bit errors may result in various effects on the recovered plaintext.

DES Algorithm

- In 1976, the NBS (later NIST) released DES and a free license for its use.
- NSA reviewed and modified the original “Lucifer” which was an IBM design to make the DES.
- Became a standard in 1977 (replaced in 2001).
- Widely used especially in banking industry since.
- Biham & Shamir in 1990, showed an efficient cryptanalysis method (differential) to attack DES.
  - The attack is more efficient for DES variants with fewer number of rounds.

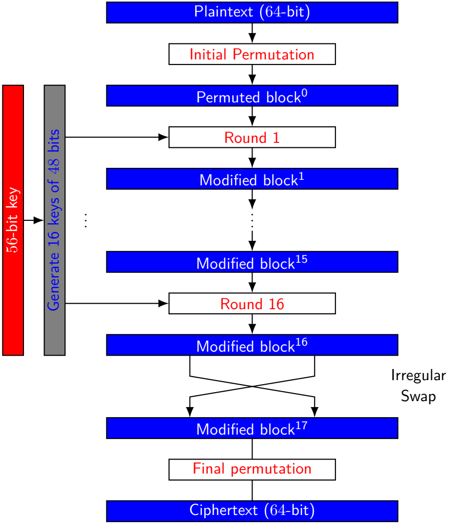
- System parameters
  - 64 bit input/output bits (block length)
  - 56 bits of key
- Principle: 16 round of Feistel system

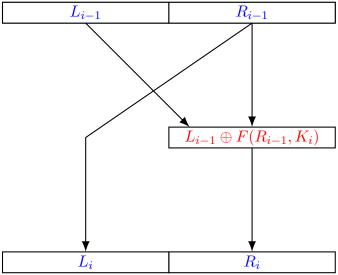
One round of a Feistel system



$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$
$$L_i = R_{i-1}$$

16 Rounds of DES





Decryption of DES

- DES decryption function is the same as the DES encryption function
  - except the round keys are applied in the reverse order.

$$
\begin{array}{ccc}
L_0 & R_0 & \\
L_1 = R_0 & R_1 = L_0 \oplus F(R_0, K_1) & \text{1st round} \\
L_2 = R_1 & R_2 = L_1 \oplus F(R_1, K_2) & \text{2nd round} \\
R_2 & L_2 & \text{Irregular swap}
\end{array}
$$

$$
\begin{array}{ccc}
R_2 & L_2 & \\
L_2 = R_1 & R_2 \oplus F(L_2, K_2) =? & \text{1st round} \\
L_1 = R_0 & R_1 \oplus F(L_1, K_1) =? & \text{2nd round} \\
L_0 & R_0 & \text{Irregular swap}
\end{array}
$$

DES Properties

- A DES weak key is a key K such that
  - $E_K(E_K(x)) = x$ for all x.
  - There are four DES weak keys.
  - For each of the four DES weak keys K, there exists 2^{32} fixed points of E_K (i.e. plaintexts x such that $E_K(x) = x$)
- A pair of DES semi-weak keys is a pair (K_1, K_2) with $E_{K_2}(E_{K_1}(x)) = x$.
  - six pairs of semi-weak keys
- Is DES a group?
  - Given any two keys K_1, K_2, does there exist a third key K_3 such that $E_{K_3}(x) = E_{K_2}(E_{K_1}(x))$?
  - Is multiple encryption equivalent to a single encryption?

Attacks to DES

- Exhaustive Search:
  - Known: X and Y (known plaintext attack)
  - Unknown: K such that $Y = DES_K(X)$
  - Idea: test all possible keys.
  - Key size (56 bits) is too small
- Differential Cryptanalysis:
  - Proposed by Biham & Shamir in 1990.
  - Principle:
    - Analyze the differences in ciphertexts for suitably chosen plaintext pairs and deduce the likelihood of certain keys.

