Notes on CO-487/CO-687: Applied Cryptography

Unversity of Waterloo

David Duan

Last Updated: January 2, 2022 (draft)

Contents

Chapter 1

Introduction

Chapter 2

Symmetric-Key Encryption

Section 1. Data Encryption Standard (DES)

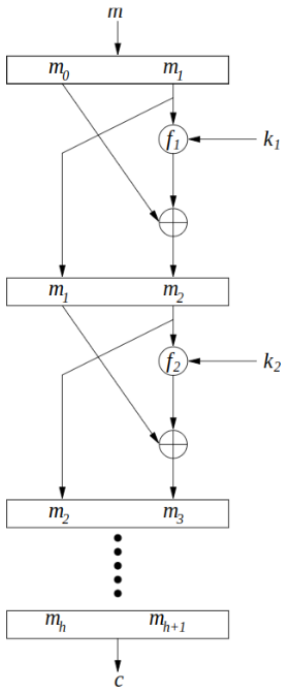
1.1. Note: A **block cipher** is a symmetric-key encryption scheme in which a fixed-length block of plaintext determines an equal-sized block of ciphertext. More formally, a block cipher consists of two paired algorithms, E for encryption and D for decryption, where $D = E^{-1}$. Both accept two inputs: a block of size n bits and a key of size k bits, and both yield an n-bit output block.

1.2. Note: Feistel ciphers is a class of block ciphers with the following components:

- Parameters: n (half the block length), h (number of rounds), and ℓ (key size).
- Notation: plaintext space $M = \{0,1\}^{2n}$, ciphertext space $C = \{0,1\}^{2n}$, key space $K = \{0,1\}^{\ell}$.
- A **key scheduling algorithm** which determines **subkeys** $k_1, k_2, \ldots, k_h$ from a key k.
- Each subkey k_i defines a **component function** $f_i : \{0,1\}^n \to \{0,1\}^n$.

Encryption takes h rounds:

- Input: Plaintext $m = (m_0, m_1)$ where $m_i \in \{0,1\}^n$.
- Round 1: $(m_0, m_1) \mapsto (m_1, m_2)$ where $m_2 = m_0 \oplus f_1(m_1)$.
- Round 2: $(m_1, m_2) \mapsto (m_2, m_3)$ where $m_3 = m_1 \oplus f_2(m_2)$.
- $\cdots$
- Round h: $(m_{h-1}, m_h) \mapsto (m_h, m_{h+1})$ where $m_{h+1} = m_{h-1} \oplus f_h(m_h)$.
- Output: Ciphertext $c = (m_h, m_{h+1})$.



Decryption applies these function in reverse order, i.e.,

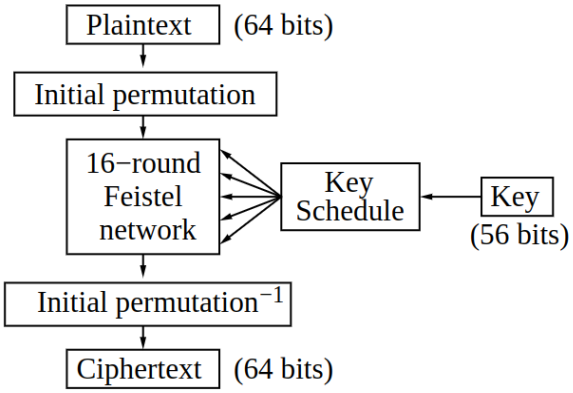
- Input: Ciphertext $c = (m_h, m_{h+1})$ and key k.
- Round 1: Compute $m_{h-1} = m_{h+1} \oplus f_h(m_h)$.
- Round 2: Compute $m_{h-2} = m_h \oplus f_{h-1}(m_{h-1})$.
- $\cdots$
- Round h: Compute $m_0 = m_2 \oplus f_1(m_1)$.
- Output: Plaintext $m = (m_0, m_1)$.

1.3. Remark: Note that we are taking the advantage that $x \oplus y \oplus y = x$ for any x, y. In other words, since XOR is commutative and self-inverse ($y \oplus y = 0$), we can decrypt the message by applying XOR with the same argument again. Thanks to this, there is no restrictions on the functions f_i's in order for the encryption procedure to be invertible.

