

Multiple Encryption and Triple DES

Given the potential vulnerability of DES to a **brute-force attack**, there has been considerable interest in finding an alternative.

One approach is to design a completely new algorithm, is to use multiple encryption with DES and multiple keys. We begin by examining the simplest example of this second alternative. We then look at the widely accepted triple DES (3DES) approach.

Multiple encryption is a technique in which an encryption algorithm is used multiple times. In the first instance, plaintext is converted to ciphertext using the encryption algorithm. This ciphertext is then used as input and the algorithm is applied again.

This process may be repeated through any number of stages.

Triple DES makes use of three stages of the DES algorithm, using a total of two or three distinct keys.

A- Double DES

The simplest form of multiple encryption has two encryption stages and two keys (Figure 7.1a). Given a plaintext P and two encryption keys K_1 and K_2 , ciphertext C is generated as:

$$C = E(K_2, E(K_1, P))$$

Decryption requires that the keys be applied in reverse order:

$$P = D(K_1, D(K_2, C))$$

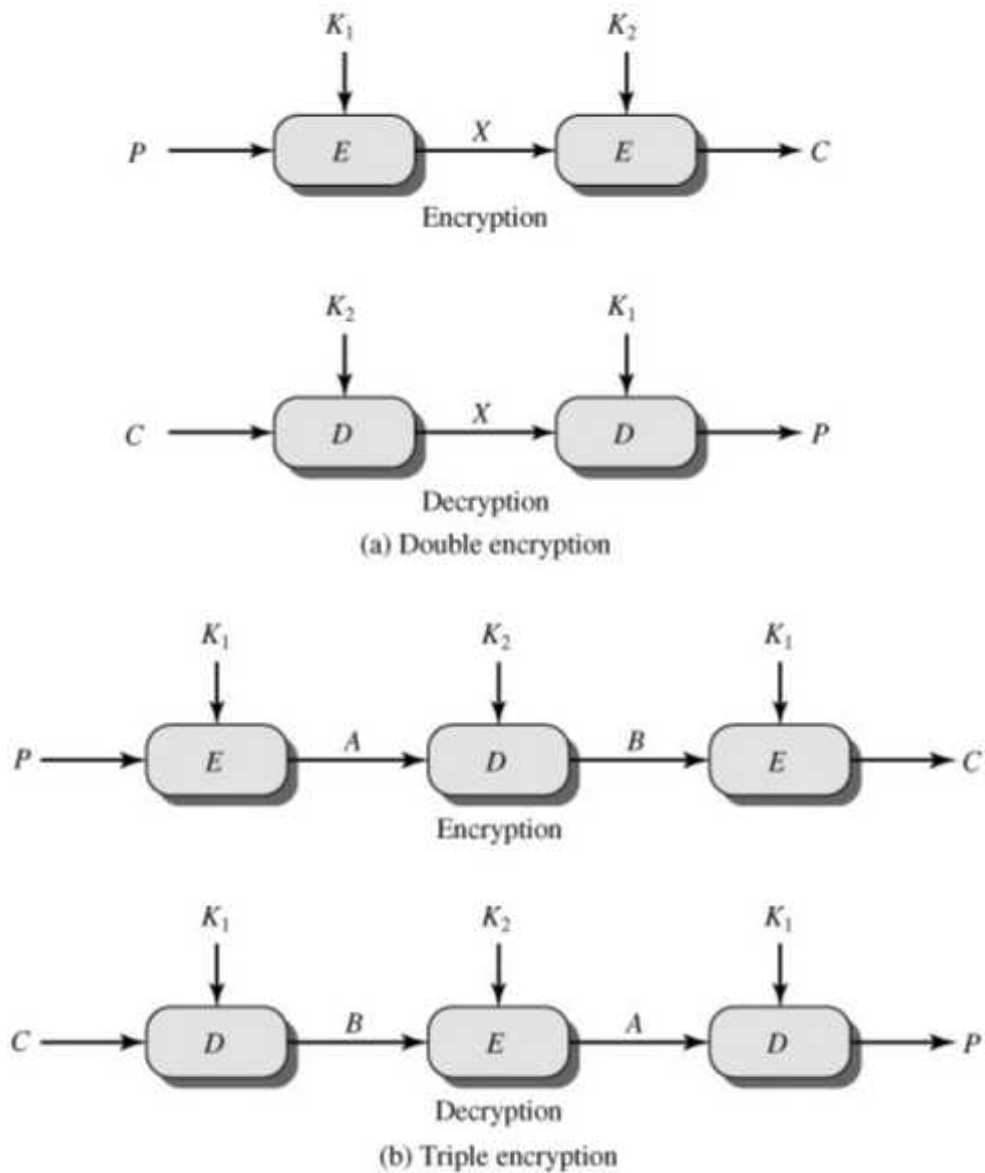


Figure 7.1. Multiple Encryption

For DES, this scheme apparently involves a key length of $56 \times 2 = 112$ bits.

$$E(K_2, E(K_1, P)) = E(K_3, P)$$

If this were the case, then double encryption, and indeed any number of stages of multiple encryption with DES, would be useless because the result would be equivalent to a single encryption with a single 56-bit key. On the face of it, it does not appear that Equation (7.3) is likely to hold.

By Marwa Al-Musawy

Consider that encryption with DES is a mapping of 64-bit blocks to 64-bit blocks. In fact, the mapping can be viewed as a permutation.

That is, if we consider all 2^{64} possible input blocks, DES encryption with a specific key will map each block into a unique 64-bit block. then decryption to recover the original plaintext would be impossible.

With 2^{64} possible inputs, so the number of different mappings that generate a permutation of the input blocks are:

$$\left(2^{64}\right)! = 10^{34738000000000000000} > \left(10^{10^{20}}\right)$$

