# Block Ciphers and DES

2021-03-23

## Contents

## 1 Block ciphers

Block ciphers are symmetric ciphers which operate on groups of bits called *blocks*. We could think of a block like an *array* of bits.

This means that both the encryption algorithm and the decryption algorithm take as input a block:

$$\begin{tikzcd}
P \arrow[r] & \boxed{Enc_k} \arrow[r] & C \\
C \arrow[r] & \boxed{Dec_k} \arrow[r] & P
\end{tikzcd}$$

Where:

$$P = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \qquad C = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

Notice that the plaintext block and ciphertext block have the same length.

### 1.1 Rounds

A block algorithm is based on the repetition of short sequences of operations called *rounds*. A round is a basic transformation which operates on a block. For example, an encryption algorithm could consist of three rounds: $C = R_3(R_2(R_1(P)))$. Each round should also have an *inverse* in order to compute back the plaintext from the ciphertext: $P = R_1^{-1}(R_2^{-1}(R_3^{-1}(C)))$.

There are two main techniques to build rounds: *substitution-permutation networks* and *Feistel schemes*.

#### 1.1.1 Round keys and key schedule algorithm

Usually round functions $(R_1, \dots R_n)$ are the same, but they are parametrized by a *round key*. A round key is a key which derives from the main key $K$. The same round function with different round keys will behave differently, and therefore will produce a different output blocks.

An algorithm called *key schedule* produces the round keys starting from the main key $K$. Round keys should always be different from each other in every round.

#### 1.1.2 Substitution-permutation networks

Substitution-permutation networks put together two important properties of cryptography[1]:

[1] The terms *confusion* and *diffusion* were introduced by Claude Shannon.

**Confusion:** the input (plaintext and key) undergoes complex transformations, which make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible.

**Diffusion:** the transformations depend equally on all bits of the input, i.e. the ciphertext doesn't reflect the statistical properties of the plaintext.

In a block cipher, a simple diffusion element is the bit *permutation* (which is used frequently in DES), while confusion is achieved by the use of *substitution* (used both in DES and in AES).

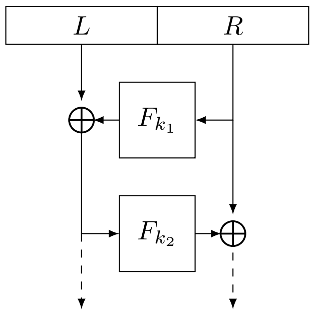
**Substitution:** Substitution boxes (S-boxes) are small lookup tables that transform chunks of 4 or 8 bits. For example, the 4-bit nibble 0000 could be mapped to 0011, while 0101 could be mapped to 0110, and so on. These S-boxes must be cryptographically strong (i.e. they should be as nonlinear as possible and have no statistical bias).

**Permutation:** The permutation could simply constist of a permutation of the bits, which is easy to implement but doesn't create to much diffusion. In practice different ciphers use operations from the linear algebra to mix up the bits, like matrix-multiplications, and so on. Those operations create strong dependencies with all the bits of the input, and therefore ensure strong diffusion.

### 1.1.3 Feistel schemes

A Feistel scheme works as follows.:

1. The input block is splitted in two halves, $L$ and $R$
2. $L$ is XOR-ed with $F(R)$, where $F$ is a substitution-permutation round
3. $L$ and $R$ are swapped
4. Step 2. and 3. are repeated a bunch of times
5. $L$ and $R$ are merged into the output block



Notice that $F_{k_1}$ and $F_{k_2}$ are actually the same function, they just take as input different round keys $K_1$ and $K_2$, which come from the key scheduling algorithm.

## 1.2 4×4 S-box cipher

Howard M. Heys introduced this simple substitution-permutation based block cipher in his lectures *A Tutorial on Linear and Differential Algebra*.