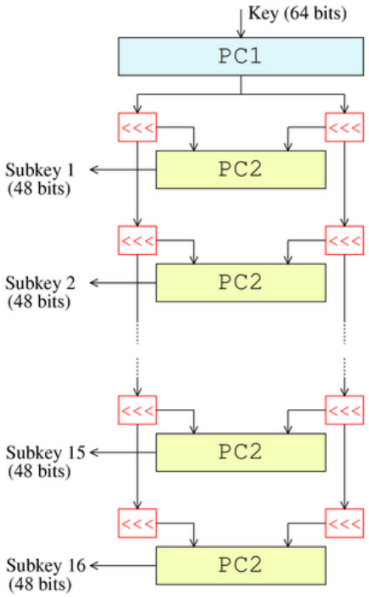
1.4. Motivation: The underlying principle of Feistel Cipher is that we want to take something "simple" and use it several times, hoping that the result is "complicated". This method is also easy to implement, as just need to implement one round of encryption; all subsequent rounds can use the same code. Also, decryption can use the same code, just with subkeys in reversed order.

1.5. Note (DES, 1970s): Data Encryption Standard is a block cipher with 64-bit blocks, 56-bit key, and 16 rounds of operation.

- DES uses a *Feistel network design.*
- Plaintext is divided into two halves.
- Key i used to generate subkeys $k_1, k_2, \ldots, k_h$.
- f_i is a *component function* whose output value depends on k_i and m_i.

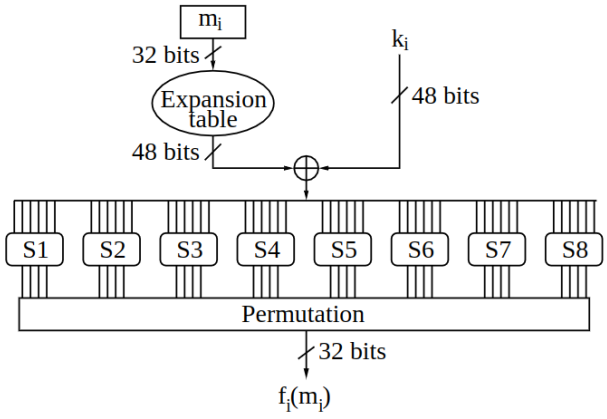


The initial permutation and its inverse can be found here: Wikipedia. Next, let us discuss the **key scheduling algorithm**, i.e., how to generate $k_1, \ldots, k_h$ from input key k. Given an input key of 64-bit, PC1 (permuted choice 1) throws away 8 bits and returns a 56-bit key. Now in each round, we break the key in two parts, left shift each half by a certain number of bits, and concatenate the results to produce subkeys.



Round number	Left shift each half by this many bits
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

It remains to look at the component functions $f_i : \{0,1\}^{32} \to \{0,1\}^{32}$ for $i \in \{1, \ldots, h\}$.



The information about S-boxes can be found here: Wikipedia. Let's dive in and see how it works. Suppose you are given an input of 101101_2. Take the middle 4 digits and convert it to decimal, you get 6. Now take the outer 2 digits and convert it to decimal, you get 3. Then your output is the $(3, 6)$ position of the table. For example, if this were for S_1, then your output is $1 = 0001_2$.

1.6. Note: Is DES secure? Recall that a symmetric key encryption scheme is said to be secure if it is *semantically secure* against a *chosen-plaintext attack* by a *computationally bounded adversary*. DES is not secure with modern computational power:

- Brute-force: 2^{56} decryptions because the key is 56 bits, so $O(2^{56})$ time and $O(1)$ space.
- Differential cryptanalysis: 2^{49} chosen plaintexts.
- Linear cryptanalysis: 2^{43} known plaintexts.

To mitigate the problems, we could encrypt multiple times, i.e., re-encrypt the ciphertext one or more time using independent keys, and hope that this operation increases the effective key length. However, this doe snot always increase security. For example, if E_π denotes the simple substitution cipher with key π, then is $E_{\pi_1} \circ E_{\pi_2}$ any more secure than E_π?

1.7. Note: In **Double-DES**, we have two keys $k = (k_1, k_2)$ where $k_1, k_2 \in_R \{0,1\}^{56}$. Encryption is $c = E_{k_2}(E_{k_1}(m))$ and decryption is $m = E_{k_1}^{-1}(E_{k_2}^{-1}(c))$. The key size is now $\ell = 112$, so exhaustive key search takes 2^{112} steps, which is infeasible. However, this is still not enough. We discuss the *meet-in-the-middle* attack.

- Suppose we have known plaintext pairs (m_i, c_i) for $i = 1, 2, 3, \ldots$
- For each $h_2 \in \{0,1\}^{56}$, compute $E_{h_2}^{-1}(c_1)$ and store $[E_{h_2}^{-1}(c_1), h_2]$ in a table.
- For each $h_1 \in \{0,1\}^{56}$, compute $E_{h_1}(m_1)$ and search for this in the table. If $E_{h_1}(m_1) = E_{h_2}^{-1}(c_1)$, then check if $E_{h_2}(m_2) = E_{h_2}^{-1}(c_2)$. If so, continue checking the rest. If all checks pass, then output (h_1, h_2) and stop.

