APPENDIX D

SIMPLIFIED AES

Simplified AES (S-AES) was developed by Professor Edward Schaefer of Santa Clara University and several of his students [MUSA03]. It is an educational rather than a secure encryption algorithm. It has similar properties and structure to AES with much smaller parameters. The reader might find it useful to work through an example by hand while following the discussion in this appendix. A good grasp of S-AES will make it easier for the student to appreciate the structure and workings of AES.

D.1 OVERVIEW

Figure D.1 illustrates the overall structure of S-AES. The encryption algorithm takes a 16-bit block of plaintext as input and a 16-bit key and produces a 16-bit block of ciphertext as output. The S-AES decryption algorithm takes an 16-bit block of ciphertext and the same 16-bit key used to produce that ciphertext as input and produces the original 16-bit block of plaintext as output.

The encryption algorithm involves the use of four different functions, or transformations: add key (A_K), nibble substitution (NS), shift row (SR), and mix column (MC), whose operation is explained subsequently.

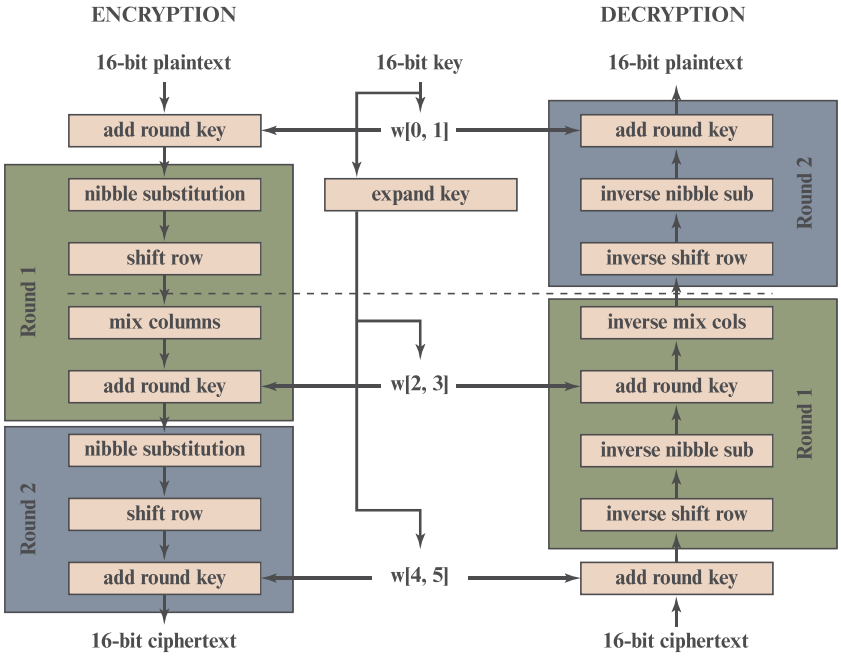


Figure D.1 S-AES Encryption and Decryption

[1]**Definition:** If f and g are two functions, then the function F with the equation $y = F(x) = g[f(x)]$ is called the **composition** of f and g and is denoted as $F = g \circ f$.

We can concisely express the encryption algorithm as a composition[1] of functions:

$$A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$$

so that A_{K_0} is applied first.

The encryption algorithm is organized into three rounds. Round 0 is simply an add key round; round 1 is a full round of four functions; and round 2 contains only three functions. Each round includes the add key function, which makes use of 16 bits of key. The initial 16-bit key is expanded to 48 bits, so that each round uses a distinct 16-bit round key.

Each function operates on a 16-bit state, treated as a 2×2 matrix of nibbles, where one nibble equals 4 bits. The initial value of the **State** matrix is the 16-bit plaintext; **State** is modified by each subsequent function in the encryption process, producing after the last function the 16-bit ciphertext. As Figure D.2a shows, the ordering of nibbles within the matrix is by column. So, for example, the first 8 bits of a 16-bit plaintext input to the encryption cipher occupy the first column of the matrix, and the second 8 bits occupy the second column. The 16-bit key is similarly organized, but it is somewhat more convenient to view the key as two bytes rather than four nibbles (Figure D.2b). The expanded key of 48 bits is treated as three round keys, whose bits are labeled as follows: $K_0 = k_0 \ldots k_{15}$; $K_1 = k_{16} \ldots k_{31}$; and $K_2 = k_{32} \ldots k_{47}$.

