Block Ciphers: DES, AES

Guevara Noubir
http://www.ccs.neu.edu/home/noubir/Courses/CSG252/F04

Textbook: "Cryptography: Theory and Applications",

Douglas Stinson, Chapman & Hall/CRC Press, 2002

Reading: Chapter 3

Outline

- Substitution-Permutation Networks
- Linear Cryptanalysis
- Differential Cryptanalysis
- DES
- AES
- Modes of Operation

Block Ciphers

- Typical design approach:
  - Product cipher: substitutions and permutations
    - Leading to a non-idempotent cipher
  - Iteration:
    - Nr: number of rounds
    - Key schedule: $k \rightarrow k^1, k^2, \ldots, k^{Nr}$,
    - Subkeys derived according to publicly known algorithm
    - w^i: state
  - Round function
    - $w^r = g(w^{r-1}, k^r)$
    - w^0: plaintext x
  - Required property of g: ?
- Encryption and Decryption sequence

SPN: Substitution Permutation Networks

- SPN: special type of iterated cipher (w/ small change)
  - Block length: $l \times m$
    - $x = x_{(1)} \,||\, x_{(2)} \,||\, \dots \,||\, x_{(m)}$
    - $x_{(i)} = (x_{(i-1)l+1}, \dots, x_{il})$
  - Components:
    - Substitution cipher $\pi_S: \{0, 1\}^l \rightarrow \{0, 1\}^l$
    - Permutation cipher (S-box) $\pi_P: \{1, \dots, lm\} \rightarrow \{1, \dots, lm\}$
  - Outline:
    - Iterate Nr times: m substitutions; 1 permutation; $\oplus$ sub-key;
- Definition of SPN cryptosytems:
  - $P = ?$; $C = ?$; $K \subseteq ?$;
  - Algorithm:
  - Designed to allow decryption using the same algorithm
    - What are the parameters of the decryption algorithm?

SPN: Example

- $l = m = 4$; $Nr = 4$;
- Key schedule:
  - k: $(k_1, \dots, k_{32})$ 32 bits
  - k^r: $(k_{4r-3}, \dots, k_{4r+12})$

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

- K = 0011 1010 1001 0100 1101 0110 0011 1111
- x = 0010 0110 1011 0111
- y = 1011 1100 1101 0110

Linear Cryptanalysis

- Assumption:
  - Assume there exists a *probabilistic* linear relationship between a subset of plaintext bits and a subset of state bits immediately preceding the last substitution
  - Attacker has a large amount of plaintext-ciphertext encrypted using the same key
- Principle:
  - Consider a set of potential sub-keys, whenever a sub-key verifies the linear relation, increment its counter
  - The sub-key best matching probability could contain the correct values of key bits

Piling-up Lemma

- Given:
  - $X_1, X_2, \ldots$ independent random variables
  - $Pr[X_i=0] = p_i$; $Pr[X_i=1] = 1-p_i$; $0 \le p_i \le 1$
  - Let $\varepsilon_i = p_i - ½$ denote the bias of the distribution
- Piling-up Lemma:
  - Let $\varepsilon_{i1,i2,\ldots,ik}$ denote the bias of the random variable $X_{i1} \oplus X_{i2} \oplus \ldots \oplus X_{ik}$. Then: $$\varepsilon_{i1,i2,\ldots,ik} = 2^{k-1}\prod_{j=1}^{k}\varepsilon_{ij}$$
- Corollary:
  - If $\varepsilon_{ij} = 0$ for one variable $\Rightarrow$ $\varepsilon_{i1,i2,\ldots,ik} = 0$

Linear Approximations of S-boxes

- S-box:
  - $\Pi_S: \{0, 1\}^m \rightarrow \{0, 1\}^n$
  - Input:
    - $X = (x_1, \ldots, x_m)$
    - Each coordinate defines a random variable X_i with bias 0
    - Variables X_is are independent
  - Output:
    - $Y = (y_1, \ldots, y_n)$
    - Variables Y_js are not independent from each other and X_is
    - If $(y_1, \ldots, y_n) \neq \Pi_S(x_1, \ldots, x_m) \Rightarrow Pr[X_1= x_1, \ldots, X_m= x_m, Y_1= y_1, \ldots, Y_n= y_n] = ?$
    - If $(y_1, \ldots, y_n) = \Pi_S(x_1, \ldots, x_m) \Rightarrow Pr[X_1= x_1, \ldots, X_m= x_m, Y_1= y_1, \ldots, Y_n= y_n] = ?$
  - Therefore: one can compute the bias of: $X_{i1} \oplus \ldots \oplus X_{ik} \oplus Y_{j1} \oplus \ldots \oplus Y_{jl}$

Example

- $Pr[X_1 \oplus X_4 \oplus Y_2=0]$= ?
- $Pr[X_3 \oplus X_4 \oplus Y_1 \oplus Y_4=0]$= ?

