Description of SM4 Block Cipher Algorithm

Alongside with Test Vectors Based on Python

Zhuocun Zhai

Department of Computer Science, University College London, UK

(Dated: March 17, 2020)

I. Overall Structure

SM4 is a block cipher algorithm. Its block length and cipher key length are both of 128 bits. SM4 has 32 rounds in total and within every round function an unbalanced Feistel structure is adopted. Also, a 32-bit round key is required for every iteration and all the 32 round keys are generated by a key expansion algorithm using the 128-bit cipher key as the input. Finally, after the 32 rounds, it adopts a reverse function to get the ciphertext.

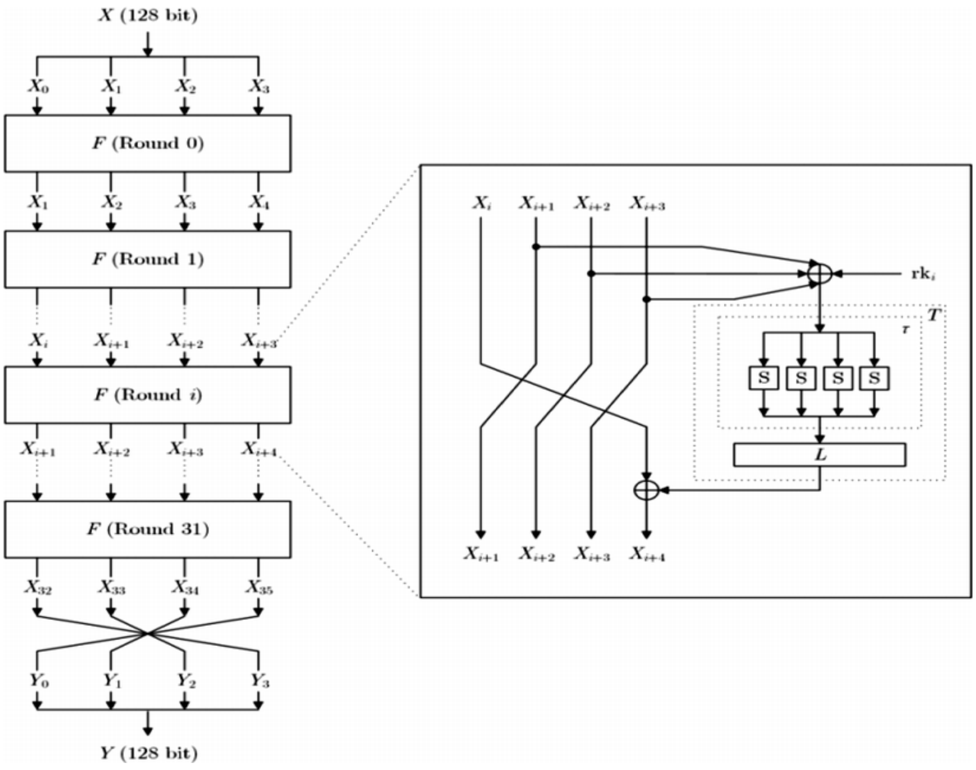

Figure 1: Encryption procedure of SM4

II. Round Function

The Round Function *F* splits the 128-bit block into 4 parts X_i, X_{i+1}, X_{i+2}, X_{i+3} (every sub-block is of 32-bit length) and then takes them alongside with the 32-bit round key rk_i as inputs. It outputs a 32-bit sub-block X_{i+4} and combines X_{i+1}, X_{i+2}, X_{i+3}, X_{i+4} together as the new 128-bit block.

The Round Function *F* has three steps:

a. XOR X_{i+1}, X_{i+2}, X_{i+3} and rk_i to get a 32-bit result.

b. Split the 32-bit result into 4 parts and put them into 4 S-boxes. The 4 S-boxes are totally the same while every S-box's input and output are both 8 bits. Details of the S-boxes are provided later in the section of S-boxes.

c. A linear transformation *L* defined as follows:
$L\ (B) = B \oplus (B<<<2) \oplus (B<<<10) \oplus (B<<<18) \oplus (B<<<24)$
where *B* is the combination of the 4 S-boxes' outputs.

d. XOR X_i and the result of step 3 - *L(B)* to get the final output X_{i+4}.

