# Computer Science 308-547A
# Cryptography and Data Security

Claude Crépeau

These notes are, largely, transcriptions by Anton Stiglic of class notes from the former course *Cryptography and Data Security (308-647A)* that was given by prof. Claude Crépeau at McGill University during the autumn of 1998-1999. These notes are updated and revised by Claude Crépeau.

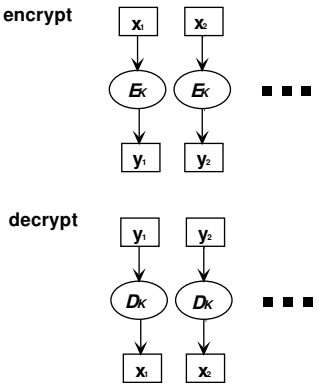
# 9 modes of operation using a block cipher

In a block cipher, when the data to be encrypted is larger than the size of the block that the cipher takes as input, one must follow a protocol that allows encryption and decryption of this data. Four *modes of operation* have been defined in [**?**] (initially presented for DES, but are good for any block cipher encryption algorithm). We present each one of them and discuss them shortly. In each case, we consider $n$ blocks of cleartext $x_1 x_2 \dots x_n$ which are encrypted to $n$ blocks of ciphertext $y_1 y_2 \dots y_n$ using an encryption algorithm $e_K$ (the decryption algorithm will be denoted $d_K$ as usual). (in DES, $|x_i| = |y_i| = 64$ bits).

## 9.1 Electronic Codebook (ECB) Mode

This is the trivial, most simple mode.

1. Encryption: for $1 \leq j \leq n,\ y_j \leftarrow e_K(x_j)$.
2. Decryption: for $1 \leq j \leq n,\ x_j \leftarrow d_K(y_j)$




Advantages and disadvantages:

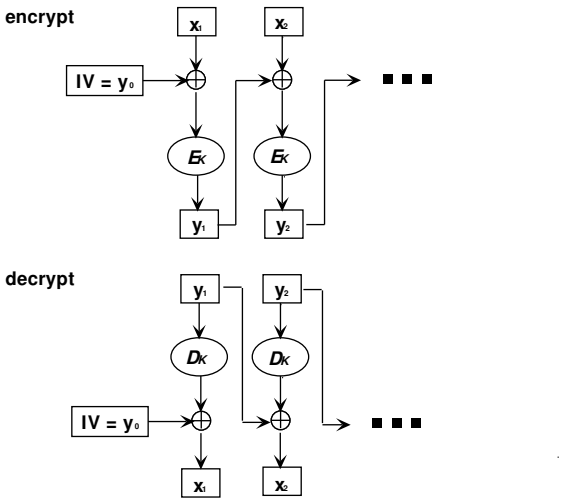
1. Identical plaintext blocks result in identical ciphertext, thus vulnerable if the data is highly structured.
2. Ideal for short messages.
3. Error propagation is limited to the blocks in which the error occurs.
4. Rearranging the order of the ciphertext blocks simply rearranges the order of the decrypted cleartext.

## 9.2 Cipher Block Chaining (CBC) Mode

This is a simple way to prevent identical cleartext blocks of becoming identical ciphertext blocks. In this mode, cipher block $i-1$ is passed through to be XORed with the cleartext block $i$, the result is encrypted to form cipher block $i$. An initialization vector ($IV$) is used to start out. This $IV$ need not be secret but it's integrity should be protected. Authentication is a way of preserving integrity, keeping $IV$ secret is also a solution to preventing tampering when authentication is not available. See [**?**] for some attacks that are possible when the $IV$ is known.

1. Encryption: $y_0 = IV$.
   For $1 \leq j \leq n, y_j \leftarrow e_K(y_{j-1} \oplus x_j)$.

2. Decryption: $y_0 = IV$.
   For $1 \leq j \leq n, x_j \leftarrow y_{j-1} \oplus d_K(y_j)$.




Advantages and disadvantages:

1. Identical plaintexts result when the same plaintext is enciphered and under the same key and $IV$. Changing the $IV$, key, or first plaintext block (say with a counter) results in different ciphertext.