**Substitution**

- The algorithm operates on 16-bit blocks.
- Each block is broken into four 4-bit sub-blocks.
- Each sub-block is fed into a $4 \times 4$ S-box (a substitution-box with 4 input bits and 4 output bits).
- In this case we use the same mapping for all S-boxes[2]
- The S-box performs a *nonlinear* mapping, i.e. the output bits cannot be represented as a linear function of the input bits.

The S-box used is the following:

| input | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| output | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

You can verify that the mapping is nonlinear[3] by simply observing that the number 0 is mapped to the hexadecimal value E (which is 14 in decimal), while a linear function always maps the null-element 0 into 0 itself. Also, the mapping is not *affine*:[4] while $\text{E}_{hex} = 14_{dec} = 0 + 14$, instead $4 \neq 1 + 14$.

**Permutation** The permutation consists of a simple transposition of the bits, or the permutation of the bit positions. In the following *permutation table* we represent for each input bit the position that it will take in the output (for example, the bit 2 is moved to position 5). We stick with the author convention, in which 1 is the rightmost bit and 16 is the leftmost bit.

| input | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| output | 1 | 5 | 9 | 13 | 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 | 12 | 16 |

**Key mixing** To achieve key mixing, data blocks are XOR-ed with a round key before being fed into a S-box.

[2] In DES all the S-boxes in a round are different.
[3] A linear function maps a certain value $x$ into $kx$, where $k \in \mathbb{R}^n$
[4] An affine function maps a certain element $x$ into $kx + c$ where $k, c \in \mathbb{R}^n$.

# 2 DES

## 2.1 History

In 1972 the US National Bureau of Standards (NBS, currently NIST) initiated a request for proposal for a standardized cipher in the USA. Up to this point in time cryptography had always been kept secret. By the early 70s, however, cryptography had become of crucial importance for a variety of commercial applications such as banking.

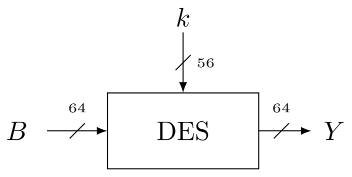
The winning proposal selected in 1974 came from a team of cryptographers working at IBM. They submitted an algorithm based on the family of Feistel's ciphers called *Lucifer*, developed in the late 60s.

In 1977 the NBS finally released all specifications of the IBM cipher (which meanwhile had passed through some modifications) as *Data Encryption Standard* to the public. However, the motivation for parts of the DES design[5] were never officially released. During the following years, with the increase of personal computers in the early 80s, it became easier to analyze the inner structure of the cipher. However, no serious weaknesses were found until 1990.

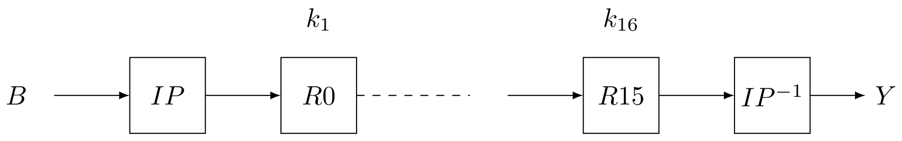
In 1999 DES was replaced by AES.

## 2.2 Description

DES is a symmetric block cipher which encrypts 64-bit blocks with a 56-bit key.



Each block passes through an initial bitwise permutation $IP$, then 16 rounds which all perform the identical operation, and eventually a final bitwise permutation $IP^{-1}$, which is the inverse of the initial permutation $IP$.



In each round a different round key $k_i$ derived from the main key $k$ (with a key schedule) is used. Each DES round implements a Feistel scheme.

[5] i.e. the design criteria.

### 2.2.1 Internal permutation

The internal permutation $IP$ maps each input bit to an output bit. This doesn't increase security, it's just to arrange the plaintext. The final permutation $IP^{-1}$ performs the reverse mapping. The following tables show the final position of each bit from 0 to 63. The appendix shows both the $IP$ and the $IP^{-1}$ structure.