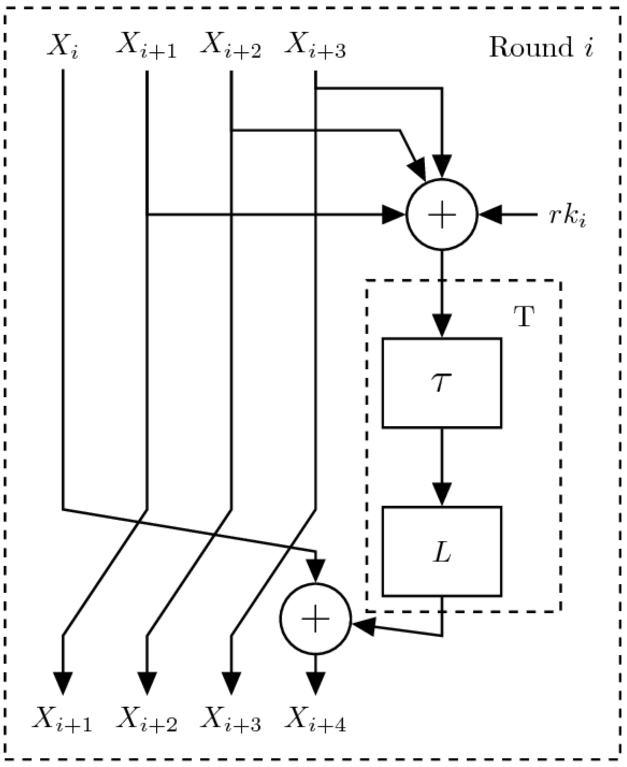

Figure 2: Round Function of SM4

III. S-boxes

The S-box is as follows:

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

Every number is hexadecimal.
For example, when the input is ‘ef’, then the output is the value in row e and column f, i.e. S-box(ef) = 84

IV. Key Schedule

A total number of 32 round keys (rk_0, rk_1, ... , rk_{31}) are required. A key expansion algorithm is adopted to generate them from the initial 128-bit encryption key. At first, it modifies the encryption key by splitting it into 4 32-bit parts and XOR every part with a parameter. The 4 parameters are defined as follows:

$$FK_0 = (A3B1BAC6), FK_2 = (56AA3350),$$

$$FK_3 = (677D9197), FK_4 = (B27022DC).$$

The key expansion algorithm is highly similar to SM4 itself. Taking the aforementioned 4 parts as inputs, it contains 32 rounds and within every round a round key is output. The only differences are as follows:

a. It substitutes round keys with 32 parameters, which are defined as follows:

00070E15,	1C232A31,	383F464D,	545B6269,
70777E85,	8C939AA1,	A8AFB6BD,	C4CBD2D9,
E0E7EEF5,	FC030A11,	181F262D,	343B4249,
50575E65,	6C737A81,	888F969D,	A4ABB2B9,
C0C7CED5,	DCE3EAF1,	F8FF060D,	141B2229,
30373E45,	4C535A61,	686F767D,	848B9299,
A0A7AEB5,	BCC3CAD1,	D8DFE6ED,	F4FB0209,
10171E25,	2C333A41,	484F565D,	646B7279.

b. The linear transformation L is modified to the following form:

$$L\,(B) = B \oplus (B <<< 13) \oplus (B <<< 23)$$

V. Decryption

The structure of decryption is the same as the encryption, but the decryption round keys are in the reverse order of the encryption round keys.

VI. Test Vectors

Based on the description of SM4 block cipher algorithm, I reimplement it in a Python environment and adopt several examples to verify the encryption and decryption procedure. One of the examples is chosen from the official document of SM4.

Input plaintext: 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10
Cipher key: 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10
Output ciphertext: 68 1E DF 34 D2 06 96 5E 86 B3 E9 4F 53 6E 42 46

Input plaintext (Binary form):
00000001001000110100010101100111100010011010101111001101111011111111
1110110111001011101010011000011101100101010000110010000010000

Cipher key (Binary form):
00000001001000110100010101100111100010011010101111001101111011111111
111011011100101110101001100001110110010101000011001000010000
Output ciphertext (Binary form):
01101000000111101101111100110100110100100000011010010110010111101000
011010110011111010010100111101010011011011100100001001000110

The whole details of the test vectors including all the variables described in the 3 files and intermediate values (in both bin and hex form) will be provided in another .csv file. Also, the Python worksheet used to generate them will also be provided.

VII. Reference

[1] SM4 Block Cipher Algorithm, Chinese Government,
http://sca.hainan.gov.cn/xxgk/bzhgf/201804/W020180409400793061524.pdf
[2] https://en.wikipedia.org/wiki/SM4_(cipher)