Therefore, it is reasonable to assume that if DES is used twice with different keys, it will produce one of the many mappings that are not defined by a single application of DES.

Meet-in-the-Middle Attack

Thus, the use of double DES results in a mapping that is not equivalent to a single DES encryption. But there is a way to attack this scheme, one that does not depend on any particular property of DES but that will work against any block encryption cipher.

If we have:

$$C = E(K_2, E(K_1, P))$$

Given a known pair, (P, C) , the attack proceeds as follows :

First : Encrypt P for all 2^{56} possible values of K_1 ,using

$$X_1 = E(K_1, P)$$

Store the value of X_1 in a **table1**.

Second: Decrypt C using all 2^{56} possible values of K_2 , using

$$X_2 = D(K_2, C)$$

Store the value of X_1 in a **table2**.

Third: Check the result against the two tables for a match. If a match occurs

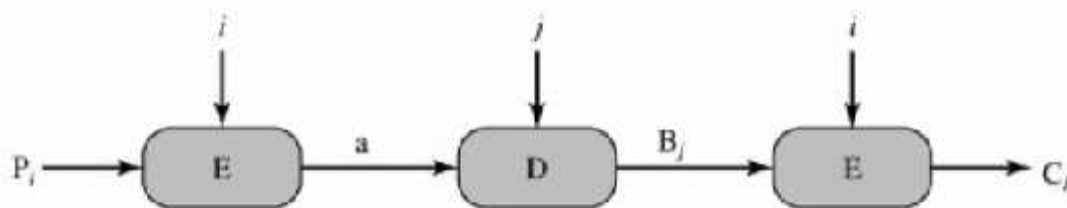
$$X = E(K_1, P) = D(K_2, C) \longrightarrow X_1 = X_2 = X$$

Then test the two resulting keys against a new known plaintext-ciphertext pair. If the two keys produce the correct ciphertext, accept them as the correct keys.

B- Triple DES

An obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys. This raises the cost of the known-plaintext attack to 2^{112} , which is beyond what is practical now and far into the future. However, it has the drawback of requiring a key length of $56 \times 3 = 168$ bits, which may be somewhat unwieldy.

1- Two keys 3DES:



(a) Two-key triple encryption with candidate pair of keys

The function follows an encrypt-decrypt-encrypt (EDE) sequence

$$C = E(K_1, D(K_2, E(K_1, P)))$$

There is no cryptographic significance to the use of decryption for the second stage. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K_1, P)$$

The cost of a brute-force key search on 3DES is on the order of 2^{112} , which is practically very difficult .

Currently, there are no practical cryptanalytic attacks on 3DES.

2- Three keys 3DES:

Three-key 3DES has an effective key length of 168 bits and is defined as follows:

$$C = E(K_3, D(K_2, E(K_1, P)))$$

Backward compatibility with DES is provided by putting $K_3 = K_2$ or $K_1 = K_2$.