The complexity for this attack is $\approx 2^{57}$ time and $O(2^{56})$ space.

1.8. Note: To avoid this attack, we could use **Triple-DES**, where the key is $k = (k_1, k_2, k_3)$, $k_i \in_R \{0,1\}^{56}$, and the ciphertext is $c = E_{k_3}(E_{k_2}(E_{k_1}(m)))$. The key length of 3-DES is $\ell = 168$, so exhaustive search takes 2^{168} steps (infeasible). Now that any meet-in-the-middle attack takes 2^{112}, which is still infeasible. Thus, the effective key length of 3-DES against exhaustive key search is ≤ 112 bits. However, the block length is now the weak link. Adversary could store a large table (of size $\leq 2^{64}$) of (m, c) pairs and do a dictionary attack. To prevent this attack, we should change secret keys frequently.

1.9. Note: Some variants:

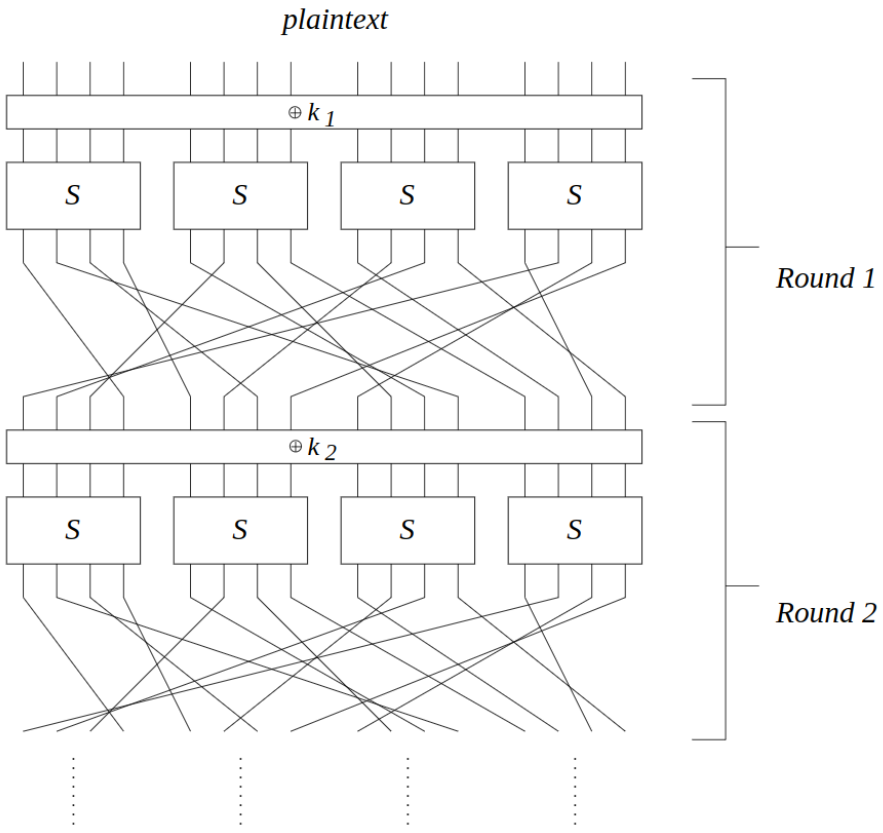
- Encrypt-Decrypt-Encrypt Triple-DES (for backward compatibility with DES)
- Two-Key Triple-DES (for storing some storage)

Section 2. Advanced Encryption Standard

2.1. Note: Requirements for AES:

- Key sizes: 128, 192, 256 bits.
- Block size: 128 bits.
- Efficient on both software and hardware platforms.
- Availability on a worldwide, non-exclusive, royalty-free basis.

2.2. Note: A **substitution-permutation network** (SPN) is a multiple-round iterated block cipher where each round consists of a **substitution** operation followed by a **permutation** operation.



- $\oplus$: Bitwise XOR.
- S: Substitution box (lookup table).
- Lines below S: Permutation.

Note that anything happens after the last time you use the key can be undone. Thus, it is common to introduce one more key-involved operation after the final round.

2.3. Note: AES is an SPN where the permutation operation consists of two linear transformations, one of which is a permutation.

- All operations are *byte-oriented* (process 8 bites at a time).
- The block size of AES is 128-bits and each round key is 128 bits.
- A key-schedule is used to generate the round keys.
- AES accepts three different key lengths. The number of rounds depends on the key length.
  - Key length = 128, $h = 10$.
  - Key length = 192, $h = 12$.
  - Key length = 256, $h = 14$.