Differential Cryptanalysis

- Requirements for 16-round DES
  - With chosen plaintext $2^{47}(X, Y)$ pairs are needed.
  - With known plaintext $2^{55}(X, Y)$ pairs are needed.
  - 2^{37} arithmetic operations are needed.
  - High storage requirement for the pairs makes the attack highly impractical.
- Remark: DES s-boxes are optimized for differential cryptanalysis (i.e. the designers were aware of this attack)

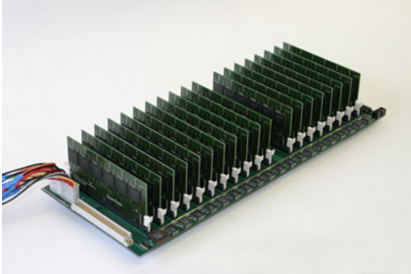
- Proposed by Matsui in 1993 & presented at CRYPTO'94
  - 2^{43} known plaintexts with complexity 2^{43} with success rate 85%.
- The actual attack is implemented
  - Using 12 HP RISC workstations running at 99 MHz
  - With 2^{47} known plaintexts, the key was discovered in 50 days.
- Remark: DES s-boxes are not optimized against this attack.

History of Attacks Against DES

Date	Proposed/implemented attack
1977	Diffie&Hellman, estimates the cost of key search engine ($20m)
1990	Biham&Shamir proposes differential cryptanalysis (2^{47} chosen ciphertext)
1993	Michael Wiener proposes a detailed hw design for key search engine; average search time: 3.5 hours @ less than $1m
1993	Matsui proposes linear cryptanalysis (243 known ciphertext)
Jun. 1997	DES Challenge I broken, distributed effort took 96 days
Feb. 1998	DES Challenge II-1 broken, distributed effort (distributed.net) took 41 days
July 1998	DES Challenge II-2 broken, key search machine deepcrack built by Electronic Frontier Foundation (EFF), 1800 ASICs, each with 24 search units (deepcrack) , $250K, 15 days average, (actual time 56 hours)
Jan. 1999	DES Challenge III broken, distributed.net + EFF's deepcrack, it took 22 hours and 15 minute

Final Results

- COPACOBANA
  - Cost-Optimized PArallel COde Breaker
  - 120 FPGA @ 100 MHz
  - Each FPGA can check four keys every 10 ns.
  - 120 FPGA can check 48 billion keys per second.
  - 8.7 days to break DES, on average
  - Material Cost : $\sim$ US$ 10K



"In 2008 their COPACOBANA RIVYERA reduced the time to break DES to less than one day, using 128 Spartan-3 5000's" http://www.sciengines.com/copacobana/

DES Alternatives

- Double DES:
  - $C = E_{K_2}(E_{K_1}(P))$ and $P = D_{K_1}(D_{K_2}(C))$, where $K_1 \neq K_2$.
- Double DES is vulnerable to meet-in-the-middle attack by Merkle and Hellman.
- Meet-in-the-middle attack
  - Assume we have P and C where $C = E_{K_2}(E_{K_1}(P))$
  - $d = E_{K_1}(P)$
  - $D_{K_2}(C) = D_{K_2}(E_{K_2}(E_{K_1}(P))) = E_{K_1}(P) = d$

Double DES

- Meet-in-the middle attack
  - Eve intercepts P and $C = E_{K_2}(E_{K_1}(P))$.
  - She computes $E_K(P)$ for all possible K and stores them.
  - She computes $D_K(C)$ for all possible K and stores them.
  - Finally, she compares the two lists.
  - If there are N keys the storage requirement is $2N$.
  - N encryption and N decryption operations and comparisons.
  - Effective key length of Double DES is 57 bits.
  - Storage requirement:
    - $N = 2^{56}, 2N = 2^{57} \rightarrow 2N \times 8 = 2^{57} * 2^3 = 2^{60}$ B

Other Alternatives

- Triple DES:
  - $C = E_{K_3}(E_{K_2}(E_{K_1}(P)))$ provides ?-bit security.
  - $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$ provides ?-bit security.
- DESX:
  - $C = K_3 \oplus E_{K_2}(K_1 \oplus P)$
  - Fairly secure
- Rijndael was elected as the Advanced Encryption Standard (AES) out of 15 candidate algorithms in 2000.