3DES has three attractions that assure its widespread use are :

First: With its 168-bit key length, it overcomes the vulnerability to brute-force attack of DES.

Second: The underlying encryption algorithm in 3DES is the same as in DES. This algorithm has been subjected to more scrutiny than any other encryption algorithm over a longer period of time.

Third: No effective cryptanalytic attack based on the algorithm rather than brute force has been found.

The principal drawback of 3DES is that:

First: The algorithm is relatively sluggish in software. 3DES, which has three times as many rounds as DES, is correspondingly slower.

Second: That both DES and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

Because of these drawbacks, 3DES is not a reasonable candidate for long-term use. As a replacement, NIST in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES), which should have a security strength equal to or better than 3DES and significantly improved efficiency.

Advanced Encryption Standard (AES)

AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications. AES does not use a Feistel structure. It uses a 128-bit block size and a key size of 128, 192, or 256 bits.

AES was designed to have the following characteristics:

- Resistance against all known attacks
- Speed and code compactness on a wide range of platforms
- Design simplicity

The AES Cipher

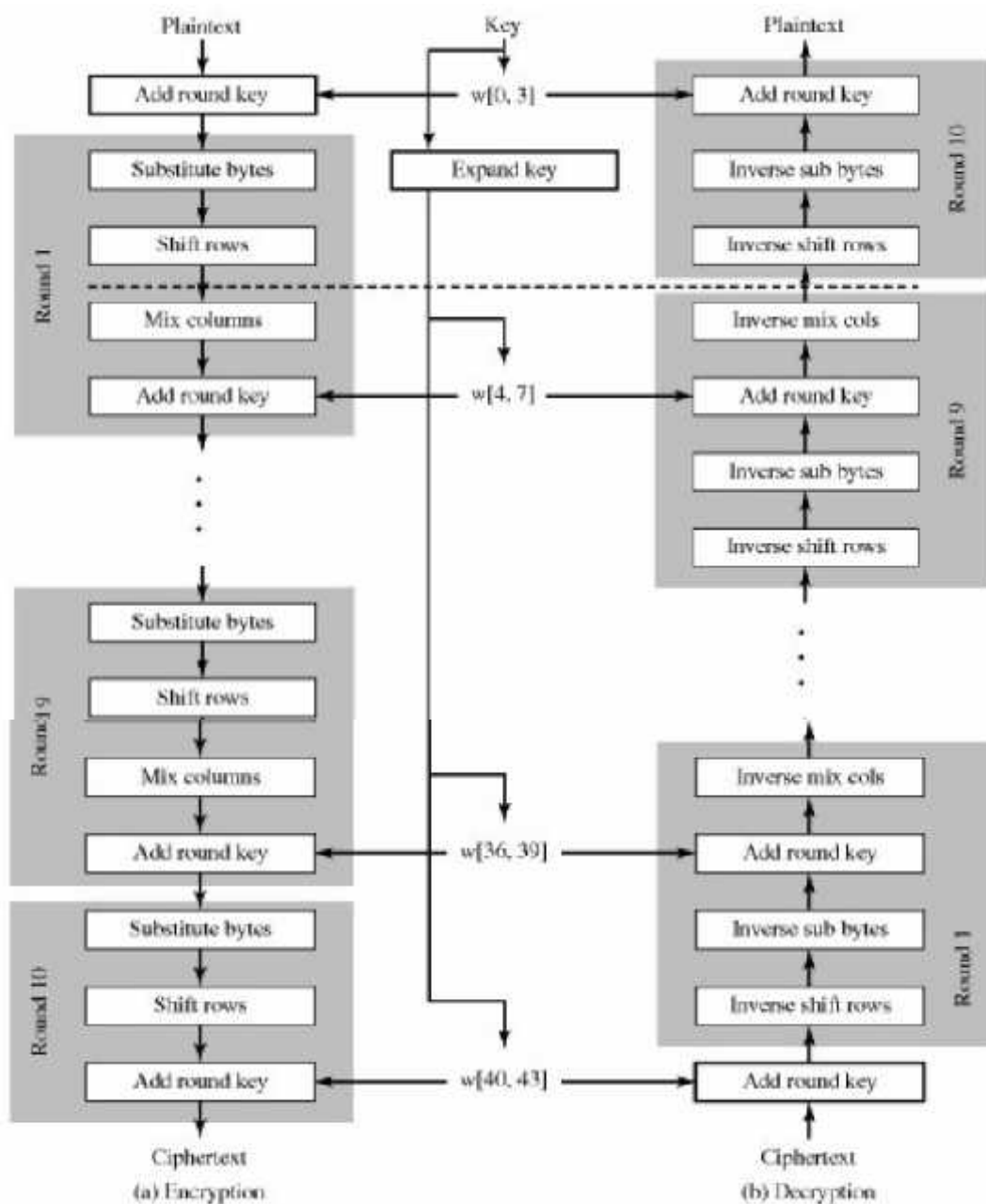
The AES specification uses the same three key size alternatives but limits the block length to 128 bits. A number of AES parameters depend on the key length ([Table 7.1](#)).