### 2.2.2 The $f$ function

The Feistel function $f$ plays a crucial role for the security of the DES cipher. In round $i$ it takes the right half $R_{i-1}$ (32-bits) of the output of the previous step and the round key $k_i$. The output is then XOR-ed with the left half of the previous step $L_{i-1}$ (32-bits).

- First, the 32-bit input is expanded to 48-bits by partitioning the input into eight 4-bit blocks which are expandend to 6-bit blocks with a $E$-box (a special type of permutation). You can find the $E$-box in the appendix.

- Then the 48-bit result of the expansion is XOR-ed with the round key $k_i$, and the eight 6-bit blocks are fed into eight different S-boxes; each S-box maps a 6-bit block to a 4-bit block[6]. Each S-box contains $2^6 = 64$ entries. Each entry is a 4-bit value. The appendix shows all the S-boxes.

  The *design criteria* behind the S-boxes are the following:

  1. Each S-box has 6 input bits and 4 output bits.
  2. No output bit of an S-box should be too close to a linear combination of the input bits.
  3. If the lowest and highest bits of the input are fixed and the four middle bits are varied, each of the possible 4-bit output values must occur exactly once.
  4. If two input to an S-box differ in exactly one bit, their outputs must differ in at least two bits.
  5. If two inputs to an S-box differ in the two middle bits, their outputs must differ in at least two bits.
  6. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must be different.
  7. For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
  8. A collision (zero output difference) at the 32-bit output of the eight S-boxes is only possible for three adjacent S-boxes.

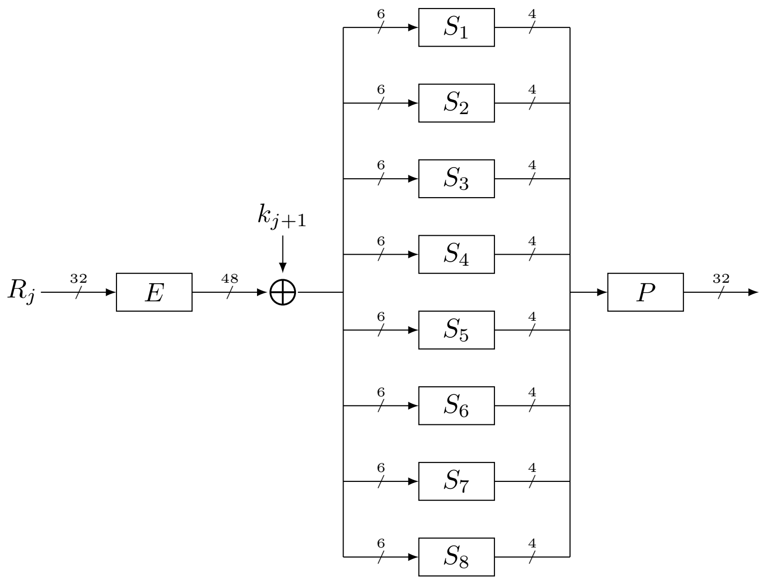
  See Exercise 4.2.7 in which properties 3, 4, 5, 6 are checked.

[6] Eight 4×6 tables were close to the maximum size which could be fit on a single integrated circuit in 1974.

- Finally, the 32-bit output passes through the $P$ permutation (see the appendix). This permutation introduces diffusion, because the four output bits of each S-box are permuted in such a way that they affect several different S-boxes in the following round.

  The diffusion created by expansion, S-boxes and the $P$ permutation guarantees that each bit at the end of the fifth round is a function of every plaintext bit and every key bit (*avalanche effect*).

The following picture represents the internal structure of the $f$ function:



See Exercise 4.2.11 for an example of computation with the $f$ function.

### 2.2.3 The importance of being nonlinear

The S-boxes are at the core of DES in terms of cryptographic strength, because they produce confusion and are the *only* nonlinear element in the algorithm, i.e.:

$$S(a) \oplus S(b) \neq S(a \oplus b)$$

Why is nonlinearity so crucial? With a nonlinear building block, an attacker could express the DES input and output with a system of linear equations where the key bits are the unknowns. Such systems can easily be solved. S-boxes were carefully designed to prevent advanced mathematical attacks, such as *differential cryptanalysis*.