AES Selection Process

- Successor to DES
- The selection process is administered by NIST
  - AES selection was an open process.
  - 1997, NIST called for candidates to replace DES.
  - Requirements were
    - Block cipher with 128-bit block size
    - Support for 128, 192, 256 bits of key sizes
    - Efficient software and hardware implementation.
  - Cryptographic community was asked to comment on five finalists: MARS(IBM), RC6(RSA), Rijndael, Serpent, Twofish.
  - NIST chose Rijndael as AES in 2000.

Joan **Dae**men &

Vincent **Rij**men

- Likely to be the most commonly used algorithm in the next decade.
- See `http://www.nist.gov/aes` for more information

Algorithm	Pentium Pro 200 Mhz Mbit/s	FPGA hardware Gbit/s
MARS	69	-
RC6	105	2,4
Rijndael	71	1,9
Serpent	27	4,9
Twofish	95	1,6

Performance : AES vs DES

- Hardware ASIC (0.12 μm)

Cipher	Area(# of gates)	Time Performance
TDES	5.5K/16.954K	334 Mbps/1.067 Gbps
AES	5.4K/20.328K/36.9K	311 Mbps/2.8 Gbps/4.459 Gbps

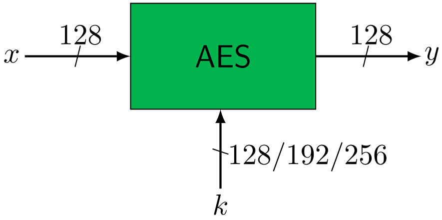
- Hardware FPGA (Virtex-E xcv1000E-8)

Cipher	Area(# of slices)	Time Performance
TDES	668/1122	136 Mbps/290 Mbps
AES	956/2529	109 Mbps/833 Mbps

- Software (AMD Opteron 8354 2.2 GHz processor under Linux)

Cipher	Mode	Time Performance
TDES	CTR	13 MiB/s
AES	CTR(128/192/256)	139/113/96 MiB/s

AES Overview



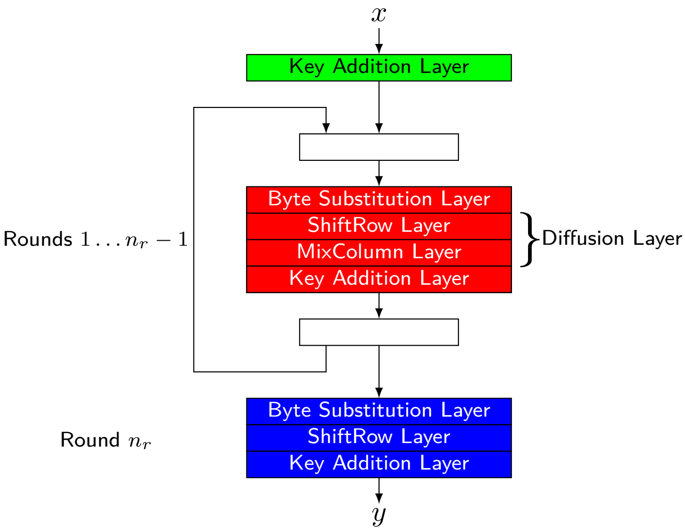
- Rijndael block size is also variable $(128/192/256)$
- Number of rounds (n_r) is a function of the key length:

Key length(in bits)	n_r
128	10
192	12
256	14

Rijndael Overview

- Not a Feistel cipher.
  - Recall: Feistel ciphers do not process the whole block in each iteration.
  - This explains why Rijndael has fewer number of rounds.
- Rijndael has three basic steps (or layers):
  - Key Addition Layer: XORing the block with the round key.
  - Byte Substitution Layer: 8-by-8 substitution (s-box). Nonlinear operation (confusion).
  - Diffusion Layer: provides the diffusion of the bits of a block. Linear operations
    - ShiftRow Layer
    - MixColumn Layer

Rijndael Encryption



The Layers

- We will assume the block and key lengths are fixed to 128-bit (16 bytes).
  - 16 bytes (128 bit) are arranged into a 4×4 matrix

$$S = \begin{pmatrix} s_0^{i-1} & s_4^{i-1} & s_8^{i-1} & s_{12}^{i-1} \\ s_1^{i-1} & s_5^{i-1} & s_9^{i-1} & s_{13}^{i-1} \\ s_2^{i-1} & s_6^{i-1} & s_{10}^{i-1} & s_{14}^{i-1} \\ s_3^{i-1} & s_7^{i-1} & s_{11}^{i-1} & s_{15}^{i-1} \end{pmatrix}$$

- Each matrix entry can be thought an element of $GF(2^8)$ with $x^8 + x^4 + x^3 + x + 1$.
  - We will occasionally do arithmetic in $GF(2^8)$.