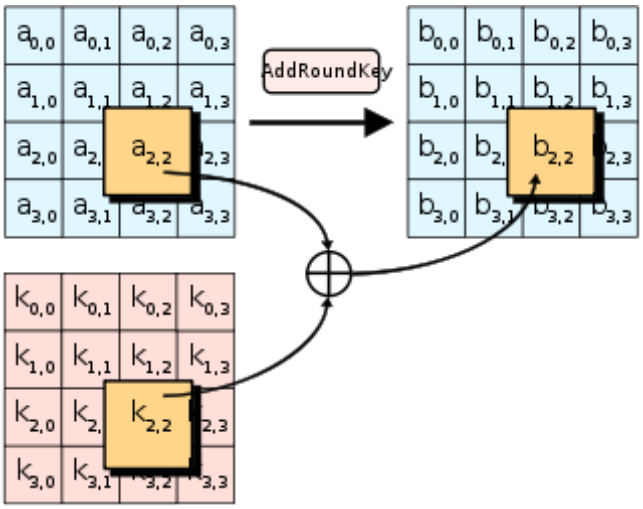
As with the previous ciphers we have studied,

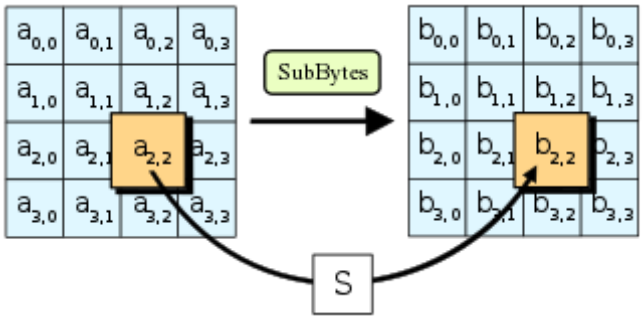
- the substitution operation (S-box) is the only non-linear component of the cipher;
- the permutation operations (permutation and linear transformation) spread out the non-linearities in each round.

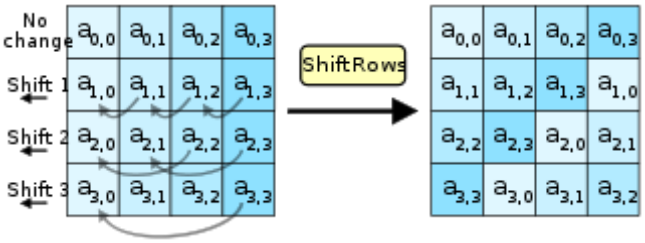
2.4. Note: Each round of AES updates a variable called *State* which consists of a 4×4 array of bytes ($4 \cdot 4 \cdot 8 = 128$, the block size). This state is initialized with the 128-bit plaintext. After h rounds are completed, one final additional round key is XOR-ed with State to produce the ciphertext (*key whitening*, see the previous page).

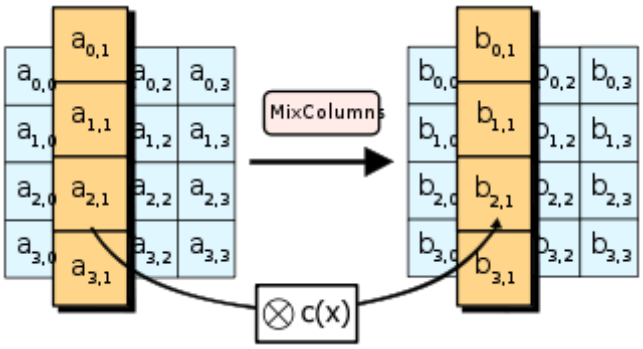
The AES round function uses four operations:

- **AddRoundKey** (key mixing): Bitwise XOR each byte of State with the corresponding byte of the round Key.
- **SubBytes** (S-box): Take each byte in State and replace it with the output of the S-box. Note that $S : \{0,1\}^8 \to \{0,1\}^i$ is a fixed and publish function.
- **ShiftRows** (permutation): Permute the bytes of State by applying a cyclic shift to each row. Shift the ith row by i steps, $i = 0, 1, 2, 3$.
- **MixColumns** (matrix multiplication / linear transformation): Most mathematically complicated step. Left multiply each column by a fixed matrix. See next page.


AddRoundKey


SubBytes
S


No change
Shift 1
Shift 2
Shift 3
ShiftRows


MixColumns
c(x)

We now look at the MixColumn operation. Viewing it as a bit operation:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2a_0 + 3a_1 + 1a_2 + 1a_3 \\ 1a_0 + 2a_1 + 3a_2 + 1a_3 \\ 1a_0 + 1a_1 + 2a_2 + 3a_3 \\ 3a_0 + 1a_1 + 1a_2 + 2a_3 \end{bmatrix}$$

- Each a_i and b_i is a byte. Regard bytes as 8-bit arrays.
- Each addition operation is a bitwise XOR.
- Multiplication by 1 is the identity.
- Multiplication by 2 is the following operation:
  - If the left-most-bit is 0, then perform a cyclic left shift.
  - If the left-most-bit is 1, discard the 1, insert a 0 on the right, and XOR with `0x1b = 00011011`.
- Multiplication by 3 is the operation $3x = 2x + 1x$.