See Exercise 4.2.8 and Exercise 4.2.9.

## 2.3 Key schedule

The key schedule derives 16 round keys of from the main key $k$, each one consisting of 48-bits.

First, note that the DES input key is often stated as 64-bit, but every 7 bits there's an odd parity bit for the previous 7 bits. It's not clear why DES was specified that way, however the eight parity bits are not part of the actual key, they do not improve security, and therefore DES is a 56-bit key cipher, not a 64-bit one.

- The parity bit are first removed from the key. Then the remaining bits go through a permutation $PC-1$ (where $PC-1$ stands for *permuted choice one*). This permutation is shown in the appendix.

- The result is split into two halves $C_0$ and $D_0$. The two 28-bit halves are cyclically shifted according to the following rules:

  + In rounds $i = 1, 2, 9, 16$ the two halves are rotated left by one bit.

  + In the other rounds, the two halves are rotated left by two bits.

- To generate the 48-bit round keys $k_i$, the two halves are permuted again with $PC-2$ (*permuted choice 2*, see appendix), which ignores the following bits: `9, 18, 22, 25, 35, 38, 43, 54`.

## 2.4 Decryption

One advantage of DES is that decryption is the same function as encryption. This is because DES is based on a Feistel network. The only difference is the key schedule: the round keys are provided in reverse order (i.e. in round 1 round key 16 is needed, in round 2 key 15, and so on).

Because the total nubmber of rotations in the key schedule algorithm is 28, we have an interesting property: $C_0 = C_{16}$ and $D_0 = D_{16}$. Therefore, $k_{16}$ can be directly derived after $PC-1$[7]:

$$k_{16} = PC2(C_{16}, D_{16}) = PC2(C_0, D_0) = PC2(PC1(k))$$

$k_{15}$ can be derived from $C_{16}, D_{16}$ through cyclic right shifts ($RS_i$):

$$k_{15} = PC2(C_{15}, D_{15}) = PC2(RS_2(C_{16}),\ RS_2(D_{16})) = PC2(RS_2(C_0), RS_2(D_0))$$

- In round 1 the key is not rotated.

- In rounds 2, 9 and 16 the two halves are rotated right by one bit.

- In the other rounds the two halves are rotated right by two bits.

[7]From now, we will omit the '-' for clarity reasons.

## 2.5 DES security

Ciphers can be attacked in several ways. With respect to cryptographic attacks, we distinguish between *exhaustive key search* (or *brute-force attacks*) and *analytical attacks*.

Although current analytical attacks against DES are not very efficient, DES can relatively easily be broken with an exhaustive key-search attack. This means that plain DES is not to be considered secure anymore.

RSA security proposed several challenges to break DES, in order to prove that a 56-bit key is too short. During DES challenge III (1999), the EFF[8] decrypted a DES-encrypted message in 22 hours and 15 minutes by using Deep Crack, a custom specifically built microchip.

### 2.5.1 3DES

A much stronger version of DES is 3DES, which consists of three subsequent DES encryptions with keys $k_1$, $k_2$, $k_3$. Usually 3DES is also implemented as EDE, i.e. the message is first encrypted with $k_1$, then decrypted with $k_2$ and finally encrypted with $k_3$.

3DES seems resistant to both brute-force attacks and any known analytical attack.

### 2.5.2 2DES and meet-in-the-middle attack

We saw that 3DES is considered enough secure, but one could ask "Why not 2DES?". The reason is quite simple: every block encryption algorithm, if applied two times, is vulnerable to the *meet-in-the-middle-attack*.

Let's consider a keylength of $\kappa$. Suppose that we first encrypt a plaintext by using a key $k_1$ and then we encrypt the ciphertext by using a key $k_2$.

$$\begin{tikzcd} x \arrow[r] & \boxed{E_{k_1}} \arrow[r] & \boxed{E_{k_2}} \arrow[r] & y \end{tikzcd}$$

A plain brute-force attack would require us to search through all possible combination of both keys, i.e. the effective key length would be $2\kappa$, and an exhaustive search would require $2^\kappa \cdot 2^\kappa = 2^{2\kappa}$ encryptions.

