WIKIPEDIA
The Free Encyclopedia

Rijndael S-box

The **Rijndael S-box** is a substitution box (lookup table) used in the Rijndael cipher, on which the Advanced Encryption Standard (AES) cryptographic algorithm is based.[1]

Forward S-box

The S-box maps an 8-bit input, c, to an 8-bit output, $s = S(c)$. Both the input and output are interpreted as polynomials over GF(2). First, the input is mapped to its multiplicative inverse in

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value $9a_{16}$ is converted into $b8_{16}$.

$GF(2^8) = GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)$, Rijndael's finite field. Zero, as the identity, is mapped to itself. This transformation is known as the *Nyberg S-box* after its inventor Kaisa Nyberg.[2] The multiplicative inverse is then transformed using the following affine transformation:

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

where $[s_7, ..., s_0]$ is the S-box output and $[b_7, ..., b_0]$ is the multiplicative inverse as a vector.

This affine transformation is the sum of multiple rotations of the byte as a vector, where addition is the XOR operation:

$$s = b \oplus (b \lll 1) \oplus (b \lll 2) \oplus (b \lll 3) \oplus (b \lll 4) \oplus 63_{16}$$

where b represents the multiplicative inverse, $\oplus$ is the bitwise XOR operator, $\lll$ is a left bitwise circular shift, and the constant $63_{16} = 01100011_2$ is given in hexadecimal.

An equivalent formulation of the affine transformation is

$$s_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

where s, b, and c are 8 bit arrays, c is 01100011_2, and subscripts indicate a reference to the indexed bit.[3]

Another equivalent is:

$$s = (b \times 31_{10} \mod 257_{10}) \oplus 99_{10}$$[4][5]

where $\times$ is polynomial multiplication of b and 31_{10} taken as bit arrays.

Inverse S-box

The inverse S-box is simply the S-box run in reverse. For example, the inverse S-box of $b8_{16}$ is $9a_{16}$. It is calculated by first calculating the inverse affine transformation of the input value, followed by the multiplicative inverse. The inverse affine transformation is as follows:

Inverse S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The inverse affine transformation also represents the sum of multiple rotations of the byte as a vector, where addition is the XOR operation:

$$b = (s \lll 1) \oplus (s \lll 3) \oplus (s \lll 6) \oplus 5_{16}$$

where $\oplus$ is the bitwise XOR operator, $\lll$ is a left bitwise circular shift, and the constant $5_{16} = 00000101_2$ is given in hexadecimal.

Design criteria

The Rijndael S-box was specifically designed to be resistant to linear and differential cryptanalysis. This was done by minimizing the correlation between linear transformations of input/output bits, and at the same time minimizing the difference propagation probability.

The Rijndael S-box can be replaced in the Rijndael cipher,[1] which defeats the suspicion of a backdoor built into the cipher that exploits a static S-box. The authors claim that the Rijndael cipher structure is likely to provide enough resistance against differential and linear cryptanalysis even if an S-box with "average" correlation / difference propagation properties is used (cf. the "optimal" properties of the Rijndael S-box).

Example implementation in C language

The following C code calculates the S-box:

```c
#include <stdint.h>

#define ROTL8(x,shift) ((uint8_t) ((x) << (shift)) | ((x) >> (8 - (shift))))

void initialize_aes_sbox(uint8_t sbox[256]) {
    uint8_t p = 1, q = 1;

    /* loop invariant: p * q == 1 in the Galois field */
    do {
        /* multiply p by 3 */
        p = p ^ (p << 1) ^ (p & 0x80 ? 0x1B : 0);

        /* divide q by 3 (equals multiplication by 0xf6) */
        q ^= q << 1;
        q ^= q << 2;
        q ^= q << 4;
        q ^= q & 0x80 ? 0x09 : 0;

        /* compute the affine transformation */
        uint8_t xformed = q ^ ROTL8(q, 1) ^ ROTL8(q, 2) ^ ROTL8(q, 3) ^ ROTL8(q, 4);

        sbox[p] = xformed ^ 0x63;
    } while (p != 1);

    /* 0 is a special case since it has no inverse */
```

```
    sbox[0] = 0x63;
}
```

References

1. "The Rijndael Block Cipher" (http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf#page=1) (PDF). Retrieved 2013-11-11.
2. Nyberg K. (1991) Perfect nonlinear S-boxes (https://link.springer.com/chapter/10.1007%2F3-540-46416-6_32). In: Davies D.W. (eds) Advances in Cryptology – EUROCRYPT '91. EUROCRYPT 1991. Lecture Notes in Computer Science, vol 547. Springer, Berlin, Heidelberg
3. "The Advanced Encryption Standard" (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf) (PDF). *FIPS PUB 197: the official AES standard*. Federal Information Processing Standard. 2001-11-26. Retrieved 2010-04-29.
4. Jörg J. Buchholz (2001-12-19). "Matlab implementation of the Advanced Encryption Standard" (http://buchholz.hs-bremen.de/aes/AES.pdf) (PDF).
5. Jie Cui; Liusheng Huang; Hong Zhong; Chinchen Chang; Wei Yang (May 2011). "An Improved AES S-box and Its Performance Analysis" (http://www.ijicic.org/ijicic-10-01041.pdf) (PDF).