Alternatively, we can regard this as a matrix transformation in the finite field $\mathrm{GF}(2^8)$:

- Regard all bytes as polynomials in the finite field $\mathrm{GF}(2^8) = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.
- Regard all integers in the matrix $1, 2, 3$ as bytes via their binary representations, e.g., $3 = 00000011_2 = x + 1$, and $2 = 00000010_2 = x$, and $1 = 00000001_2 = 1$.
- Perform all additions and multiplications in the finite field $\mathrm{GF}(2^8)$.

A third alternative is to regard it as a polynomial multiplication.

- The MixColumns matrix is a cyclic matrix, i.e., each row is a rotation of the previous row.
- Multiplication by a cyclic $N \times N$ matrix corresponds to polynomial multiplication modulo $X^N - 1$.
- Regard the column of a_i's as a polynomial in $\mathrm{GF}(2^8)[X] : a_0 + a_1X + a_2X^2 + a_3X^3$.
- Modulo $X^4 - 1$, we compute

  $$(02 + 01 \cdot X + 01 \cdot X^2 + 03 \cdot X^3) \cdot (a_0 + a_1X + a_2X^2 + a_3X^3)$$

  to obtain $b_0 + b_1X + b_2X^2 + b_3X^3$.

2.5. Note: We are ready to describe the AES Encryption scheme.

- From the key k, derive $h + 1$ round keys $k_0, k_1, \ldots, k_h$ via the key scheduler.

- The encryption function:

$$
\begin{array}{l}
\text{State} \leftarrow \text{plaintext} \\
\text{State} \leftarrow \text{State} \oplus k_0 \\
\text{for } i = 1 \ldots h-1 \text{ do} \\
\quad \text{State} \leftarrow \text{SubBytes (State)} \\
\quad \text{State} \leftarrow \text{ShiftRows (State)} \\
\quad \text{State} \leftarrow \text{MixColumns (State)} \\
\quad \text{State} \leftarrow \text{State} \oplus k_i \\
\text{State} \leftarrow \text{SubBytes(State)} \\
\text{State} \leftarrow \text{ShiftRows(State)} \\
\text{State} \leftarrow \text{State} \oplus k_h \\
\text{ciphertext} \leftarrow \text{State}
\end{array}
$$

  Note that MixColumns is not applied in the final round, because it can be easily undone.

2.6. Note (AES Key Schedule):

- For 128-bit keys, AES has ten rounds, so we need eleven subkeys.
- Each k_i is a 32-bit word (viewed as a 4-byte array).
- Each group of four k_i's forms a 128-bit subkey.
- The first round subkey (k_0, k_1, k_2, k_3) equals the actual AES key.

The functions $f_i : \{0,1\}^{32} \to \{0,1\}^{32}$ are defined as follows:

- Left-shift the input cyclically by 8-bits.
- Apply the AES S-box to each byte.
- Bitwise XOR the left-most byte with a constant which varies by rounding according to the following table.

Round	`constant`	`Round`	constant
1	`0x01`	6	`0x20`
2	`0x02`	7	`0x40`
3	`0x04`	8	`0x80`
4	`0x08`	9	`0x1B`
5	`0x10`	10	`0x36`

- Output the result.

Section 3. Linear Cryptanalysis

3.1. Note: Recall that a *substitution-permutation network* is a type of iterated block cipher where each round consists of a *substitution* operation followed by a *permutation* operation. Examples include the component function of DES and AES.

- The key k influences the result of the substitution step.
- One technique is to XOR the S-box inputs with the key hits before the S-box is applied.
- From k, one derives round key $k_1, k_2, \ldots, k_h, k_{h+1}$ using a key scheduling algorithm, where h denotes the number of rounds.

3.2. Note: The *Heys cipher* is a toy cipher discussed in his paper, "A Tutorial in Linear and Differential Cryptanalysis", with the following properties:

- 4-round SPN.
- 16-bit block size.
- All S-boxes are identical (which is the first line of the DES S_1).

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

- All permutations are identical (the butterfly or transpose permutation).

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

- No key scheduling algorithm (i.e., each round's subkey is independently chosen).
- 80-bit key.

The K_5 subkey prevents an adversary from reversing the final round of substitution and permutation. This technique is called *key whitening*.