Table (7.1) AES parameters

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

Figure (7-1) shows the overall structure of AES.

- The input to the encryption and decryption algorithms is a **single 128-bit block**. this block is depicted as a **square matrix of bytes**. This block is copied into the **State** array, which is modified at each stage of encryption or decryption. After the final stage, **State** is copied to an output matrix.
- The **128-bit key** is depicted as a **square matrix of bytes**. This key is then expanded into an array of key schedule words; each word is **four bytes** and the total key schedule is **44 words** for the 128-bit key.



Figure(7.1)AES Encryption and Decryption

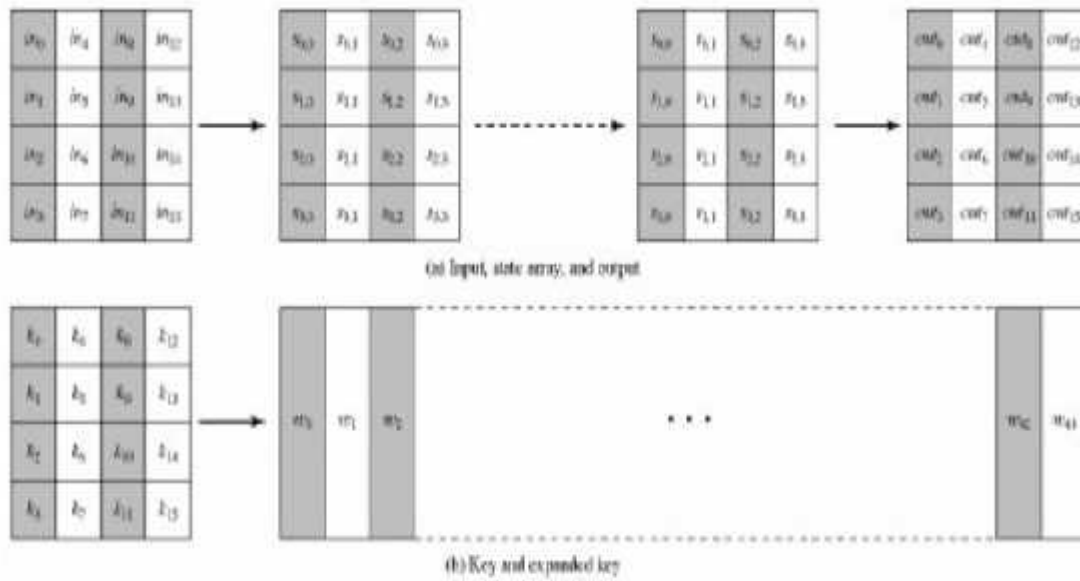


Figure 7.2. AES Data Structures

Before delving into details, we can make several comments about the overall AES structure:

- a- The key that is provided as input is expanded into an array of forty-four 32-bit words, $w[i]$. Four distinct words (128 bits) serve as a round key for each round; these are indicated in Figure 7.1
- b- Four different stages are used, one of permutation and three of substitution:
 - **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block
 - **ShiftRows:** A simple permutation
 - **MixColumns:** A substitution that makes use of arithmetic over $GF(2^8)$
 - **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key.