2. An error in a ciphertext block $y_j$ will affect decipherment of blocks $x_j$ and $x_{j+1}$ (we leave it as an exercise to convince yourself of this). If this is the only error, all the other blocks decrypt to the correct plaintext blocks.

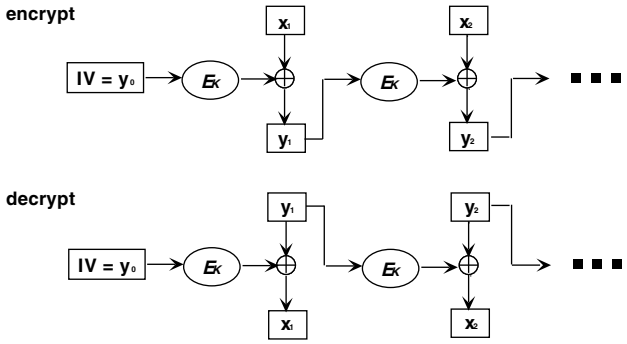
3. Rearrangement of the ciphertext blocks highly affects decryption.

## 9.3 Cipher Feedback (CFB) Mode

It is sometimes useful to be able to send $r-bit$ plaintext blocks, where $r \leq b$ ($b$ being the block length of input of the encryption function). This scheme enables to transform DES into a stream cipher (using the length of bit or byte, i.e. $r = 1$ or $r = 8$). The scheme uses an $IV$. We use a $b$-bit $IV$ and split the cleartext message in r-bit blocks $x_1 x_2 \dots x_u$, $(1 \leq r \leq b)$.

1. Encryption: $I_1 \leftarrow IV$ ($I_j$ is the input value in a shift register)
   For $1 \leq j \leq u$:
   (a) $O_j \leftarrow e_K(I_j)$.
   (b) $pad_j \leftarrow$ the $r$ leftmost bits of $O_j$.
   (c) $y_j \leftarrow x_j \oplus pad_j$.
   (d) $I_{j+1} \leftarrow 2^r I_j + y_j$ (Shift $y_j$ into right end of shift register).
2. Decryption: $I_1 \leftarrow IV$.
   For $1 \leq j \leq u$, upon receiving $y_j$:
   $x_j \leftarrow y_j \oplus pad_j$, where $pad_j, O_j$ and $I_j$ are computed as above.

The following figure depicts the CFB mode for $r = b$:




Advantages and disadvantages:

1. Identical plaintexts: see CBC mode.
2. An error in a ciphertext block $y_j$ will affect decipherment of $x_j$ and the next $\lceil b/r \rceil$ ciphertext blocks (until $b$ bits of ciphertext are processed, after which the error is out of the shift register).

3. This mode can't be used for public key algorithms, since the decryption needs only $e_K$.

## 9.4 Output Feedback (OFB) mode

This mode of operation may also be used to create a stream cipher. We use a $b$-bit $IV$ and split the cleartext message in r-bit blocks $x_1x_2\ldots x_u$, $(1 \leq r \leq b)$.

1. Encryption: $z_0 \leftarrow IV$
   For $1 \leq j \leq u$:
   (a) $z_j \leftarrow e_K(z_{j-1})$.
   (b) $pad_j \leftarrow$ the $r$ leftmost bits of $z_j$.
   (c) $y_j \leftarrow x_j \oplus pad_j$.
2. Decryption: $I_1 \leftarrow IV$.
   For $1 \leq j \leq u$, upon receiving $y_j$:
   $x_j \leftarrow y_j \oplus pad_j$, where $pad_j, z_j$ are computed as above.

We now summarize the advantages and disadvantages:

1. Identical plaintexts: same as CFB and CBC.
2. Error propagation is limited to the precise bit position of the error.

## 9.5 MAC from a bloc cipher

Since CBC and CFB modes propagate errors in the input, these can be used for MAC's. We describe how *Alice* can use CBC mode to produce a MAC for a message $m = x_1x_2\ldots x_n$. *Alice* begins by using an $IV$ consisting of all zeroes. Then *Alice* computes the ciphertext $y_1y_2\ldots y_n$ using CBC mode of operation with a secret key $K$ that she shares with $Bob$, $y_n$ is the MAC. Alice sends the plaintext-MAC pair $(m, y_n)$ to $Bob$. When $Bob$ receives the message with the tag, he can verify that $m$ was really sent by *Alice* by computing $y_n$ the same way *Alice* did, if the $y_n$ is the same as the MAC he received, he is convinced that *Alice* was the one that sent $m$. $Bob$ is convinced since it would be highly unlikely for $Oscar$ to produce a MAC since he does not know the key $K$ and since modifying $m$ even slightly causes the MAC to be modified greatly. Of course, the security relies in the trust of the block cipher algorithm used).

