1

The Advanced Encryption Standard (AES)

2 The Basic Algorithm

For simplicity, we restrict to 128 bits, and firstly give a brief outline of the algorithm. The algorithm consists of 10 rounds. Each round has a round key, derived from the original key. There is also a 0th round key using the original of 128 bits. A round starts with an input of 128 bits and produces an output of 128 bits.

There a four basic step, called layers, that are used to form the rounds:

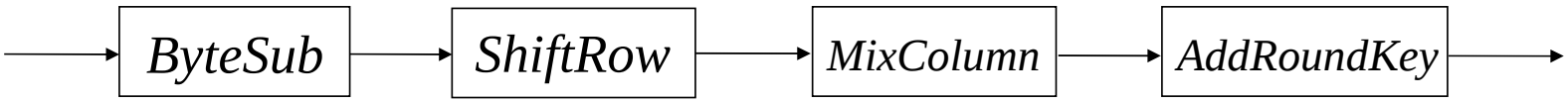
(1) The ByteSub (**SB**) Transformation: This non-linear layer is for resistance to differential and linear cryptanalysis attacks.

(2) The ShiftRow (**SR**) Transformation: This linear mixing step causes diffusion of the bits over multiple rounds.

(3) The MixColumn (**MC**) Transformation: This layer has a purpose similar to ShiftRow.

(4) AddRoundKey (**ARK**) Transformation: The round key is XORed with the result of the above layer.

A round is then



Rijndael Encryption
(1) **ARK**, using the 0th round key. (2) Nine rounds of **BS**, **SR**, **MC**, **ARK**, using round keys 1 to 9. (3) A final round: **BS**, **SR**, **ARK**, using the 10th round key.

The final round omits Mixcolumn layer.

3 The Layers

The 128 input bits are grouped into 16 bytes of 8 bits each, call them

$$a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, a_{0,1}, a_{1,1}, \cdots, a_{3,3},$$

and are arranged int 4×4 matrix

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}.$$

In the following, we'll need to work with the finite field $GF(2^8)$. The model of $GF(2^8)$ depends on a choice of irreducible polynomial of degree 8. The choice for Rijndeal is $X^8 + X^4 + X^3 + X + 1$. The elements of $GF(2^8)$ can be represented by bytes. They can added by XOR. They also be multiplied in a certain way. Each element has a multiplicative inverse.

3.1 The ByteSub Transformation

S – Box (16×16)

99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
208	239	170	251	67	77	51	133	69	249	2	127	80	69	159	168
81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
205	12	19	236	95	151	68	23	196	167	126	**61**	100	93	25	115
96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
186	120	37	46	28	166	180	198	232	221	116	**31**	75	189	139	138
112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

3.1 The ByteSub Transformation (Continued)

Wirte a byte as 8 bits : *abcdefgh*. Look for the entry in the *abcd* row and *efgh* column. For example, if the input byte is 10001011, we look in row 9 and column 12. The entry is 61, which is 111101 in binary. The output of ByteSub is again a 4×4 matrix of bytes.

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}.$$

3.2 The ShiftRow Transformation

The four rows of the matrix are shifted cyclically to the left by offsets of 0,1,2, and 3, to obtain

$$\begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{bmatrix}.$$

3.3 The MixColumn Transformation

The output of the ShiftRow step is a 4×4 matrix $(c_{i,j})$ with entries in $GF(2^8)$. Multiply this by a matrix, again with entries in $GF(2^8)$, to produce the output $(d_{i,j})$, as follows:

$$\begin{bmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{bmatrix} \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix}.$$

3.4 The RoundKey Addition

The round key, derived from the original key consists of 128 bits, which are arranged in a 4×4 matrix $(k_{i,j})$ consisting of bytes. This is XORed with the output $(d_{i,j})$ in the MixColumn step :

$$\begin{bmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix} e_{0,0} & e_{0,1} & e_{0,2} & e_{0,3} \\ e_{1,0} & e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,0} & e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,0} & e_{3,1} & e_{3,2} & e_{3,3} \end{bmatrix}.$$

3.5 The Key Schedule

The original key consists of 128 bits, which are generated into a 4×4 matrix of bytes. Label the first four colums $W(0), W(1), W(2), W(3)$. The new columns are generated recursively. If $4 \nmid i$, then $W(i) = W(i-4) \oplus W(i-1)$. If $4 \mid i$, then $W(i) = W(i-4) \oplus T(W(i-1))$, where $T(W(i-1))$ is the transformation of $W(i-1)$.

$$\text{Let } W(i-1) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}. \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \rightarrow \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix} \xrightarrow{S-box} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix} \rightarrow \begin{bmatrix} e \oplus (10)^{(i-4)/4} \\ f \\ g \\ h \end{bmatrix}$$

$$= T(W(i-1)).$$

The round key for the ith round consists of the columns $W(4i), W(4i+1), W(4i+2), W(4i+3)$.