Figure D.3 shows the essential elements of a full round of S-AES.

Decryption is also shown in Figure D.1 and is essentially the reverse of encryption:

$$A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ A_{K_2}$$

in which three of the functions have a corresponding inverse function: inverse nibble substitution (INS), inverse shift row (ISR), and inverse mix column (IMC).

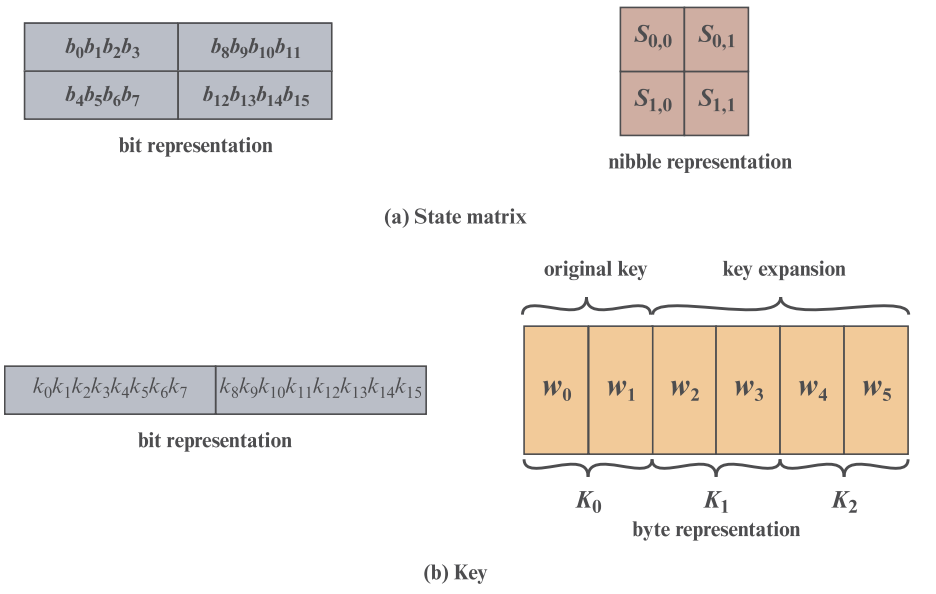


Figure D.2 S-AES Data Structures

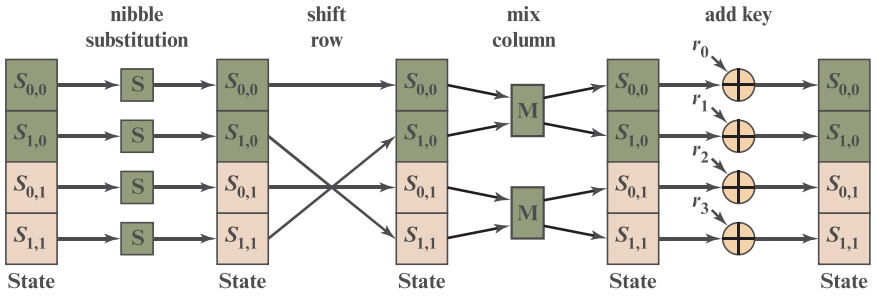


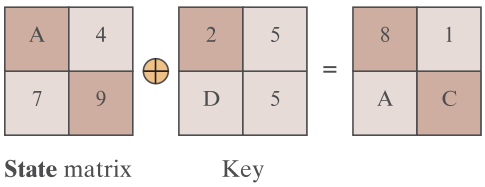
Figure D.3 S-AES Encryption Round

D.2 S-AES ENCRYPTION AND DECRYPTION

We now look at the individual functions that are part of the encryption algorithm.