3.3. Note: We introduce some notations.

- $P = (P_1 P_2 \ldots P_{16})$: plaintext, where P_j denotes its jth bit.
- $C = (C_1 C_2 \ldots C_{16})$: ciphertext, where C_j denotes its jth bit.
- $K_i = (K_{i,1} K_{i,2} \ldots K_{i,16})$: ith subkey, where $K_{i,j}$ denotes its jth bit.
- $U_i, i \in \{1, 2, 3, 4\}$: the 16-bit block of bits at the input of the ith round of S-boxes.
- $V_i, i \in \{1, 2, 3, 4\}$: the 16-bit block of bits at the output of the ith round of S-boxes.
- $U_{i,j}$ and $V_{i,j}$ have similar meaning to $K_{i,j}$.

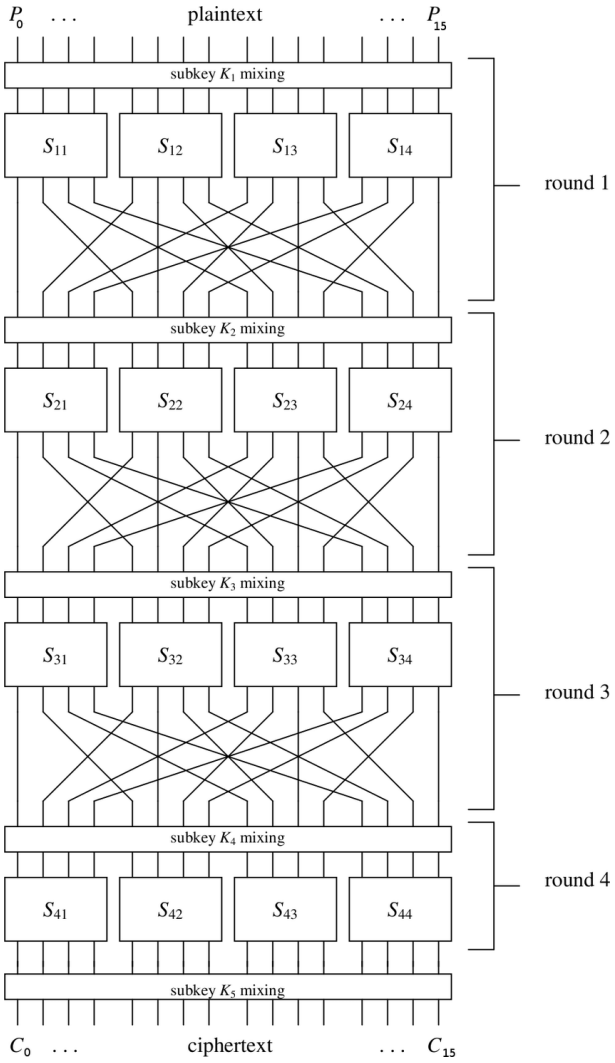


Figure 2.1: Heys Cipher

3.4. Note: The basic idea of **linear cryptanalysis** is to look for linear (boolean) relations among the bits which hold with probability much different from 50%. Intuitively, a perfect cipher should not give you any hints to exploit (partial information) and everything should look random (i.e., holds with probability 50%). Our goal is to identify cases where the probability is far from random. For example,

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \cong 0$$

In the above equation, $U_{4,6}, U_{4,8}, U_{4,14}$, and $U_{4,16}$ depend only on the ciphertext C and the eight key bits $K_{5,5}, K_{5,6}, K_{5,7}, K_{5,8}, K_{5,13}, K_{5,14}, K_{5,15}, K_{5,16}$. Given enough known-plaintext pairs (P, C), we can guess the appropriate key bits until a guess is found for which the relation holds with probability much different from 50%, over all the known (P, C)-pairs.

3.5. Note: Our goal is to find linear relations which hold with abnormally large or abnormally small probability. Let us start with the S-box. Let X_1, X_2, X_3, X_4 be the four input bits (from left to right) and Y_1, Y_2, Y_3, Y_4 be the four output bits.

X	X_1	X_2	X_3	X_4	Y	Y_1	Y_2	Y_3	Y_4	$X_2 \oplus X_3$	$Y_1 \oplus Y_3 \oplus Y_4$
0	0	0	0	0	14	1	1	1	0	0	0
1	0	0	0	1	4	0	1	0	0	0	0
2	0	0	1	0	13	1	1	0	1	1	0
3	0	0	1	1	1	0	0	0	1	1	1
4	0	1	0	0	2	0	0	1	0	1	1
5	0	1	0	1	15	1	1	1	1	1	1
6	0	1	1	0	11	1	0	1	1	0	1
7	0	1	1	1	8	1	0	0	0	0	1
8	1	0	0	0	3	0	0	1	1	0	1
9	1	0	0	1	10	1	0	1	0	0	1
10	1	0	1	0	6	0	1	1	0	1	1
11	1	0	1	1	12	1	1	0	0	1	0
12	1	1	0	0	5	0	1	0	1	1	1
13	1	1	0	1	9	1	0	0	1	1	0
14	1	1	1	0	0	0	0	0	0	0	1
15	1	1	1	1	7	0	1	1	1	0	0

So how often does $X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4$? Observe there are only 4 out of 16 cases that this equality does not hold. We say that this linear relation holds with probability $3/4$ and fails with probability $1/4$. This is quite different from $1/2$! We say that the **bias** of this linear relation is $1/4 = 3/4 - 1/2$. More formally, the bias of a probability p is defined to be $p - 1/2$.