- In general one computes the biases for all possible subsets:

$$(\bigoplus_{i=1}^{4} a_i X_i) \oplus (\bigoplus_{i=1}^{4} b_i Y_i)$$

- Derive a table (a, b) $\Rightarrow$ Number of times we get a 0: $N_L(a, b)$
  Entry for a=3, b=9: 2
  Bias $\varepsilon(a, b) = (N_L(a, b)-8)/16$

  $N_L(3, 9) = 2$
  $N_L(2, E) = 2$

Π_S Table

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1

Linear Attack on SPN

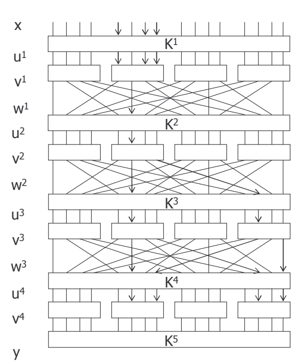
- Approach:
  - Find a set of linear approximations of S-boxes that can be used to derive a linear approximation of the SPN (excluding the last round)
  - Derive the bias value using pilling-up lemma
  - This linear approximation depends on some subset of the key bits
  - For each possible pair (plaintext, ciphertext), and each key increment the counter of the subkey if the approximation equation gives a 0
  - Hopefully the correct value for sub-keys will have the expected bias

Example

- Approximation equations:
  - T_1 has bias ¼
  - T_2 has bias -¼
  - T_3 has bias -¼
  - T_4 has bias -1/4

$$T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1$$
$$T_2 = U_6^2 \oplus V_6^2 \oplus V_8^2$$
$$T_3 = U_6^3 \oplus V_6^3 \oplus V_8^3$$
$$T_4 = U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3$$

- If T_1, T_2, T_3, T_4, were **independent** then the bias of
  - $T_1 \oplus T_2 \oplus T_3 \oplus T_4$ would be -1/32
  - We will make this non-rigorous approximations, because it seems to work in practice
- $T_1 \oplus T_2 \oplus T_3 \oplus T_4 = X_5 \oplus X_7 \oplus X_8 \oplus V^3_6 \oplus V^3_8 \oplus V^3_{14} \oplus V^3_{16} \oplus K^1_5 \oplus K^1_7 \oplus K^1_8 \oplus K^2_6 \oplus K^3_6 \oplus K^3_{14} = X_5 \oplus X_7 \oplus X_8 \oplus U^4_6 \oplus U^4_{14} \oplus U^4_8 \oplus U^4_{16} \oplus K^1_5 \oplus K^1_7 \oplus K^1_8 \oplus K^2_6 \oplus K^3_6 \oplus K^3_{14} \oplus K^4_6 \oplus K^4_8 \oplus K^4_{14} \oplus K^4_{16}$



Example (Cont.)

- If the key bits in:
  - $K^1_5 \oplus K^1_7 \oplus K^1_8 \oplus K^2_6 \oplus K^3_6 \oplus K^3_{14} \oplus K^4_6 \oplus K^4_8 \oplus K^4_{14} \oplus K^4_{16}$ **are fixed**
- Then the random variable:
  - $X_5 \oplus X_7 \oplus X_8 \oplus U^4_6 \oplus U^4_{14} \oplus U^4_8 \oplus U^4_{16}$ has bias $\pm 1/32$
- This allows us to derive 8 bits for last subkey:
  - $K^5_{(2)}$ and $K^5_{(2)}$
- Outline of alg:
  - Build a table for all possible 256 values of $K^5_{(2)}$ and $K^5_{(4)}$
  - For each (x, y) pair of plaintext and ciphertext; for each candidate subkey:
    - Obtain the values of $u4_{(2)}$ and $u4_{(4)}$ by decrypting y
    - Compute $x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_{14} \oplus u^4_8 \oplus u^4_{16}$
    - If equal 0 : increment counter of corresponding $K^5_{(2)}$ and $K^5_{(2)}$
  - The table entry that has bias close to 1/32 is the right value for $K^5_{(2)}$ and $K^5_{(2)}$