The Byte Substitution Layer 1/2

- Each byte in the matrix is changed to another byte by the following operations:
  1. Each byte in S is an element of $GF(2^8)$, $A(x)$.
  2. Find the multiplicative inverse of $A(x)$, $T(x) = A^{-1}(x)$.
  3. Apply the affine transformation defined by

$$\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

The Byte Substitution Layer 2/2

- The result is another 4×4 matrix whose entries are bytes.

$$\begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{pmatrix} \leftarrow \begin{pmatrix} s_0^{i-1} & s_4^{i-1} & s_8^{i-1} & s_{12}^{i-1} \\ s_1^{i-1} & s_5^{i-1} & s_9^{i-1} & s_{13}^{i-1} \\ s_2^{i-1} & s_6^{i-1} & s_{10}^{i-1} & s_{14}^{i-1} \\ s_3^{i-1} & s_7^{i-1} & s_{11}^{i-1} & s_{15}^{i-1} \end{pmatrix}$$

- You can use a table with 256 entries whose entries are bytes in order to implement this layer.

The Shift Row Layer

- Four rows of the matrix are shifted cyclically to the left by offsets of 0, 1, 2, 3.

$$\begin{pmatrix} c_0 & c_4 & c_8 & c_{12} \\ c_1 & c_5 & c_9 & c_{13} \\ c_2 & c_6 & c_{10} & c_{14} \\ c_3 & c_7 & c_{11} & c_{15} \end{pmatrix} = \begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_5 & b_9 & b_{13} & b_1 \\ b_{10} & b_{14} & b_2 & b_6 \\ b_{15} & b_3 & b_7 & b_{11} \end{pmatrix} \leftarrow \begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{pmatrix}$$

The Mix Column Layer

$$\begin{pmatrix} d_0 & d_4 & d_8 & d_{12} \\ d_1 & d_5 & d_9 & d_{13} \\ d_2 & d_6 & d_{10} & d_{14} \\ d_3 & d_7 & d_{11} & d_{15} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} c_0 & c_4 & c_8 & c_{12} \\ c_1 & c_5 & c_9 & c_{13} \\ c_2 & c_6 & c_{10} & c_{14} \\ c_3 & c_7 & c_{11} & c_{15} \end{pmatrix}$$

- $02 = 0000\ 0010$
- $03 = 0000\ 0011$
- The Shift Row and the Mix Column Layers performs linear transformations,
  - i.e., $DIFF(A) \oplus DIFF(B) = DIFF(A \oplus B)$

The Round Key Addition

- A simple XORing operation

$$\begin{pmatrix} s_0^i & s_4^i & s_8^i & s_{12}^i \\ s_1^i & s_5^i & s_9^i & s_{13}^i \\ s_2^i & s_6^i & s_{10}^i & s_{14}^i \\ s_3^i & s_7^i & s_{11}^i & s_{15}^i \end{pmatrix} = \begin{pmatrix} d_0 & d_4 & d_8 & d_{12} \\ d_1 & d_5 & d_9 & d_{13} \\ d_2 & d_6 & d_{10} & d_{14} \\ d_3 & d_7 & d_{11} & d_{15} \end{pmatrix} \oplus \begin{pmatrix} k_0^i & k_4^i & k_8^i & k_{12}^i \\ k_1^i & k_5^i & k_9^i & k_{13}^i \\ k_2^i & k_6^i & k_{10}^i & k_{14}^i \\ k_3^i & k_7^i & k_{11}^i & k_{15}^i \end{pmatrix}$$

- The matrix whose entries are s_j^i is the output of the round i

The Key Schedule

- The original key consists of 128 bits (16 B)
- We need round keys for the 10 (12 or 14) rounds
- The nonlinear SBOX function is used to generate round keys