Add Key

The add key function consists of the bitwise XOR of the 16-bit **State** matrix and the 16-bit round key. Figure D.4 depicts this as a columnwise operation, but it can also be viewed as a nibble-wise or bitwise operation. The following is an example.

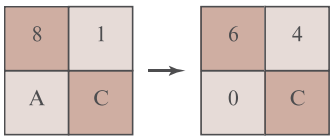


The inverse of the add key function is identical to the add key function, because the XOR operation is its own inverse.

Nibble Substitution

The nibble substitution function is a simple table lookup (Figure D.4). AES defines a 4×4 matrix of nibble values, called an S-box (Table D.1a), that contains a permutation of all possible 4-bit values. Each individual nibble of **State** is mapped into a new nibble in the following way: The leftmost 2 bits of the nibble are used as a row value and the rightmost 2 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 4-bit output value. For example, the hexadecimal value A references row 2, column 2 of the S-box, which contains the value 0. Accordingly, the value A is mapped into the value 0.

Here is an example of the nibble substitution transformation.



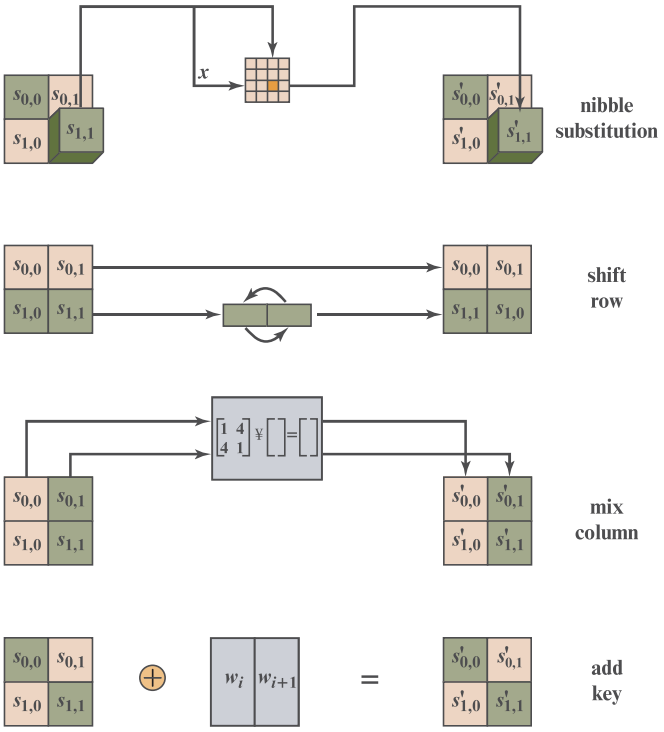


Figure D.4 S-AES Transformations

Table D.1 S-AES S-Boxes

		j			
		00	01	10	11
i	00	9	4	A	B
	01	D	1	8	5
	10	6	2	0	3
	11	C	E	F	7

(a) S-Box

		j			
		00	01	10	11
i	00	A	5	9	B
	01	1	7	8	F
	10	6	0	2	3
	11	C	4	D	E

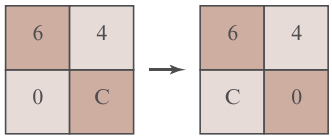
(b) Inverse S-Box

Note: Hexadecimal numbers in shaded boxes; binary numbers in unshaded boxes.

The inverse nibble substitution function makes use of the inverse S-box shown in Table D.1b. Note, for example, that the input 0 produces the output A, and the input A to the S-box produces 0.

Shift Row

The shift row function performs a one-nibble circular shift of the second row of **State**; the first row is not altered (Figure D.4). The following is an example.



The inverse shift row function is identical to the shift row function, because it shifts the second row back to its original position.