Requirements for Linear Cryptanalysis

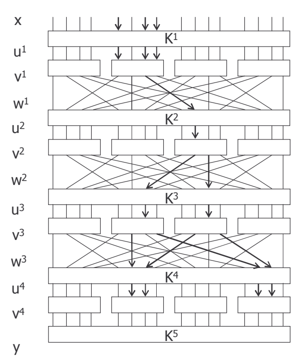
- A bias of ε requires:
  - $T = c\varepsilon^{-2}$ pairs of plaintext-ciphertext
- For the previous example T = 8000 was usually successful $\Rightarrow c \approx 8$

Differential Cryptanalysis

- Similar to Linear Cryptanalysis:
  - Compares the x-or of two inputs to the x-or of the corresponding two outputs: $x' = x \oplus x^*$ and $y' = y \oplus y^*$
  - Adversary has a large number of chosen plaintext tuples (x, x*, y, y*) s.t. x′ is fixed
  - Approach:
    - For each candidate key: decrypt y, and y*
    - For each candidate key: compute the values of certain state bits
    - If the state bits match the most likely value for the input x-or then increment the candidate key counter
- Proposed Encryption Standard (PES) which is the original proposal for the International Data Encryption Standard (IDEA used in PGP) was modified to resist to this kind of attacks
- GSM A3 algorithm is sensitive to this kind of attacks
  - SIM card secret key can be recovred => GSM cloning

Differential Cryptanalysis

- $\Delta(x') = \{(x, x \oplus x') : x \in \{0, 1\}^m\}$
- $N_D(x', y') = |\{(x, x^*) \in \Delta(x') : \Pi_S(x) \oplus \Pi_S(x^*) = y'\}|$
- Similarly to Linear cryptanalysis build the table: $N_D(a, b)$
- Define *propagation ratio*: $R_p(a', b') = \frac{N_D(a', b')}{2^m}$
- *Differential Trail*:
  - Is multi-round combination
  - Permutations and key XORs do not impact the differential
- Sub-key recovery and filtering
- Example:
  - $R_p(1011, 0010) = ½$; $R_p(0100, 0110) = 3/8$; $R_p(0010, 0101) = 3/8$; $R_p(0010, 0101) = 3/8$
  - Propagation ratio of differential trail: 27/1024
  - Complete attack requires between 50 and 100 tuples to find sub-key

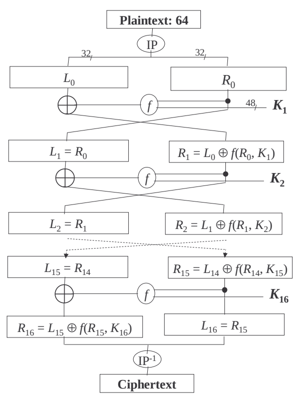


Today's Block Encryption Algorithms

- Key size:
  - Two short => easy to guess
- Block size:
  - Two short easy to build a table by the attacker: (plaintext, ciphertext)
  - Reasonable size: 64 bits
- Properties:
  - One-to-one mapping
  - Mapping should look random to someone who doesn't have the key
  - Efficient to compute/reverse
- How?
  - Substitution (small chunks) & permutation (long chunks)
  - Multiple rounds

Data Encryption Standard (DES)

- Developed by IBM for the US government
- Based on Lucifer (64-bits, 128-bits key in 1971)
- To respond to the National Bureau of Standards CFP (now called NIST)
  - Modified characteristics (with help from NSA):
    - 64-bits block size, 56 bits key length
  - Concerns about trapdoors, key size, sbox structure
- Adopted in 1977 as the DES (FIPS PUB 46, ANSI X3.92) and reaffirmed in 1994 for 5 more years
- Today replaced by AES