Substitute Bytes Transformation

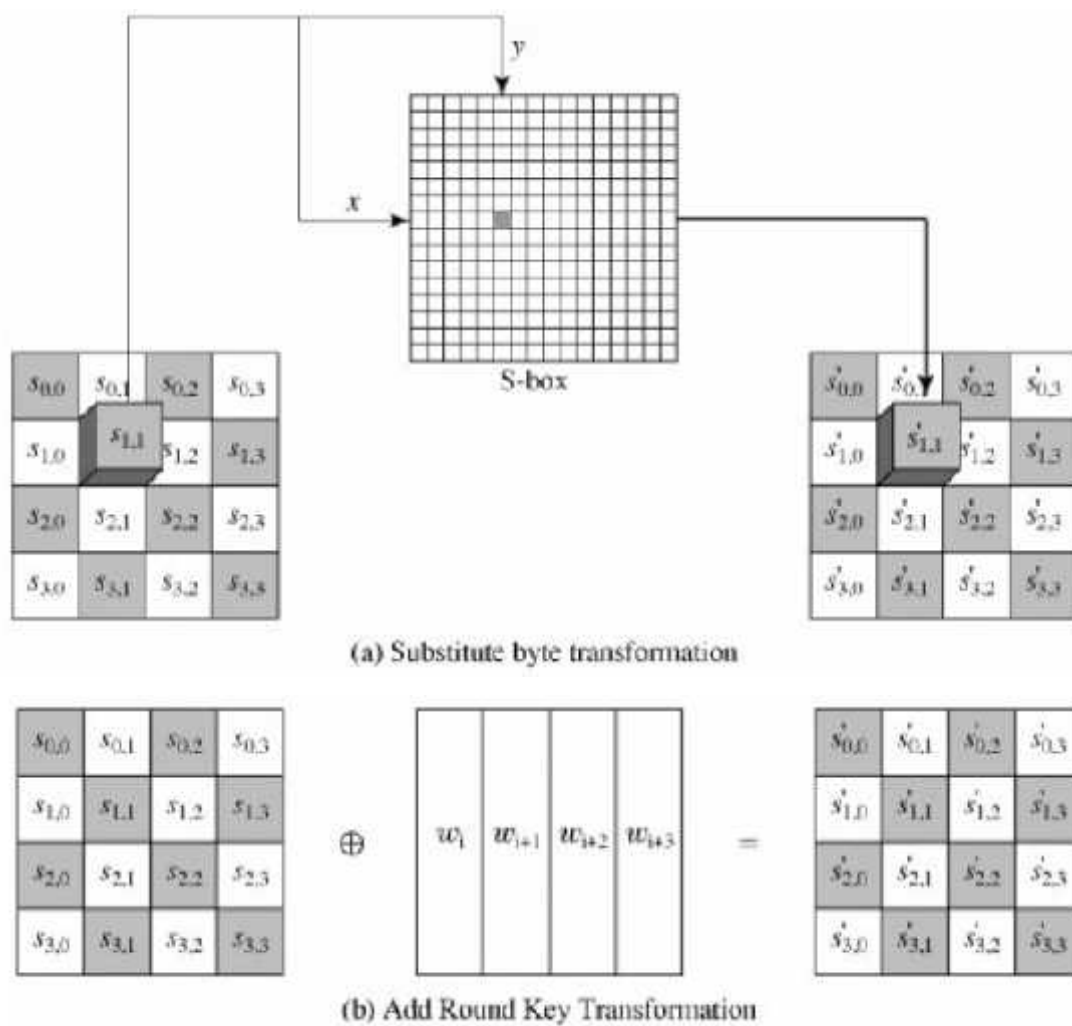
Forward and Inverse Transformations

The **forward substitute byte transformation**, called SubBytes, is a simple table lookup (Figure 7.3a).

AES defines a **16 x 16 matrix of byte values**, called an **S-box** (Table 7.2a), that contains a permutation of all possible 256, 8-bit values. Each individual byte of **State** is mapped into a new byte in the following way:

- The leftmost 4 bits of the byte are used as a row value .
- The rightmost 4 bits are used as a column value.
- These row and column values serve as indexes into the S-box to select a unique 8-bit output value.

For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}. Accordingly, the value {95} is mapped into the value {2A}.



Figure(7-3). AES Byte-Level Operations

Table 8.1. AES S-Boxes

(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	S2	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	F8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(b) Inverse S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D6	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	S3	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Here is an example of the SubBytes transformation:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

 →

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6