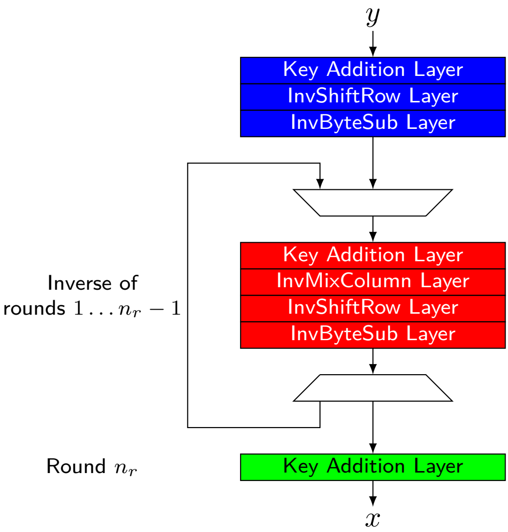
Decryption

- Rijndael is not Feistel cipher;
  - Thus each layer must actually be inverted.
  - Operations in each layer are invertible:
    - InvByteSub
    - InvShiftRow (Shift right instead of left)
    - InvMixColumn
    - The inverse of MixColumn exists because 4×4 matrix used in MixColumn is invertible.

  InvMixColumn matrix

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \qquad 0E = 0000\ 1110 \rightarrow$$

Rijndael Decryption



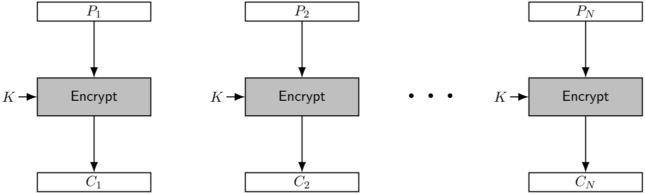
Final Remarks

- In every round, each bit in the block are treated uniformly
  - This has the effect of diffusing the input bits faster
  - After two rounds each of the 128 output bits depends on each of the 128 input bits.
- S-box is constructed using a very simple algebraic mapping,
  - $x \rightarrow x^{-1}$ in $GF(2^8)$ $\rightarrow$highly nonlinear; balanced
  - Its simplicity removes any suspicions about a certain *trapdoor*, which was believed to exist in DES for years.
- Key scheduling utilizes highly nonlinear SubByte mapping.
- No known attacks are better than brute force for seven or more rounds

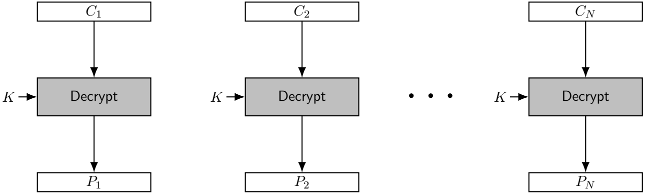
Modes of Operations

- Block ciphers encrypt fixed size blocks
  - DES and 3DES encrypt 64-bit blocks
  - AES encrypts 128-bit blocks
- In practise, we have arbitrary amount of information to encrypt
  - DES, 3DES, AES and other block ciphers are used in different modes in order to encrypt larger data blocks
- NIST SP 800-38A defines 5 modes can be used with any block cipher

Modes of Operations - Electronic Codebook (ECB)



Encryption



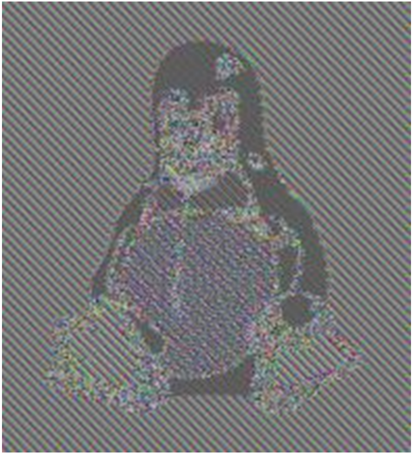
Decryption

Modes of Operations - ECB

- The plaintext P is broken into n-bit blocks, i.e. $P = P_1||P_2||P_3||\ldots||P_L$
- The ciphertext consists of the blocks $C = C_1||C_2||C_3||\ldots||C_L$ where $C_i = E_K(P_i)$ for $i = 1, 2, \ldots, L$.
- If P_L is not a n-bit block, padding is required.
- Each block is encrypted/decrypted independently of other blocks.
- Errors in a single block do not propagate to other blocks.
- Loss of a block does not affect decryption of other blocks.
- Identical plaintext blocks (under the same key) result in identical ciphertext blocks. (substitution cipher)