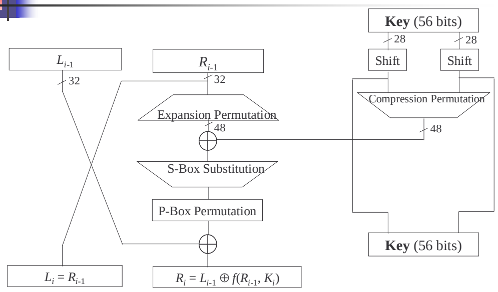
Feistel Cipher:

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

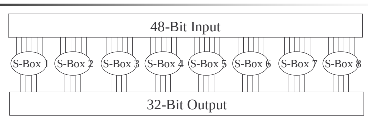
Feistel Cipher

- Function f does not have to be injective!
  - $L_i = R_i$-1
  - $R_i = L_i$-1 $\oplus$ $f(R_i$-1, $K_i)$
- How can we invert one round?

One DES Round



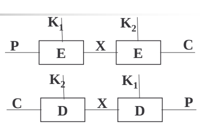
S-Box Substitution



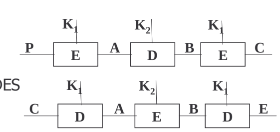
- S-Box heart of DES security
- S-Box: 4x16 entry table
  - Input 6 bits: $b_1b_2b_3b_4b_5b_6$
    - 2 bits (b_1b_6): determine the table row (1-4)
    - 4 bits ($b_2b_3b_4b_5$): determine the table entry
  - Output: 4 bits
  - Example: $S_1(101000) = 1101$
- S-Boxes are optimized against Differential cryptanalysis

Double/Triple DES

- Double DES
  - Vulnerable to Meet-in-the-Middle Attack [DH77]



- Triple DES
  - Used two keys K_1 and K_2
  - Compatible with simple DES (K1=K2)
  - Used in ISO 8732, PEM



DES Linear/Differential Cryptanalysis

- Differential cryptanalysis
  - "Rediscovered" by E. Biham & A. Shamir in 1990
  - Based on a chosen-plaintext attack:
    - Analyse the difference between the ciphertexts of two plaintexts which have a known fixed difference
    - The analysis provides information on the key
  - 8-round DES broken with 2^{14} chosen plaintext and complexity 2^{9}
  - 16-round DES requires 2^{47} chosen plaintext and complexity 2^{37}
- DES design took into account this kind of attacks
- Linear cryptanalysis
  - Uses linear approximations of the DES cipher (M. Matsui 1993)
  - Applied to DES:
    - Requires 2^{43} known plaintext encrypted with the same key
    - Time: 40 days to generate the pairs, 10 days to find the key

Breaking DES

- Electronic Frontier Foundation built a "DES Cracking Machine" [1998]
  - Attack: brute force
  - Inputs: two ciphertext
  - Architecture:
    - PC
    - Array of custom chips that can compute DES
      24search units/chip x 64chips/board x 27 boards
  - Power:
    - Searches 92 billion keys per second
    - Takes 4.5 days for half the key space
    - Successfully broke "DES Challenge II-2" in 56 hours
  - Cost:
    - $130'000 (all the material: chips, boards, cooling, PC etc.)
    - $80'000 (development from scratch)

The Advanced Encryption Standard (AES)

http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

- AES = Rijndael
- Designed by Rijmen-Daemen (Belgium)
- Key size: 128/192/256 bit
- Block size: 128 bits of data
- Properties: **iterative** rather than **Feistel** cipher
  - Treats data in 4 groups of 4 bytes
  - Operates on an entire block in every round
- Designed to be:
  - Resistant against known attacks
  - Speed and code compactness on many CPUs
  - Design simplicity

AES Outline

- Algorithm:
  1. Initialize State $\leftarrow x \oplus$ RoundKey;
  2. For each of the Nr-1 rounds:
     1. SubBytes(State);
     2. ShiftRows(State);
     3. MixColumns(State);
     4. AddRoundKey(State);
  3. Last round:
     1. SubBytes(State);
     2. ShiftRows(State);
     3. AddRoundKey(State);
  4. Output $y \leftarrow$ State

