is represented by the byte sequence

$$a_0\, a_1\, a_2\, \ldots\, a_{13}\, a_{14}\, a_{15}, \tag{3.4}$$

where

$$\begin{aligned} a_0 &= \{r_0\, r_1\, \ldots\, r_7\}; \\ a_1 &= \{r_8\, r_9\, \ldots\, r_{15}\}; \\ &\vdots \\ a_{15} &= \{r_{120}\, r_{121}\, \ldots\, r_{127}\}. \end{aligned} \tag{3.5}$$

As described in Section 3.2, the bits within any individual byte are indexed in decreasing order from left to right. This ordering is more natural for the finite field arithmetic on bytes that is described in Section 4. The two types of bit indices for byte sequences are illustrated in Table 2.

Table 2. Indices for bytes and bits

Bit index in sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Byte index	0								1								...
Bit index in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

3.4 The State

Internally, the algorithms for the AES block ciphers are performed on a two-dimensional (four-by-four) array of bytes called the *state*. In the state array, denoted by s, each individual byte has two indices: a row index r in the range $0 \le r < 4$ and a column index c in the range $0 \le c < 4$. An individual byte of the state is denoted by either $s_{r,c}$ or $s[r,c]$.

In the specifications for the AES block cipher algorithms in Section 5, the first step is to copy the input array of bytes $in_0,\ in_1,\ \ldots,\ in_{15}$ to the state array s as follows:

$$s[r,c] = in[r+4c] \qquad \text{for } 0 \le r < 4 \text{ and } 0 \le c < 4. \tag{3.6}$$

A sequence of transformations is then applied to the state array, after which its final value is copied to the output array of bytes $out_0,\ out_1,\ \ldots,\ out_{15}$ as follows:

$$out[r+4c] = s[r,c] \qquad \text{for } 0 \le r < 4 \text{ and } 0 \le c < 4. \tag{3.7}$$

The correspondence between the indices of the input and output with the indices of the state array is illustrated in Fig. 1.

The specifications of CIPHER(), KEYEXPANSION(), and INVCIPHER() are given in Sections 5.1, 5.2, and 5.3, respectively.

5.1 CIPHER()

The rounds in the specification of CIPHER() are composed of the following four byte-oriented transformations on the state:

- SUBBYTES() applies a substitution table (S-box) to each byte.
- SHIFTROWS() shifts rows of the state array by different offsets.
- MIXCOLUMNS() mixes the data within each column of the state array.
- ADDROUNDKEY() combines a round key with the state.

The four transformations are specified in Sections 5.1.1–5.1.4. In those specifications, the transformed bit, byte, or block is denoted by appending the symbol $'$ as a superscript on the original variable $\left(\text{i.e., by } b_i', b', s_{i,j}', \text{ or } s'\right)$.

The round keys for ADDROUNDKEY() are generated by KEYEXPANSION(), which is specified in Section 5.2. In particular, the key schedule is represented as an array w of $4 * (Nr + 1)$ words.

CIPHER() is specified in the pseudocode in Alg. 1.

Algorithm 1 Pseudocode for CIPHER()

```
 1: procedure CIPHER(in, Nr, w)
 2:     state ← in                                          ▷ See Sec. 3.4
 3:     state ← ADDROUNDKEY(state, w[0..3])                 ▷ See Sec. 5.1.4
 4:     for round from 1 to Nr − 1 do
 5:         state ← SUBBYTES(state)                         ▷ See Sec. 5.1.1
 6:         state ← SHIFTROWS(state)                        ▷ See Sec. 5.1.2
 7:         state ← MIXCOLUMNS(state)                       ▷ See Sec. 5.1.3
 8:         state ← ADDROUNDKEY(state, w[4 ∗ round..4 ∗ round + 3])
 9:     end for
10:     state ← SUBBYTES(state)
11:     state ← SHIFTROWS(state)
12:     state ← ADDROUNDKEY(state, w[4 ∗ Nr..4 ∗ Nr + 3])
13:     return state                                        ▷ See Sec. 3.4
14: end procedure
```

The first step (Line 2) is to copy the input into the state array using the conventions from Sec. 3.4. After an initial round key addition (Line 3), the state array is transformed by Nr applications of the round function (Lines 4–12); the final round (Lines 10–12) differs in that the MIXCOLUMNS() transformation is omitted. The final state is then returned as the output (Line 13), as described in Section 3.4.

Table 4. SBOX(): substitution values for the byte `xy` (in hexadecimal format)

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	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	52	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
x	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	e8	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

the substitution value would be determined by the intersection of the row with index ‘5’ and the column with index ‘3’ in Table 4, so that $s'_{r,c} = \{\texttt{ed}\}$.