Mix Column

The mix column function operates on each column individually. Each nibble of a column is mapped into a new value that is a function of both nibbles in that column. The transformation can be defined by the following matrix multiplication on **State** (Figure D.4):

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

Performing the matrix multiplication, we get

$$\begin{aligned} s'_{0,0} &= s_{0,0} \oplus (4 \bullet s_{1,0}) \\ s'_{1,0} &= (4 \bullet s_{0,0}) \oplus s_{1,0} \\ s'_{0,1} &= s_{0,1} \oplus (4 \bullet s_{1,1}) \\ s'_{1,1} &= (4 \bullet s_{0,1}) \oplus s_{1,1} \end{aligned}$$

Where arithmetic is performed in GF(2^4), and the symbol • refers to multiplication in GF(2^4). Annex D.1 provides the addition and multiplication tables. The following is an example.

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 6 & 4 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 7 & 3 \end{bmatrix}$$

The inverse mix column function is defined as

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

We demonstrate that we have indeed defined the inverse in the following fashion.

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix}$$

The preceding matrix multiplication makes use of the following results in GF(2^4): $9 + (2 \bullet 4) = 9 + 8 = 1$ and $(9 \bullet 4) + 2 = 2 + 2 = 0$. These operations can be verified using the arithmetic tables in Annex D.1 or by polynomial arithmetic.

The mix column function is the most difficult to visualize. Accordingly, we provide an additional perspective on it in Annex D.2.

D.3 KEY EXPANSION

For key expansion, the 16 bits of the initial key are grouped into a row of two 8-bit words. Figure D.5 shows the expansion into six words, by the calculation of four new words from the initial two words. The algorithm is as follows:

$$w_2 = w_0 \oplus g(w_1) = w_0 \oplus \text{RCON}(1) \oplus \text{SubNib}(\text{RotNib}(w_1))$$

$$w_3 = w_2 \oplus w_1$$

$$w_4 = w_2 \oplus g(w_3) = w_2 \oplus \text{RCON}(2) \oplus \text{SubNib}(\text{RotNib}(w_3))$$

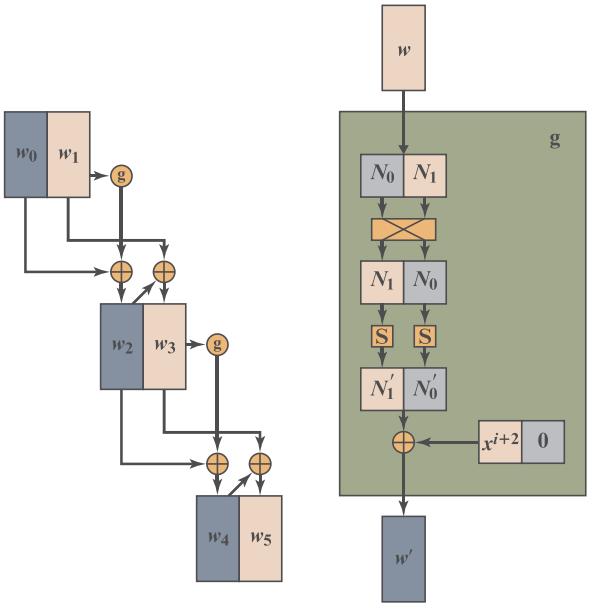
$$w_5 = w_4 \oplus w_3$$

RCON is a round constant, defined as follows: $RC[i] = x^{i+2}$, so that $RC[1] = x^3 = 1000$ and $RC[2] = x^4 \bmod ((x^4 + x + 1) = x + 1 = 0011$. RC[$i$] forms the leftmost nibble of a byte, with the rightmost nibble being all zeros. Thus, RCON(1) = 10000000 and RCON(2) = 00110000.