The meet-in-the-middle attack requires much more computational power:

- We (the adversary) know both the plaintext $x$ and the ciphertext $y$.
- We first compute $2^\kappa$ encryptions of the plaintext $x$, one for each possible value of the key. We build a lookup table with all the encrypted values, indexed by the value of $k_1$ that we used. In the case of DES, this would consist in $2^{56}$ operations.

[8] Electronic Frontier Foundation.

- Then we start decrypting the ciphertext $y$ with all possible values of the key $k_2$ ($2^\kappa$). If we find a match inside the lookup table, then we have the key $k_2$. Not only: remember that the lookup table is indexed by the value of $k_1$, this means that we also have $k_1$.

In the end, in the worst case we perform $2^\kappa + 2^\kappa = 2 \cdot 2^\kappa = 2^{\kappa+1}$ operations, which are definitely less than $2^{2\kappa}$. That's why double-encryption should be avoided with all the block ciphers.

**Example:** Let's say we have a plaintext $p$ and a ciphertext $c$. First we must encrypt $p$ with all possible $2^\kappa$ values of the key[9]:

| k | $\mathtt{Enc_k(p)}$ |
|---|---|
| 0 | 31269 |
| 1 | 161804 |
| ... | ... |
| ... | ... |
| $2^{\kappa-2}$ | 21838121 |
| $2^{\kappa-1}$ | 193490 |

Then we start decrypting $c$ with all the possible $k_2$ key values.

$$\mathtt{k} = \mathtt{0} \Rightarrow \mathtt{Dec_k(c)} = \mathtt{1235443}$$

$$\mathtt{k} = \mathtt{1} \Rightarrow \mathtt{Dec_k(c)} = \mathtt{21838121}$$

We're lucky! It took us just two additional decryption to find a match. We look in the table and we see that the value 21838121 is in position $2^{\kappa-2}$. In the end, the key $k_1$ is equal to $2^{\kappa-2}$, while the key $k_2$ is equal to 1, and we performed just $2^\kappa + 2$ computations.

## 3 Exercises

**Exercise 4.2.7** Check properties 3, 4, 5, 6 for S-box $S_1$.

- (3) It's easy to check that each row contains different numbers. This means that for a fixed pair (first bit - last bit) there are not two different middle bits configurations that produce the same output.

- (4) Let's consider all the inputs which differ in exactly one bit; you can check that the corresponding output bits will differ in at least two bits. Let's take as an example the two inputs 000000, which is mapped to $14 = 1110_2$, and 000001, which is mapped to $0 = 0000_2$. While 000000 and 000001 differ in one bit, 1110 and 0000 differ in three bits.

[9]For the sake of simplicity we consider just the decimal values.

- (5) Take as an example 000000, which is mapped to $14 = 1110_2$ and 001100, which is mapped to $11 = 1011_2$. As you can see 1110 and 1011 differ in two bits.
- (6) Take as an example 000000, which is mapped to $14 = 1110_2$ and 110000, which is mapped to $15 = 1111_2$. The output is clearly different.

**Exercise 4.2.8** Check that $S_1(4) \oplus S_1(23) \neq S_1(4 \oplus 23)$.

$$4 = 000100_2 \qquad 23 = 010111_2$$
$$S_1(4) = 13 = 1101_2 \qquad S_1(23) = 15 = 1111_2$$
$$S_1(4 \oplus 23) = S_1(010011) = 6 = 0110_2$$
$$S_1(4) \oplus S_1(23) = 0010_2 = 2$$
$$2 \neq 6$$