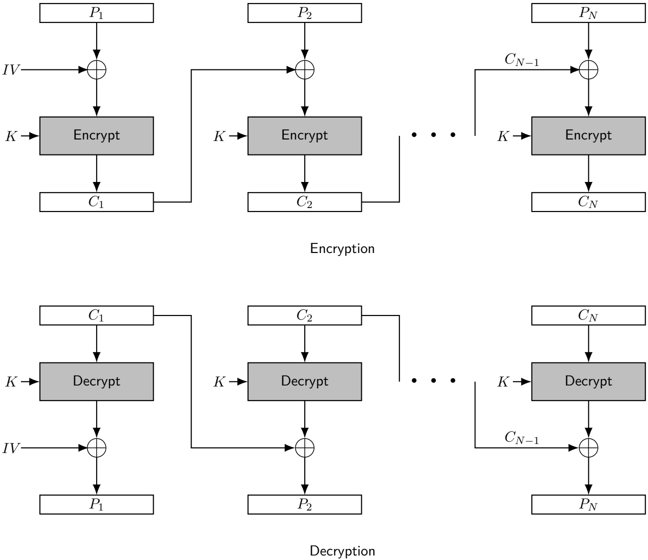
Image Encryption with ECB

Plaintext Image



Ciphertext with ECB

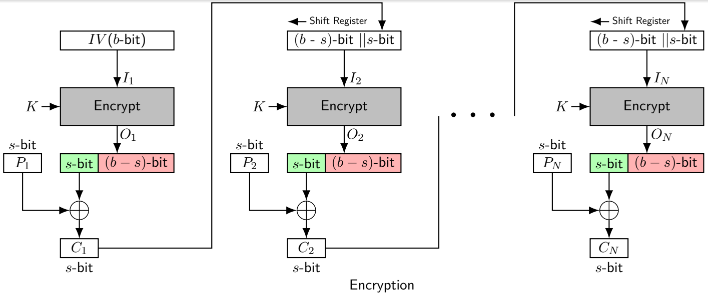
Modes of Operations - Cipher Block Chaining (CBC)

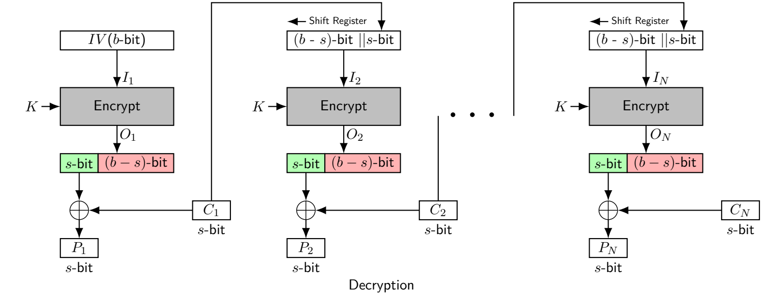


Modes of Operations - CBC

- $C_i = E_K(P_i \oplus C_{i-1})$
- $P_i =$?
- If P_L is not a n-bit block, padding is required.
- Encryption of a block depends on the encryption of previous blocks.
- Errors in a single block or malicious block substitutions propagates to the next block.
  - Self-synchronizing

Modes of Operations - Cipher Feedback (CFB)