For example, suppose the key is 2D55 = 0010 1101 0101 0101 = w_0w_1. Then

$$\begin{aligned} w_2 &= 00101101 \oplus 10000000 \oplus \text{SubNib}(01010101) \\ &= 00101101 \oplus 10000000 \oplus 00010001 = 10111100 \end{aligned}$$

$$w_3 = 10111100 \oplus 01010101 = 11101001$$



(a) Overall algorithm (b) Function g

Figure D.5 S-AES Key Expansion

$$w_4 = 10111100 \oplus 00110000 \oplus \text{SubNib}(10011110)$$
$$= 10111100 \oplus 00110000 \oplus 00101111 = 10100011$$

$$w_5 = 10100011 \oplus 11101001 = 01001010$$

D.4 THE S-BOX

The S-box is constructed as follows:

1. Initialize the S-box with the nibble values in ascending sequence row by row. The first row contains the hexadecimal values (0, 1, 2, 3); the second row contains (4, 5, 6, 7); and so on. Thus, the value of the nibble at row i, column j is $4i + j$.
2. Treat each nibble as an element of the finite field GF(2^4) modulo $x^4 + x + 1$. Each nibble $a_0a_1a_2a_3$ represents a polynomial of degree 3.
3. Map each byte in the S-box to its multiplicative inverse in the finite field GF(2^4) modulo $x^4 + x + 1$; the value 0 is mapped to itself.
4. Consider that each byte in the S-box consists of 4 bits labeled (b_0, b_1, b_2, b_3). Apply the following transformation to each bit of each byte in the S-box. The AES standard depicts this transformation in matrix form:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Here prime ($'$) indicates that the variable is to be updated by the value on the right. Remember that addition and multiplication are being calculated modulo 2.

Table D.1a shows the resulting S-box. This is a nonlinear, invertible matrix. The inverse S-box is shown in Table D.1b.

D.5 S-AES STRUCTURE

We can now examine several aspects of interest concerning the structure of AES. First, note that the encryption and decryption algorithms begin and end with the add key function. Any other function, at the beginning or end, is easily reversible without knowledge of the key and so would add no security but just a processing overhead. Thus, there is a round 0 consisting of only the add key function.

The second point to note is that round 2 does not include the mix column function. The explanation for this in fact relates to a third observation, which is that although the decryption algorithm is the reverse of the encryption algorithm, as clearly seen in Figure D.1, it does not follow the same sequence of functions. Thus,

Encryption: $A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$
Decryption: $A_{K_0} \circ INS \circ ISR \circ IMC \circ A_{K_1} \circ INS \circ ISR \circ A_{K_2}$

From an implementation point of view, it would be desirable to have the decryption function follow the same function sequence as encryption. This allows the decryption algorithm to be implemented in the same way as the encryption algorithm, creating opportunities for efficiency.

Note that if we were able to interchange the second and third functions, the fourth and fifth functions, and the sixth and seventh functions in the decryption sequence, we would have the same structure as the encryption algorithm. Let's see if this is possible. First, consider the interchange of INS and ISR. Given a state N consisting of the nibbles (N_0, N_1, N_2, N_3), the transformation INS(ISR(N)) proceeds as

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} N_0 & N_2 \\ N_3 & N_1 \end{pmatrix} \rightarrow \begin{pmatrix} IS[N_0] & IS[N_2] \\ IS[N_3] & IS[N_1] \end{pmatrix}$$

where IS refers to the inverse S-Box. Reversing the operations, the transformation ISR(INS(N) proceeds as

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} IS[N_0] & IS[N_2] \\ IS[N_1] & IS[N_3] \end{pmatrix} \rightarrow \begin{pmatrix} IS[N_0] & IS[N_2] \\ IS[N_3] & IS[N_1] \end{pmatrix}$$

which is the same result. Thus, $INS(ISR(N)) = ISR(INS(N))$.