As another example, if you compute the results of $X_1 \oplus X_4 = Y_2$, then you see that the relation holds exactly $1/2$ of the times and there is no bias.

The bias can be negative too, which means the relation is *unlikely* to hold. One example is $X_3 \oplus X_4 = Y_1 \oplus Y_4$, which holds with probability $2/16$ an the bias is $-6/16 = -3/8$.

To summarize,

- We have:
  - $X_2 \oplus X_3 \cong Y_1 \oplus Y_3 \oplus Y_4$ with probability $12/16 = 3/4$
  - $X_1 \oplus X_4 \cong Y_2$ with probability $8/16 = 1/2$
  - $X_3 \oplus X_4 \cong Y_1 \oplus Y_4$ with probability $2/16 = 1/8$
- For the S-box from the Heys cipher,
  - $X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 \cong 0$ with probability $3/4$ (bias $1/4$)
  - $X_1 \oplus X_4 \oplus Y_2 \cong 0$ with probability $1/2$ (bias 0)
  - $X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 \cong 0$ with probability $1/8$ (bias $-3/8$)

3.6. Note: We can build a **linear approximation table** from this.

Input Sum \ Output Sum	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	–2	–2	0	0	–2	+6	+2	+2	0	0	+2	+2	0	0
2	0	0	–2	–2	0	0	–2	–2	0	0	+2	+2	0	0	–6	+2
3	0	0	0	0	0	0	0	0	+2	–6	–2	–2	+2	+2	–2	–2
4	0	+2	0	–2	–2	–4	–2	0	0	–2	0	+2	+2	–4	+2	0
5	0	–2	–2	0	–2	0	+4	+2	–2	0	–4	+2	0	–2	–2	0
6	0	+2	–2	+4	+2	0	0	+2	0	–2	+2	+4	–2	0	0	–2
7	0	–2	0	+2	+2	–4	+2	0	–2	0	+2	0	+4	+2	0	+2
8	0	0	0	0	0	0	0	0	–2	+2	+2	–2	+2	–2	–2	–6
9	0	0	–2	–2	0	0	–2	–2	–4	0	–2	+2	0	+4	+2	–2
A	0	+4	–2	+2	–4	0	+2	–2	+2	+2	0	0	+2	+2	0	0
B	0	+4	0	–4	+4	0	+4	0	0	0	0	0	0	0	0	0
C	0	–2	+4	–2	–2	0	+2	0	+2	0	+2	+4	0	+2	0	–2
D	0	+2	+2	0	–2	+4	0	+2	–4	–2	+2	0	+2	0	0	+2
E	0	+2	+2	0	–2	–4	0	+2	–2	0	0	–2	–4	+2	–2	0
F	0	–2	–4	–2	–2	0	+2	0	0	–2	+4	–2	–2	0	+2	0

For a linear relation

$$a_1X_1 \oplus a_2X_2 \oplus a_3X_3 \oplus a_4X_4 = b_1Y_1 \oplus b_2Y_2 \oplus b_3Y_3 \oplus b_4Y_4$$

where $a_i, b_i \in \{0,1\}$ for $i = 1,2,3,4$, the *input sum* is the value of $a_1a_2a_3a_4$ in binary and the output sum is the value $b_1b_2b_3b_4$ in binary. The bias of this linear relation is $x/16$ where x is the value of the table cell.

For example, consider $X_3 \oplus X_4 \cong Y_1 \oplus Y_4$, or equivalently, $X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 \cong 0$. We have $a_1a_2a_3a_4 = 0011_2 = 3_{10}$ and $b_1b_2b_3b_4 = 1001_2 = 9_{10}$, and the corresponding value in table cell is -6; the bias is $-6/16 = -3/18$.