Finite Fields - Break

- $GF(2) = Z_2$
- $GF(2^n) = \{ (a_{n-1}, \ldots, a_1, a_0) \mid a_i \in GF(2) \}$
  - Addition can be carried out bit by bit
  - Multiplication mod a *primitive polynomial*
  - Generation of $GF(2^n)$: done by polynomial modulo a primitive polynomial of degree n, m(x)
  - Elements of GF(2) can be represented as a polynomial $(a_{n-1}, \ldots, a_1, a_0) = a(x)$
    $\equiv a_{n-1}x^{n-1} + \ldots + a_1x + a_0 \bmod m(x)$
  - Textbook Notation: $F_2^n = Z_2[x]/m(x)$
  - AES uses n=8, and $m(x) = x^8+x^4+x^3+x+1$

Primitive Polynomial

- A polynomial f(x) over a field Q is said to be *irreducible* if f(x) cannot be factored over Q
- A polynomial f(x) over a field Q is said to be *primitive* if every root of f(x) generates the extension field
  - Over GF(2) a polynomial is primitive if the smallest k for which f(x) divides x^k+1 is $k = 2^n-1$
- Example.
  - $f(x) = x^4+x^3+x^2+x+1$ over GF(2)
    f(x) is irreducible but not primitive
  - $g(x) = x^4+ x+1$
    g(x) is primitive

Test Irreducibility

- To show $x^8+x^4+x^3+x+1$ is irreducible:
  - If the number of terms is odd over GF(2), then it cannot be divisible by x+1
  - Try dividing by polynomials of degree 2, $x^2 + x + 1$
  - Try polynomials of degree 3, x^3+x+1 and $x^3+x^2 +1$
  - Try polynomials of degree 4, $x^4+x^3 +x^2+x+1$, $x^4+x^3 +1$, x^4+x^2+1, x^4+x+1
  - Do not require any more testing beyond degree 4

Test Primitivity

- To show $x^4+x^3+x^2+x+1$ is not primitive:
  - Take α to be a root of the polynomial, that is, $\alpha^4 = \alpha^3 + \alpha^2 + \alpha + 1$
  - $\alpha^5 = \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha^3 + \alpha^2 + \alpha + 1 + \alpha^3 + \alpha^2 + \alpha$
    $= 1$
- To show $x^8+x^4+x^3+x+1$ is primitive
  - Take α to be a root of the polynomial, that is, $\alpha^8 = \alpha^4 + \alpha^3 + \alpha + 1$ (00011011)=(1b)
  - $\alpha^9 = \alpha^5 + \alpha^4 + \alpha^2 + \alpha$ (00110110)=(36)
  - $\alpha^{12} = \alpha^7 + \alpha^5 + \alpha^3 + \alpha + 1$ (10101011)=(ab)

Multiplication in GF(2^8)

- $\alpha^9 * \alpha^{12} = \alpha^{9+12} = \alpha^{21 \bmod 127}$
- (36) $*$ (ab) = (00110110) $*$ (10101011) = 11110010

AES (Continued)

- State: 16 bytes structured in a array

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

- Each byte is seen as an element of $\mathbf{F}_{2^8}$=GF(2^8)
  - $\mathbf{F}_{2^8}$ finite field of 256 elements
    - Operations
      - Elements of $\mathbf{F}_{2^8}$ are viewed as polynomials of degree 7 with coefficients {0, 1}
      - Addition: polynomials addition $\Rightarrow$ XOR
      - Multiplication: polynomials multiplication modulo $x^8+x^4+x^3+x+1$

SubBytes

Input: a = ($a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$)
Output: b = ($b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$)

If a $\neq$ 0 then a $\leftarrow$ a^{-1};

$$\begin{bmatrix} b_0\\ b_1\\ b_2\\ b_3\\ b_4\\ b_5\\ b_6\\ b_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} a_0\\ a_1\\ a_2\\ a_3\\ a_4\\ a_5\\ a_6\\ a_7 \end{bmatrix} + \begin{bmatrix} 1\\ 1\\ 0\\ 0\\ 0\\ 1\\ 1\\ 0 \end{bmatrix}$$

SubBytes is non-linear
Example: SubBytes(0x53) = 0xED; // note that $0x53^{-1}$ = 0xCA
SubBytes can also be defined as a 16x16 table