Now consider the operation of inverse mix column followed by add key: $IMC(A_{K_1}(N))$ where the round key K_1 consists of the nibbles $(k_{0,0}, k_{1,0}, k_{0,1}, k_{1,1})$. Then

$$\begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\left(\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}\right) = \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} \oplus N_0 & k_{0,1} \oplus N_2 \\ k_{1,0} \oplus N_1 & k_{1,1} \oplus N_3 \end{pmatrix}$$

$$= \begin{pmatrix} 9(k_{0,0} \oplus N_0) \oplus 2(k_{1,0} \oplus N_1) & 9(k_{0,1} \oplus N_2) \oplus 2(k_{1,1} \oplus N_3) \\ 2(k_{0,0} \oplus N_0) \oplus 9(k_{1,0} \oplus N_1) & 2(k_{0,1} \oplus N_2) \oplus 9(k_{1,1} \oplus N_3) \end{pmatrix}$$

$$= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) \oplus (9N_0 \oplus 2N_1) & (9k_{0,1} \oplus 2k_{1,1}) \oplus (9N_2 \oplus 2N_3) \\ (2k_{0,0} \oplus 9k_{1,0}) \oplus (2N_0 \oplus 9N_1) & (2k_{0,1} \oplus 9k_{1,1}) \oplus (2N_2 \oplus 9N_3) \end{pmatrix}$$

$$= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) & (9k_{0,1} \oplus 2k_{1,1}) \\ (2k_{0,0} \oplus 9k_{1,0}) & (2k_{0,1} \oplus 9k_{1,1}) \end{pmatrix} \oplus \begin{pmatrix} (9N_0 \oplus 2N_1) & (9N_2 \oplus 2N_3) \\ (2N_0 \oplus 9N_1) & (2N_2 \oplus 9N_3) \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}$$

All of these steps make use of the properties of finite field arithmetic. The result is that $IMC(A_{K_1}(N)) = IMC(K_1) \oplus IMC(N)$. Now let us define the inverse round key for round 1 to be $IMC(K_1)$ and the inverse add key operation IA_{K_1} to

be the bitwise XOR of the inverse round key with the state vector. Then we have $\text{IMC}(\text{A}_{K_1}(N)) = \text{IA}_{K_1}(\text{IMC}(N))$. As a result, we can write the following:

Encryption: $\text{A}_{K_2} \circ \text{SR} \circ \text{NS} \circ \text{A}_{K_1} \circ \text{MC} \circ \text{SR} \circ \text{NS} \circ \text{A}_{K_0}$
Decryption: $\text{A}_{K_0} \circ \text{INS} \circ \text{ISR} \circ \text{IMC} \circ \text{A}_{K_1} \circ \text{INS} \circ \text{ISR} \circ \text{A}_{K_2}$
Decryption: $\text{A}_{K_0} \circ \text{ISR} \circ \text{INS} \circ \text{A}_{\text{IMC}(K_1)} \circ \text{IMC} \circ \text{ISR} \circ \text{INS} \circ \text{A}_{K_2}$

Both encryption and decryption now follow the same sequence. Note that this derivation would not work as effectively if round 2 of the encryption algorithm included the MC function. In that case, we would have

Encryption: $\text{A}_{K_2} \circ \text{MC} \circ \text{SR} \circ \text{NS} \circ \text{A}_{K_1} \circ \text{MC} \circ \text{SR} \circ \text{NS} \circ \text{A}_{K_0}\text{s}$
Decryption: $\text{A}_{K_0} \circ \text{INS} \circ \text{ISR} \circ \text{IMC} \circ \text{A}_{K_1} \circ \text{INS} \circ \text{ISR} \circ \text{IMC} \circ \text{A}_{K_2}$

There is now no way to interchange pairs of operations in the decryption algorithm so as to achieve the same structure as the encryption algorithm.

ANNEX D.1 ARITHMETIC IN GF(2^4)

Table D.2 shows the addition and multiplication tables in GF(2^4) modulo $x^4 + x + 1$. For example, consider the product $(4 \bullet \text{C}) = (0100 \bullet 1100)$. In terms of polynomials, this is the product $[x^2 \times (x^3 + x^2)] \bmod (x^4 + x + 1) = (x^5 + x^4) \bmod (x^4 + x + 1)$. Because the degree of the polynomial to the right of the mod operator is greater than or equal to the modulus, a division is required to determine the remainder:

$$
\begin{array}{r|l}
x^4 + x + 1 & x + 1 \\
\cline{2-2}
 & x^5 + x^4 \\
 & x^5 + \quad x^2 + x \\
\cline{2-2}
 & \quad x^4 + x^2 + x \\
 & \quad x^4 + \quad x + 1 \\
\cline{2-2}
 & \quad x^2 + \quad 1
\end{array}
$$

In binary, the remainder is expressed as 0101, or 5 in hexadecimal. Thus $(4 \bullet C) = 5$, which agrees with the multiplication table in Table D.2.