3.7. Note: How do probabilities stack? Suppose

$$\text{Prob}\,(X_1 = i) = \begin{cases} p_1 & i = 0 \\ 1 - p_1 & i = 1 \end{cases} \qquad \text{Prob}\,(X_2 = i) = \begin{cases} p_2 & i = 0 \\ 1 - p_2 & i = 1 \end{cases}$$

If X_1 and X_2 are independent, then

$$\text{Prob}\,(X_1 = i, X_2 = j) = \begin{cases} p_1p_2 & i = 0, j = 0 \\ p_1\,(1 - p_2) & i = 0, j = 1 \\ (1 - p_1)\,p_2 & i = 1, j = 0 \\ (1 - p_1)\,(1 - p_2) & i = 1, j = 1 \end{cases}$$

Then the probability that $X_1 \oplus X_2 = 0$ is

$$\begin{aligned}\text{Prob}\,(X_1 \oplus X_2 = 0) &= \text{Prob}\,(X_1 = 0, X_2 = 0) + \text{Prob}\,(X_1 = 1, X_2 = 1) \\ &= p_1 p_2 + (1 - p_1)\,(1 - p_2)\end{aligned}$$

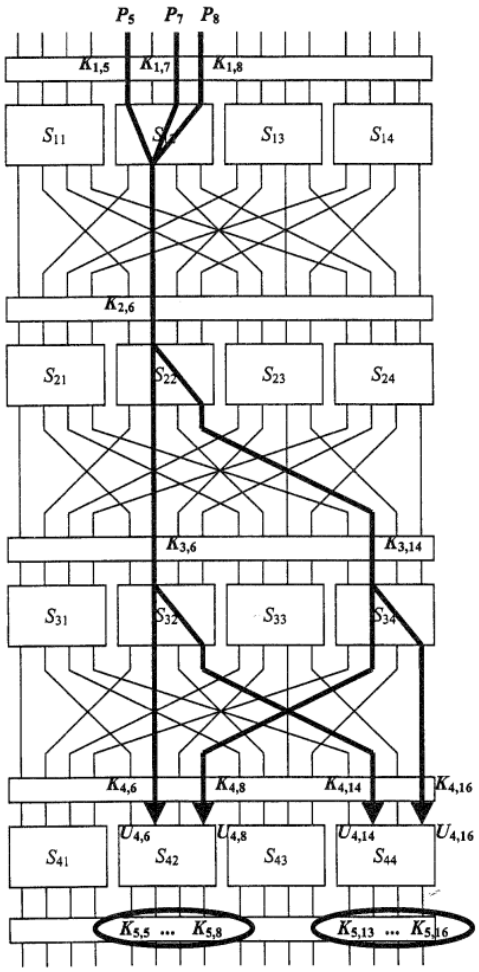
Now suppose $p_1 = 1/2 + \varepsilon_1$ has bias ε_1 and $p_2 = 1/2 + \varepsilon_2$ has bias ε_2. Then

$$\text{Prob}\,(X_1 \oplus X_2 = 0) = p_1 p_2 + (1 - p_1)\,(1 - p_2) = \frac{1}{2} + 2\varepsilon_1\varepsilon_2.$$

Hence, $\text{Prob}(X_1 \oplus X_2 = 0)$ has bias $2\varepsilon_1\varepsilon_2$. Generalizing this, we obtain the following Theorem.

3.8. Theorem (Piling-Up Lemma, Matsui, 1993): *Let $X_1, X_2, \ldots, X_n$ be independent binary random variables with bias $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, respectively. Then*

$$\text{Prob}\,(X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i.$$



Section 4. Differential Cryptanalysis

4.1. Note: Differential cryptanalysis is a *chosen-plaintext attack*. The attacker must choose certain plaintexts strategically and obtain the corresponding ciphertexts.

- Let P, P' be two plaintexts.
- Let C, C' be their encryptions.
- Set $\Delta P := P \oplus P'$ and $\Delta C := C \oplus C'$.
- Look for values of ΔC which occur with abnormally high probability for a given ΔP.

As with linear cryptanalysis, we first perform these steps on the S-box, then gradually scale up to the entire cipher.

- Let X and X' be two plaintexts.
- Let Y and Y' be their encryptions.
- Set $\Delta X := X \oplus X'$ and $\Delta Y := Y \oplus Y'$.
- Tabulate, for each ΔX, the possible values for ΔY.
- Look for values of ΔY which occurs with abnormally high probability for a given ΔX.

4.2. Remark: Fill the rest.

Section 5. Stream Ciphers

5.1. Note: A block cipher is a SKES which breaks up the plaintext into blocks of a fixed length, and encrypts the block one at a time. In contrast, a **stream cipher** encrypts a plaintext one character (usually a bit) at a time.

5.2. Note: Stream ciphers are a practical version of one-time pads. Instead of using a random key, use a pseudorandom keystream derived from a short key.

- Advantages vs block ciphers: speed, simplicity, ease of hardware implementation.
- Disadvantages vs block ciphers: linearity (each bit of plaintext influences exactly one bit of cipher text) and potential disaster if keystream is re-used.

The keystream should be "indistinguishable" from a random sequence. In particular, don't use UNIX random number generator for cryptography!