# 10 DES

The Data Encryption Standard (DES) was introduced in 1977 (a complete description was introduced in the *Federal Information Processing Standards (FIPS) Publication* number 46 of 1977, a more recent description can be found in [**?**]). It introduced the Data Encryption Algorithm (DEA), which is a symmetric encryption scheme that has been used as a standard for a number of years (until 1998). Currently, the Advanced Encryption Standard (AES) is becoming the new standard, the encryption. DEA can be mathematically modeled much better than most other symmetric schemes, also its security has been studied for many years and for these reasons we introduce this protocol.

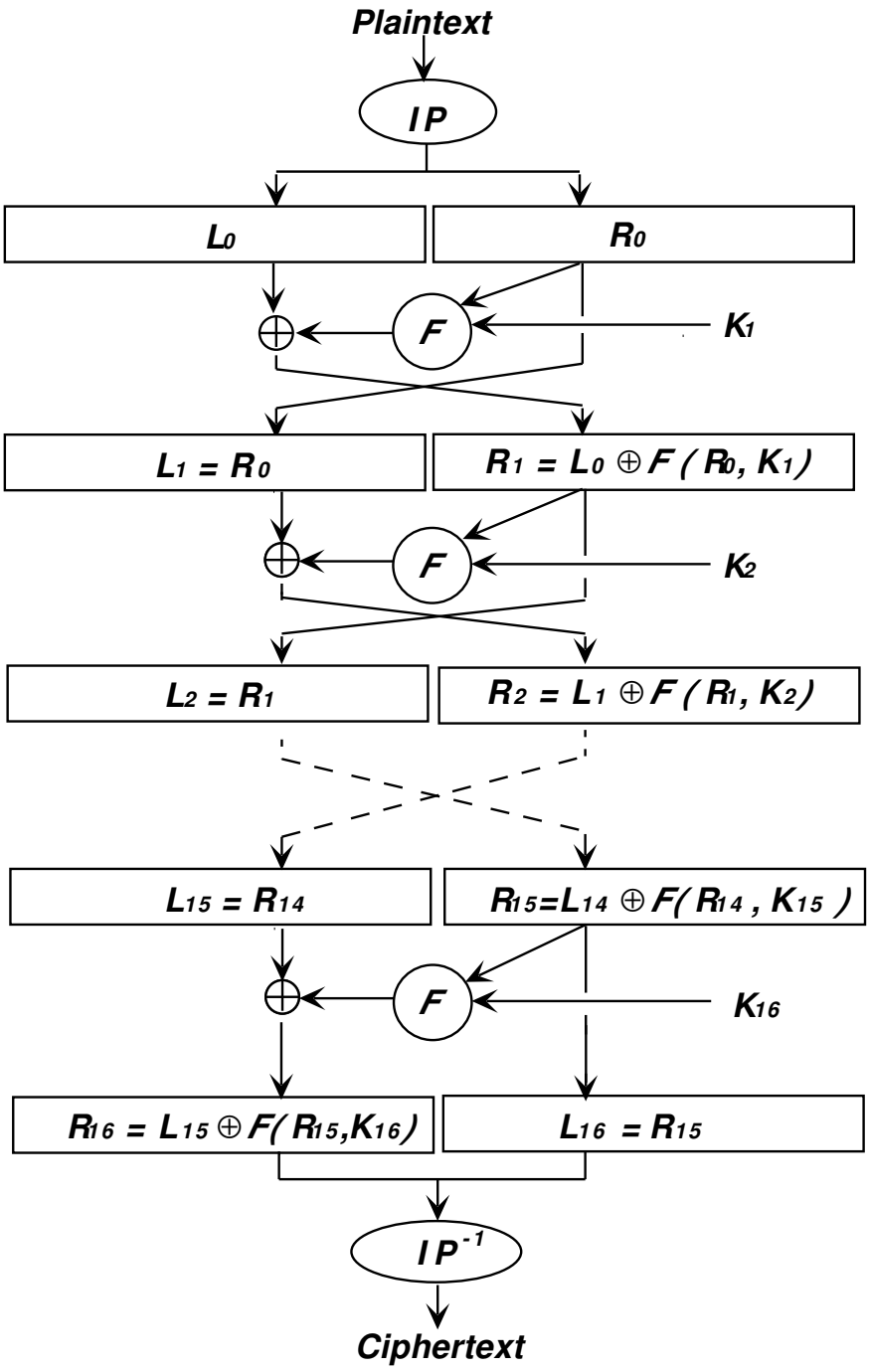
DES is a block cipher which encrypts blocks of 64 bits, with a 56 bit key, into ciphertexts of 64 bits. This key is usually found in a 64 bit format, where the bits 8, 16,.., 64 serve as parity bits (they are set so that each byte of the resulting key contains an odd number of 1's). The key used for encryption is the same key that is used for decryption (this is what defines symmetric schemes).