3.6 The Construction of the S-Box

The S-box has a simple mathematical description. The inverse of the byte $x_7x_6x_5x_4x_3x_2x_1x_0$ in $GF(2^8)$ can be represented by $y_7y_6y_5y_4y_3y_2y_1y_0$. Suppose the inverse of the byte 00000000 is 00000000. The entry of $x_7x_6x_5x_4x_3x_2x_1x_0$ in the S-box can be compute by

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix}.$$

3.6 The Construction of the S-Box (Continued)

Example 3 The inverse of the byte 11001011 in $GF(2^8)$ is 00000100.

We calculate

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

This yield the byte $00011111 = 31$. We check the row $1100 + 1 = 13$ and the column $1011 + 1 = 12$ in the S-box. We also obtian the entry 31.

4 Decryption

Each of the steps ByteSub, ShiftRow, MixColumn, and AddRoundKey is invertible:

(1) The inverse of ByteSub is another lookup table, called InvByteSub (**IBS**).

(2) The inverse of ShiftRow is obtained by shifting the rows to the right instead of to the left, yielding InvShiftRow (**ISR**).

(3) The transformation InvMixColumn (IMC) is given by multiplication by the matrix

$$\begin{bmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{bmatrix}.$$

(4) AddRoundKey is its own inverse.

Therefore,

Rijndael encryption		Rijndael decryption
ARK		ARK, ISR, IBS
BS, SR, MC, ARK		ARK, IMC, ISR, IBS
…	$\Rightarrow$	…
BS, SR, MC, ARK		ARK, IMC, ISR, IBS
BS, SR, ARK.		ARK.

We can rewrite the decryption to achieve the same structure as encryption. Clearly, the oder of ISR and IBS can be reversed. Applying MC and then ARK to a matrix $(c_{i,j})$ is gave as

$(c_{i,j}) \to (m_{i,j})(c_{i,j}) \to (e_{i,j}) = (m_{i,j})(c_{i,j}) \oplus (k_{i,j})$.

The inverse is obtained by solving $(e_{i,j}) = (m_{i,j})(c_{i,j}) \oplus (k_{i,j})$. Since $(c_{i,j}) = (m_{i,j})^{-1}$ $((e_{i,j}) \oplus (k_{i,j})) = (m_{i,j})^{-1}(e_{i,j}) \oplus (m_{i,j})^{-1}(k_{i,j})$, the process is

$(e_{i,j}) \to (m_{i,j})^{-1}(e_{i,j}) \to (m_{i,j})^{-1}(e_{i,j}) \oplus (k'_{i,j})$,

where $(k'_{i,j}) = (m_{i,j})^{-1}(k_{i,j})$. The first arrow is IMC. Let InvAddRoundKey(IARK) be XORing with $(k'_{i,j})$. We can use "IMC and IARK" to replace "ARK and IMC".

Now, the decryption is given by

Rijndael decryption

ARK, IBS, ISR

IMC, IARK, IBS, ISR

…

IMC, IARK, IBS, ISR

ARK.

Rijndael Decryption
(1) ARK, using the 10th round key. (2) Nine rounds of IBS, ISR, IMC, IARK, using round keys 9 to 1. (3) A final round: IBS, ISR, ARK, using the 0th round key.

To keep the perfect structure, the MC is omitted in the last round of the encryption.

5 Design Consideration

(1) Unlike the Feistel system, all bits are treat uniformly. This has effect of diffusing the input bits faster. It can be shown that two rounds are sufficient to obtain full diffusion.

(2) The S-box is constructed in an explicit and simple algebraic way so as to avoid the mysteries of trapdoors built into the algorithm. It is excellent at resisting differential and linear cryptanalysis, as well as interpolation attacks.

(3) The SR step is added to resist truncated differentials and square attack.

(4) The MC causes diffusion among the bytes.

(5) The ARK involves nonlinear mixing of the key bits. The mixing is designed to resist the known part key attack. The round constants are used to eliminate symmetries.

(6) The number of rounds was chosen to be 10 because there are attacks that are better than brute force up to seven rounds in 2004. No known attack beats brute force for seven or more rounds. It was felt that three extra rounds provide a large enough margin of safety.