**Exercise 4.2.9** Check $S_1(0) \neq 0$. This shows that $S_1$ is non linear.

$$S_1(0) = 14 \neq 0$$

**Exercise 4.2.11** Compare the output of $f$ for inputs $R_j = 0$ and $R'_j = 1$ with $k_{j+1} = 0$.

- First we compute the expansion through the $E$-box of both $R_j$ and $R'_j$:

$$E(R_j) = 000\ldots000 = 0^{48} \qquad E(R'_j) = 111\ldots111 = 1^{48}$$

- Then the expanded blocks are XOR-ed with the key $k_{j+1} = 0^{48}$:

$$E(R_j) \oplus k_{j+1} = 0^{48} \qquad E(R'_j) \oplus k_{j+1} = 1^{48}$$

- Now we divide the 48-bit blocks in eight 6-bit sub blocks, which are sent to the S-boxes. In this the sub blocks $B_j$ are all the same:

$$B_j = 0^6 \qquad B'_j = 1^6$$

  Let's apply the S-boxes to each block:

| | | | | | |
|---|---|---|---|---|---|
| $S_1(B_j) =$ | $14 =$ | 1110 | $S_1(B'_j) =$ | $13 =$ | 1101 |
| $S_2(B_j) =$ | $15 =$ | 1111 | $S_2(B'_j) =$ | $9 =$ | 0101 |
| $S_3(B_j) =$ | $10 =$ | 1010 | $S_3(B'_j) =$ | $12 =$ | 1100 |
| $S_4(B_j) =$ | $7 =$ | 0111 | $S_4(B'_j) =$ | $14 =$ | 1110 |
| $S_5(B_j) =$ | $2 =$ | 0010 | $S_5(B'_j) =$ | $3 =$ | 0011 |
| $S_6(B_j) =$ | $12 =$ | 1100 | $S_6(B'_j) =$ | $13 =$ | 1101 |
| $S_7(B_j) =$ | $4 =$ | 0100 | $S_7(B'_j) =$ | $12 =$ | 1100 |
| $S_8(B_j) =$ | $13 =$ | 1101 | $S_8(B'_j) =$ | $11 =$ | 1011 |

- Finally, we apply $P$ to the concatenation of all the bits:

$$P(11101111101001110010110001001101) =$$
$$= f(R_j) = 11011000110110001101101110111100$$
$$P(11010101110011100011110111001011) =$$
$$= f(R'_j) = 00111000110100111111100111011011$$

## A Appendix

Read from left to right, top to bottom.

$IP$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

$IP^{-1}$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

$E$

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

*S-boxes*

**How to read:** each row is indexed by the concatenation of the rightmost and the leftmost bit of the 6-bit value, while each column is indexed by the 4 middle bits (represented in decimal for compactness reasons). For example if the input is `001101`, the row is the one with label `01` (first and last bit) while the column is the one with label `0110` = `6`.

| S1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 01 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 10 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 11 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

| S2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 01 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 10 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 11 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |

| S3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 01 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 10 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 11 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |

| S4 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| 01 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| 10 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| 11 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |

| S5 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| 01 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| 10 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| 11 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |

| S6 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| 01 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| 10 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| 11 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |

| S7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| 01 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| 10 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| 11 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |

| S8 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| 01 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| 10 | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| 11 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

$P$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 7 | 20 | 21 | 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 | 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 | 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 | 22 | 11 | 4 | 25 |

$PC-1$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 60 | 52 | 44 | 36 | 63 | 55 | 47 | 39 |
| 31 | 23 | 15 | 7 | 62 | 54 | 46 | 38 |
| 30 | 22 | 14 | 6 | 61 | 53 | 45 | 37 |
| 29 | 21 | 13 | 5 | 28 | 20 | 12 | 4 |

$PC-2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 | 3 | 28 |
| 15 | 6 | 21 | 10 | 23 | 19 | 12 | 4 |
| 26 | 8 | 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 | 30 | 40 |
| 51 | 45 | 33 | 48 | 44 | 49 | 39 | 56 |
| 34 | 53 | 46 | 42 | 50 | 36 | 29 | 32 |