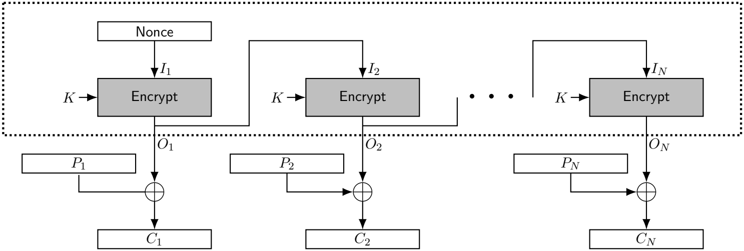




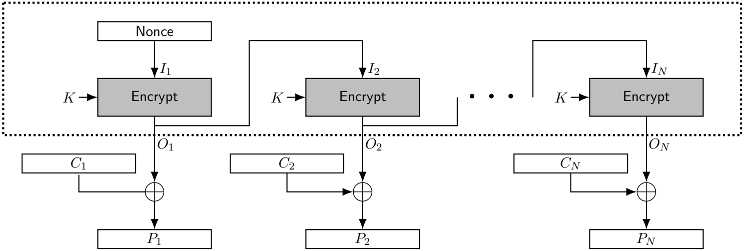
Modes of Operations - CFB

- Stream cipher mode
- A single s-bit unit can be encrypted without having to wait for entire block of data to be available.
  - s may be 1, 8 or more until block size.
  - 8 is generally preferred.
- Encryption of a s-bit unit depends on the encryption of previous s-bit units.
- Errors in a single s-bit unit or malicious s-bit unit substitutions propagates to the next s-bit units.
  - Self-synchronizing
- Implementation of decryption is not needed.

Modes of Operations - Output Feedback (OFB)



Encryption



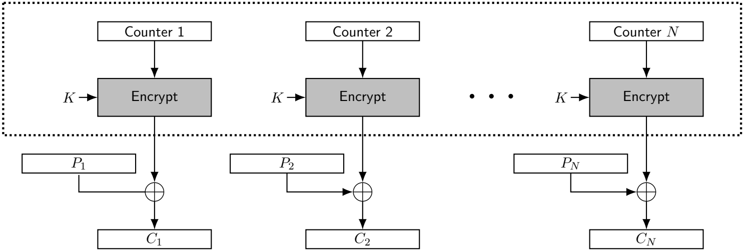
Decryption

Modes of Operations - OFB

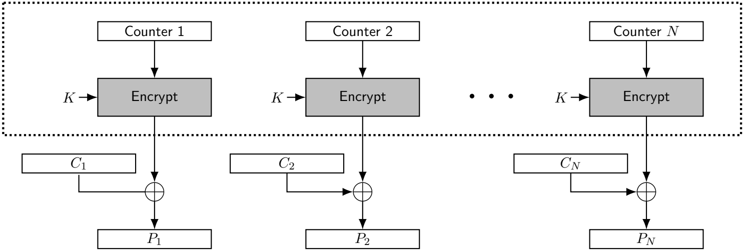
- Stream cipher mode
- Key-stream generation cannot be parallelized.
  - Can be computed before encryption/decryption.
- Implementation of decryption is not needed.
- Loss of a block affects all coming blocks.
- Errors in a single block do not propagate to other blocks.

- Same IV should not be used twice for the same key (general problem of using IV)
  - Two ciphertext blocks are XORed and the random sequence is cancelled
  - The attacker obtains XOR of two plaintexts
  - That is why IV is sometimes called as nonce (means "used only once")

Modes of Operations - Counter (CNT)



Encryption



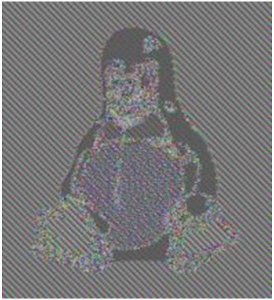
Decryption

Modes of Operations - CNT

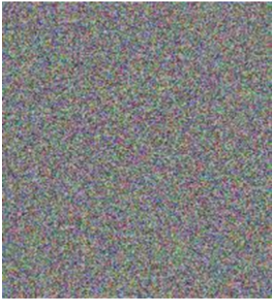
- Stream cipher mode
- Ciphertext blocks can be decrypted independently.
  - Can be computed before encryption/decryption.
  - Can be parallelized for speed
  - we can perform selective decryption
- Implementation of decryption is not needed.
- Loss of a block affects all coming blocks.
- Errors in a single block do not propagate to other blocks.
- For the same key, the counter value should not repeat
  - Same problem as in OFB

Image Encryption with Different Modes

Plaintext Image



Ciphertext with ECB



Ciphertext with other modes