Table D.2 Arithmetic in GF(2^4) modulo $x^4 + x + 1$

(a) Addition

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B

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

(b) Multiplication

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

ANNEX D.2 THE MIX COLUMN FUNCTION

The mix column function operates on each column individually. Each nibble of a column is mapped into a new value that is a function of both nibbles in that column. The transformation is defined by the following matrix multiplication on **State** (Figure D.4).

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

We can recast this in terms of polynomials as follows. The value 1 corresponds to the polynomial 1 and the value 4 (binary 100) corresponds to the polynomial x^2. Thus, we have

$$\begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

Remember that multiplication is performed modulo $(x^4 + x + 1)$. Using the polynomial formulation allows us to develop a simple explanation of the arithmetic involved. Referring back to the representation of the state matrix in Figure D.2a, we can recast the mix column multiplications as follows:

$$\begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} \begin{bmatrix} b_0x^3 + b_1x^2 + b_2x + b_3 & b_8x^3 + b_9x^2 + b_{10}x + b_{11} \\ b_4x^3 + b_5x^2 + b_6x + b_7 & b_{12}x^3 + b_{13}x^2 + b_{14}x + b_{15} \end{bmatrix}$$

Let's perform the multiplication of the first row of the left-hand matrix with the first column of the right-hand matrix to get the entry in the upper left-hand corner of the target matrix; that is, the polynomial value for $s'_{0,0}$. We have

$$\begin{aligned} s'_{0,0} &= (b_0x^3 + b_1x^2 + b_2x + b_3) + (x^2)\,(b_4x^3 + b_5x^2 + b_6x + b_7) \\ &= b_4x^5 + b_5x^4 + (b_0 \oplus b_6)x^3 + (b_1 \oplus b_7)x^2 + b_2x + b_3 \end{aligned}$$

It can easily be shown that

$$x^5 \bmod (x^4 + x + 1) = (x^2 + x)$$
$$x^4 \bmod (x^4 + x + 1) = (x + 1)$$

The reader is invited to do the polynomial division to demonstrate these equalities. Using these results, we have

$$\begin{aligned} s'_{0,0} &= b_4(x^2 + x) + b_5(x + 1) + (b_0 \oplus b_6)x^3 + (b_1 \oplus b_7)x^2 + b_2x + b_3 \\ &= (b_0 \oplus b_6)x^3 + (b_1 \oplus b_4 \oplus b_7)x^2 + (b_2 \oplus b_4 \oplus b_5)x + (b_3 \oplus b_5) \end{aligned}$$

Expressed in terms of bits, the four bits of $s'_{0,0}$ are

$$s'_{0,0} = [(b_0 \oplus b_6), (b_1 \oplus b_4 \oplus b_7), (b_2 \oplus b_4 \oplus b_5), (b_3 \oplus b_5)]$$

Similarly, we can show that

$$\begin{aligned} s'_{1,0} &= [(b_2 \oplus b_4), (b_0 \oplus b_3 \oplus b_5), (b_0 \oplus b_1 \oplus b_6), (b_1 \oplus b_7)] \\ s'_{0,1} &= [(b_8 \oplus b_{14}), (b_9 \oplus b_{12} \oplus b_{15}), (b_{10} \oplus b_{12} \oplus b_{13}), (b_{11} \oplus b_{13})] \\ s'_{1,1} &= [(b_{10} \oplus b_{12}), (b_8 \oplus b_{11} \oplus b_{13}), (b_8 \oplus b_9 \oplus b_{14}), (b_9 \oplus b_{15})] \end{aligned}$$