5.1.2 SHIFTROWS()

SHIFTROWS() is a transformation of the state in which the bytes in the last three rows of the state are cyclically shifted. The number of positions by which the bytes are shifted depends on the row index r, as follows:

$$s'_{r,c} = s_{r,(c+r) \bmod 4} \qquad \text{for } 0 \le r < 4 \text{ and } 0 \le c < 4. \tag{5.5}$$

SHIFTROWS() is illustrated in Figure 3. In that representation of the state, the effect is to move each byte by r positions to the left in the row, cycling the left-most r bytes around to the right end of the row. The first row, where $r = 0$, is unchanged.

5.2 KEYEXPANSION()

KEYEXPANSION() is a routine that is applied to the key to generate $4*(Nr+1)$ words. Thus, four words are generated for each of the $Nr+1$ applications of ADDROUNDKEY() within the specification of CIPHER(), as described in Section 5.1.4. The output of the routine consists of a linear array of words, denoted by $w[i]$, where i is in the range $0 \le i < 4*(Nr+1)$.

KEYEXPANSION() invokes 10 fixed words denoted by $Rcon[j]$ for $1 \le j \le 10$. These 10 words are called the *round constants*. For AES-128, a distinct round constant is called in the generation of each of the 10 round keys. For AES-192 and AES-256, the key expansion routine calls the first eight and seven of these same constants, respectively. The values of $Rcon[j]$ are given in hexadecimal notation in Table 5:

Table 5. Round constants

j	$Rcon[j]$	j	$Rcon[j]$
1	`[01,00,00,00]`	6	`[20,00,00,00]`
2	`[02,00,00,00]`	7	`[40,00,00,00]`
3	`[04,00,00,00]`	8	`[80,00,00,00]`
4	`[08,00,00,00]`	9	`[1b,00,00,00]`
5	`[10,00,00,00]`	10	`[36,00,00,00]`

The value of the left-most byte of $Rcon[j]$ in polynomial form is x^{j-1}. Note that for $j > 0$, these bytes may be generated by successively applying XTIMES() to the byte represented by x^{j-1} (see Eq. 4.5).

Two transformations on words are called within KEYEXPANSION(): ROTWORD() and SUBWORD(). Given an input word represented as a sequence $[a_0, a_1, a_2, a_3]$ of four bytes,

$$\textsc{RotWord}([a_0, a_1, a_2, a_3]) = [a_1, a_2, a_3, a_0], \tag{5.10}$$

and

$$\textsc{SubWord}([a_0, \ldots, a_3]) = [\textsc{SBox}(a_0), \textsc{SBox}(a_1), \textsc{SBox}(a_2), \textsc{SBox}(a_3)]. \tag{5.11}$$

The expansion of the key proceeds according to the pseudocode in Alg. 2. The first Nk words of the expanded key are the key itself. Every subsequent word $w[i]$ is generated recursively from the preceding word, $w[i-1]$, and the word Nk positions earlier, $w[i-Nk]$, as follows:

- If i is a multiple of Nk, then $w[i] = w[i-Nk] \oplus \textsc{SubWord}(\textsc{RotWord}(w[i-1])) \oplus Rcon[i/Nk]$.
- For AES-256, if $i+4$ is a multiple of 8, then $w[i] = w[i-Nk] \oplus \textsc{SubWord}(w[i-1])$.
- For all other cases, $w[i] = w[i-Nk] \oplus w[i-1]$.