ShiftRows

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

$\leftarrow$

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

MixColumn

- Input:
  - c: column index;
  - s: state;
- Output:
  - s: new state
- For i =0 to 3
  - $t_i \leftarrow s_{i,c}$;
- $u_0 \leftarrow x \cdot t_0 \oplus (x+1) \cdot t_1 \oplus t_2 \oplus t_3$
- $u_1 \leftarrow x \cdot t_1 \oplus (x+1) \cdot t_2 \oplus t_3 \oplus t_0$
- $u_2 \leftarrow x \cdot t_2 \oplus (x+1) \cdot t_3 \oplus t_0 \oplus t_1$
- $u_3 \leftarrow x \cdot t_3 \oplus (x+1) \cdot t_0 \oplus t_1 \oplus t_2$
- For i =0 to 3
  - $s_{i,c} \leftarrow u_i$;
- Note:
  - Multiplications are in $\mathbf{F}_{2^8}$

Key Expansion

- 10-round case (128 bits key)
  - RCon[1-10] ← CONSTANTS
  - For i = 0 to 3
    do w[i] ← (key[4i], key[4i+1], key[4i+2], key[4i+3])
    // w has 4 blocks of 32 bits
  - For i = 4 to 43
    do temp ← w[i-1]
    if i ≡ 0 (mod 4) then temp ← SubWord(RotWord(temp)) ⊕ RCon[i/4]
    w[i] ← w[i-4] ⊕ temp
  - Return (w[0], …, w[43])

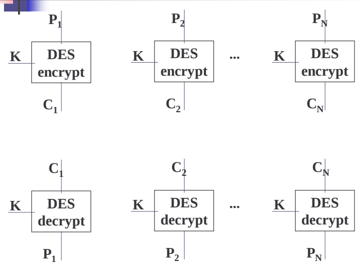
Implementation Aspects

- Can be efficiently implemented on a 8-bit CPU
  - Byte substitution works on bytes using a table of 256 entries
  - Shift rows is simple byte shifting
  - Add round key works on byte XORs
  - Mix columns requires matrix multiply in $\mathbf{F}_{2^8}$ which works on byte values, can be simplified to use a table lookup

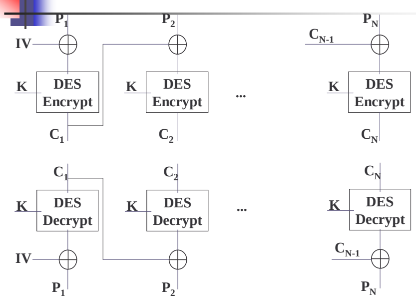
Implementation Aspects

- Can be efficiently implement on 32-bit CPU
  - Redefine steps to use 32-bit words
  - Can pre-compute 4 tables of 256-words
  - Then each column in each round can be computed using 4 table lookups + 4 XORs
  - At a cost of 16Kb to store tables
  - See Problem Set 4.
- Designers believe this very efficient implementation was a key factor in its selection as the AES cipher

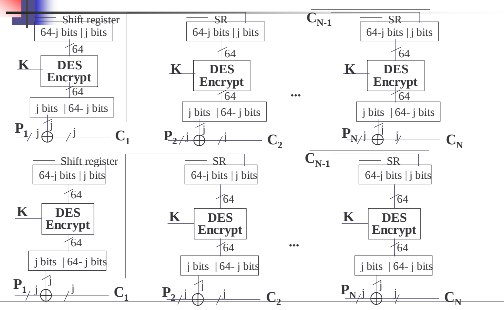
Modes of Operation DES/AES: Electronic Codebook (ECB)



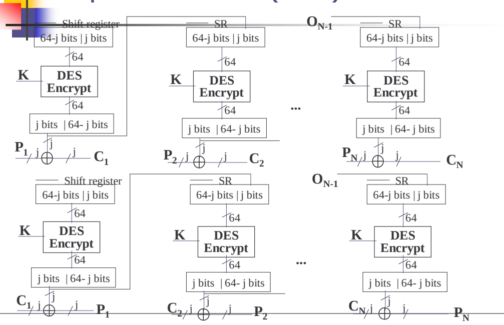
Cipher Block Chaining (CBC)



Cipher Feedback (CFB)



DES Modes: Output Feedback (OFB)



Counter (CTR)

- A "new" mode, though proposed early on
- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key & counter value for every plaintext block (never reused)

  $C_i = P_i \text{ XOR } O_i$

  $O_i = DES_{K1}(i)$
- Uses:
  - High-speed network encryptions