We start by giving a general description of the protocol. The block that is to be encrypted is subjected to an initial permutation $IP$, the result is split into two halves, the left 32 bit half will be called $L_0$, the right half $R_0$. Then 16 iterations of a function $f$ (which we will describe later) are then computed. $L_iR_i$, $1 \leq i \leq 16$, is computed by the following rule (known as a Fiestel structure):

$$L_i = R_{i-1}$$

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

The $K_i$'s are bit-strings of 48 bits resulting from a function $KS$ of the initial key $K$. $KS$ is called the key schedule function (described latter). Finally, we reverse the two resulting blocks and compute the inverse of $IP$ on them (i.e., we compute $IP^{-1}(R_{16}, L_{16})$).


Plaintext
IP
L0
R0
F
K1
L1 = R0
R1 = L0 ⊕ F ( R0, K1)
F
K2
L2 = R1
R2 = L1 ⊕ F ( R1, K2)
L15 = R14
R15=L14 ⊕ F( R14 , K15 )
F
K16
R16 = L15 ⊕ F( R15,K16)
L16 = R15
IP-1
Ciphertext

## 10.1 $IP$ permutation

The $IP$ permutation is described by the following table:

| 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 |

This means that the 58th bit of $x$ is the first bit of $IP(x)$, the 50th bit is the second bit of $IP(x)$, etc... The inverse of $IP$, $IP^{-1}$, can be computed using this next table:

| $IP^{-1}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

## 10.2 The key selection function $KS$

$KS$ is a function from $\{i|1 \leq i \leq 16\} \times \{w|w\ is\ a\ 64 - bitsstring\}$ to $\{w|w\ is\ a\ 48 - bitstring\}$. We have that

$$K_n = KS(n, K)$$

where $K$ is the initial 64-bit key (we recall that 8 bits are only used for parity check). The $K_i$'s are computed from an initial 64-bit key $K$ as follows:

1. Discard the parity-check bits (bit 8, 16, ... 64) and apply *permuted choice 1* ($PC1$) to the remaining bits of $K$. Denote $C_0D_0 = PC1(K)$

where $C_0$ is the first 28 bits of $PC1(K)$ and $D_0$ the remaining 28-bit string.

2. For $i = 1$ to 16, compute

$$C_i = LS_i(C_{i-1}),$$

$$D_i = LS_i(D_{i-1}),$$

We then have that

$$K_i = PC2(C_i, D_i).$$

$LS_i$ is just a cyclic shift (to the left) defined as follows:

**if** $i = 1, 2, 9, 16$ **then** shift one position

**else** shift two positions.

Permuted choice 1 ($PC1$) is defined by the following table:

PC1

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 |

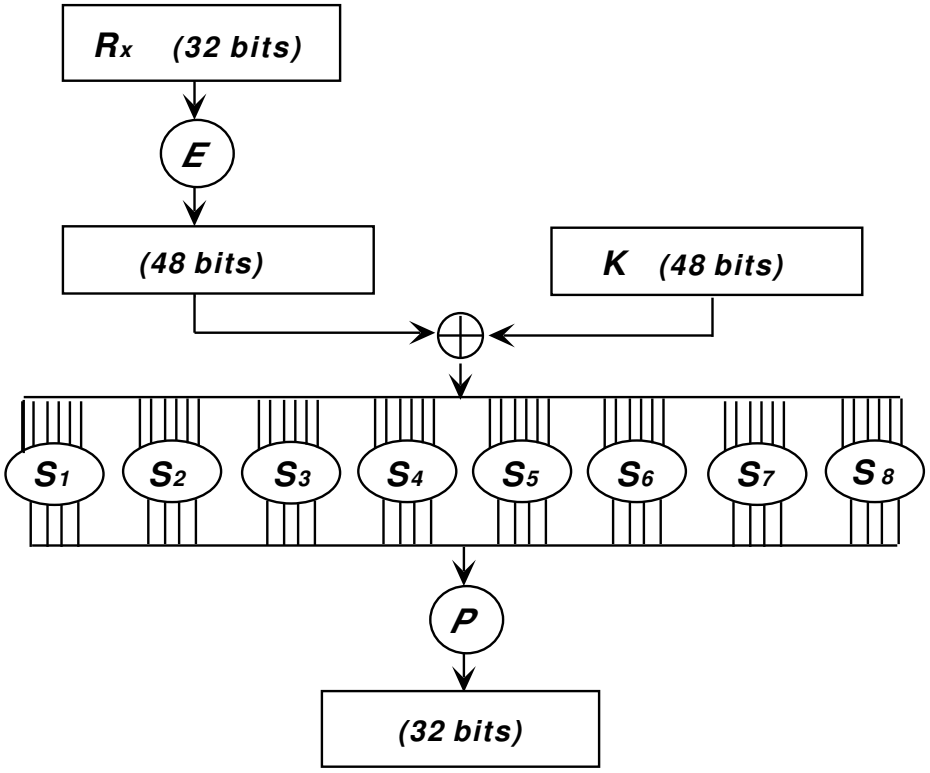
We interpret as follows: The upper half determines how the bits of $C_0$ are chosen, the bottom half determines the bits that are chosen from $D_0$. For 64-bit key $K$, whose bits are numbered 1 through 64, the bits of $C_0$ are bits 57, 49, 41, ... 44, 36 of $K$. The bits of $D_0$ are bits 63, 55, 47, ..., 12, 4 of $K$. $PC2$ is defined by this next table

PC2

| | | | | | |
|---|---|---|---|---|---|
| 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 |

This table is interpreted in the same way as the table defining $IP$.

## 10.3 The DES function $f$




Finally, to complete the description of DES, we describe the function $f$. $f$ is a function which takes two arguments, a 32-bit string which we will denote by $R_x$ and a 48-bit string which is denoted by $K_x$, and produces a

32-bit string. The computation of $f(R_x, K_x)$ is defined as follows:

1. Apply the expansion function $E$ to $R_x$ (the result is a 48-bit string)
2. Compute $B \leftarrow E(R_x) \oplus K_x$
3. Separate $B$ into chunks of 6-bit strings $B = B_1B_2B_3B_4B_5B_6B_7B_8$
4. Apply the *Selection functions* (also called *S-boxes*) $S_i$ to $B_i$, $1 \leq i \leq 8$. Denote $C_i = S_i(B_i)$
5. Permute the 32-bit string $C = C_1C_2C_3C_4C_5C_6C_7C_8$ using the fixed permutation $P$. The result is $f(R_x, K_x) = P(C)$.

The expansion function $E$, which takes a 32-bit string to a 48-bit string, is defined by the following table:

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 |

The permutation function $P$ is defined by this following table:

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 |

Both tables are interpreted as the table defining $IP$.

Each selection function $S_1, S_2, ..., S_8$ takes a 6-bit block and yields a 4-bit block. $S_1$ is defined the following way. The columns are indexed by the number corresponding to the four middle bits (out of six) whereas the rows are indexed by the the number corresponding two extreme bits (out of six):

$S_1$

| $MIDs:$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $EXTs$ | | | | | | | | | | | | | | | | |
| 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

$S_2, S_3, .., S_8$ are defined by the following tables:

$S_2$

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

$S_3$

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

$S_4$

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

$S_5$

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

$S_6$

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

$S_7$

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

$S_8$

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