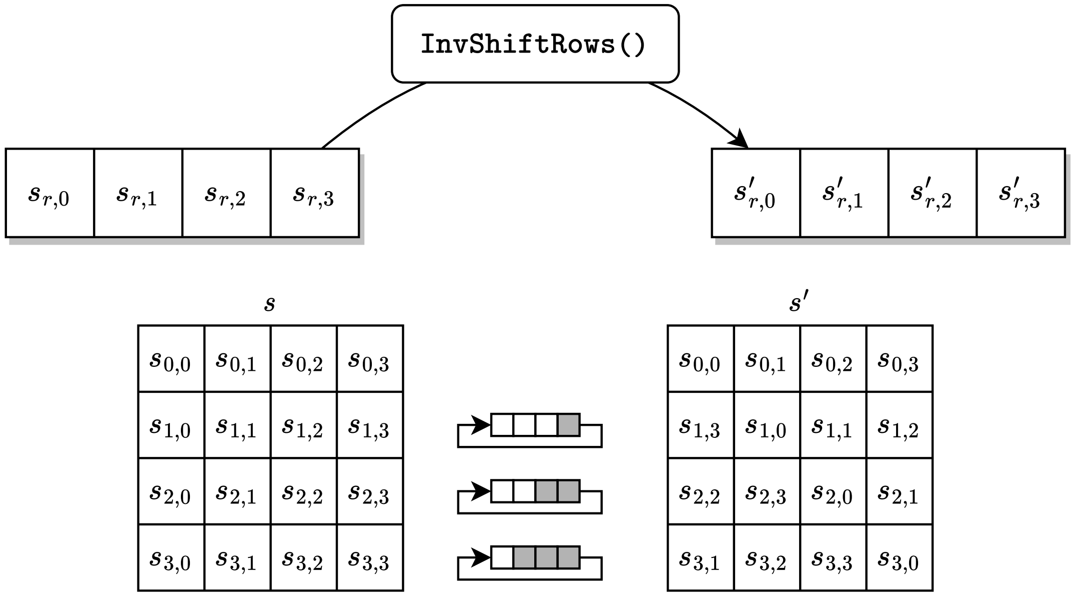

Figure 9. Illustration of INVSHIFTROWS()

5.3.2 INVSUBBYTES()

INVSUBBYTES() is the inverse of SUBBYTES(), in which the inverse of SBOX(), denoted by INVSBOX(), is applied to each byte of the state. INVSBOX() is derived from Table 4 by switching the roles of inputs and outputs, as presented in Table 6:

Table 6. INVSBOX(): substitution values for the byte `xy` (in hexadecimal format)

		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	d8	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	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Appendix B — Cipher Example

The following diagram shows the values in the state array as the cipher progresses for a block length and a key length of 16 bytes each (i.e., $Nb = 4$ and $Nk = 4$).

```
Input   = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34
Key     = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
```

The Round Key values are taken from the Key Expansion example in Appendix A.1.

Round Number	Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key Value
input	32 88 31 e0				2b 28 ab 09
	43 5a 31 37				7e ae f7 cf
	f6 30 98 07				15 d2 15 4f
	a8 8d a2 34				16 a6 88 3c
1	19 a0 9a e9	d4 e0 b8 1e	d4 e0 b8 1e	04 e0 48 28	a0 88 23 2a
	3d f4 c6 f8	27 bf b4 41	bf b4 41 27	66 cb f8 06	fa 54 a3 6c
	e3 e2 8d 48	11 98 5d 52	5d 52 11 98	81 19 d3 26	fe 2c 39 76
	be 2b 2a 08	ae f1 e5 30	30 ae f1 e5	e5 9a 7a 4c	17 b1 39 05
2	a4 68 6b 02	49 45 7f 77	49 45 7f 77	58 1b db 1b	f2 7a 59 73
	9c 9f 5b 6a	de db 39 02	db 39 02 de	4d 4b e7 6b	c2 96 35 59
	7f 35 ea 50	d2 96 87 53	87 53 d2 96	ca 5a ca b0	95 b9 80 f6
	f2 2b 43 49	89 f1 1a 3b	3b 89 f1 1a	f1 ac a8 e5	f2 43 7a 7f
3	aa 61 82 68	ac ef 13 45	ac ef 13 45	75 20 53 bb	3d 47 1e 6d
	8f dd d2 32	73 c1 b5 23	c1 b5 23 73	ec 0b c0 25	80 16 23 7a
	5f e3 4a 46	cf 11 d6 5a	d6 5a cf 11	09 63 cf d0	47 fe 7e 88
	03 ef d2 9a	7b df b5 b8	b8 7b df b5	93 33 7c dc	7d 3e 44 3b
4	48 67 4d d6	52 85 e3 f6	52 85 e3 f6	0f 60 6f 5e	ef a8 b6 db
	6c 1d e3 5f	50 a4 11 cf	a4 11 cf 50	d6 31 c0 b3	44 52 71 0b
	4e 9d b1 58	2f 5e c8 6a	c8 6a 2f 5e	da 38 10 13	a5 5b 25 ad
	ee 0d 38 e7	28 d7 07 94	94 28 d7 07	a9 bf 6b 01	41 7f 3b 00
5	e0 c8 d9 85	e1 e8 35 97	e1 e8 35 97	25 bd b6 4c	d4 7c ca 11
	92 63 b1 b8	4f fb c8 6c	fb c8 6c 4f	d1 11 3a 4c	d1 83 f2 f9
	7f 63 35 be	d2 fb 96 ae	96 ae d2 fb	a9 d1 33 c0	c6 9d b8 15
	e8 c0 50 01	9b ba 53 7c	7c 9b ba 53	ad 68 8e b0	f8 87 bc bc