Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 13, 2018

R. Tse
Ribose
W. Wong
Hang Seng Management College

September 9, 2017

The SM4 Block Cipher Algorithm And Its Modes Of Operations
draft-crypto-sm4-00

Abstract

This document describes the SM4 symmetric blockcipher algorithm published as GB/T 32907-2016 by the Organization of State Commercial Administration of China (OSCCA).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 13, 2018.

Copyright Notice

Table of Contents

1. Introduction

SM4 [GBT.32907-2016] is a cryptographic standard issued by the Organization of State Commercial Administration of China [OSCCA] as authorized cryptographic algorithms for the use within China. The algorithm is applicable published in public.

SM4 is a symmetric encryption algorithm, specifically a blockcipher, designed for data encryption.

This document does not aim to introduce a new algorithm, but to provide a clear and open description of the SM4 algorithm in English, and also to serve as a stable reference for IETF documents that utilize this algorithm.

While this document is similar to [SM4-En] in nature, [SM4-En] is a textual translation of "SMS4" [SM4] published in 2006, and this document follows the updated text and structure of [GBT.32907-2016]. The sections 1 to 7 of this document are intentionally mapped to the corresponding sections 1 to 7 of the [GBT.32907-2016] standard for convenience of the reader.

1.1. History

The "SMS4" algorithm (the former name of SM4) was invented by Shu-Wang Lu [LSW-Bio], first published in 2003 as part of [GB.15629.11-2003], then published independently in 2006 [SM4] by OSCCA, officially renamed to "SM4" in 2012 in [GMT-0002-2012] published by OSCCA, and finally standardized in 2016 as a Chinese National Standard (GB Standard) [GBT.32907-2016].

SMS4 was originally created for use in protecting wireless networks [SM4], and is mandated in the Chinese National Standard for Wireless LAN WAPI (Wired Authentication and Privacy Infrastructure) [GB.15629.11-2003]. A proposal was made to adopt SMS4 into the IEEE 802.11i standard, but the algorithm was eventually not included due to concerns of introducing inoperability with existing ciphers.

The latest SM4 standard [GBT.32907-2016] was proposed by OSCCA, standardized through TC 260 of the Standardization Administration of the People's Republic of China (SAC), and was drafted by the following individuals at the Data Assurance and Communication Security Research Center (DAS Center) of the Chinese Academy of

Sciences, the China Commercial Cryptography Testing Center and the Beijing Academy of Information Science & Technology (BAIST):

- Shu-Wang Lu
- Dai-Wai Li
- Kai-Yong Deng
- Chao Zhang
- Peng Luo
- Zhong Zhang
- Fang Dong
- Ying-Ying Mao
- Zhen-Hua Liu

1.2. Applications

SM4 (and SMS4) has prevalent hardware implementations [SM4-FPGA] [SM4-VLSI], due to its being the only OSCCA-approved symmetric encryption algorithm allowed for use in China.

SM4 can be used with multiple modes (See Section 8).

1.3. Cryptanalysis

A number of attacks have been attempted on SM4, such as [SM4-Analysis] [SM4-Linear], but there are no known feasible attacks against the SM4 algorithm by the time of publishing this document.

There are, however, security concerns with regards to side-channel attacks [SideChannel] when the SM4 algorithm is implemented in a device [SM4-Power].

For instance, [SM4-Power] illustrated an attack by measuring the power consumption of the device. A chosen ciphertext attack, assuming a fixed correlation between the sub-keys and data mask, is able to recover the round key successfully. When the SM4 algorithm is implemented in hardware, the parameters/keys SHOULD be randomly generated without fixed correlation.

There have been improvements to the hardware embodiment of SM4 such as [SM4-VLSI] that may resist such attacks.

In order to improve security of the SM4 cryptographic process, secure white-box implementations such as [SM4-WhiteBox] have been proposed. Speed enhancements, such as [SM4-HiSpeed], have also been proposed.

2. Terms and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms and definitions apply to this document.

block length
: Bit-length of a message block.

key length
: Bit-length of a key.

key expansion algorithm
: An operation that converts a key into a round key.

rounds
: The number of iterations that the round function is run.

round key
: A key used in each round on the blockcipher, derived from the input key, also called a subkey.

word
: a 32-bit quantity

S-box
: The S (substitution) box function produces 8-bit output from 8-bit input, represented as Sbox(.)

3. Symbols And Abbreviations

S xor T
: bitwise execlusive-or of two 32-bit vectors S and T. S and T will always have the same length.

a <<< i
: 32-bit bitwise cyclic shift on a with i bits shifted left.

4. Compute Structure

The SM4 algorithm is a blockcipher, with block size of 128 bits and a key length of 128 bits.

Both encryption and key expansion uses 32 rounds of a nonlinear key schedule per block. Each round processes one of the four 32-bit words that constitute the block.

The structure of encryption and decryption are identical, except that the round key schedule has its order reversed during decryption.

Using a 8-bit S-box, it only uses exclusive-or, cyclic bit shifts and S-box lookups to execute.

5. Key And Key Parameters

Encryption key length is 128-bits, and represented below, where each MK_i, (i = 0, 1, 2, 3) is a word.

MK = (MK_0, MK_1, MK_2, MK_3)

The round key schedule is derived from the encryption key, represented as below where each rk_i (i = 0, ..., 31) is a word:

(rk_0, rk_1, ... , rk_31)

System parameters used for key expansion is represented as FK, where each FK_i (i = 0, ..., 3) is a word:

FK = (FK_0, FK_1, FK_2, FK_3)

Constant parameters used for key expansion is represented as CK, where each CK_i (i = 0, ..., 31) is a word:

CK = (CK_0, CK_1, ... , CK_31)

6. Round Function F

6.1. Round Parameter Structure

Given the 128-bit input below, where each $$X_i$ is a 32-bit word:

(X_0, X_1, X_2, X_3)

And the round key rk is a 32-bit word:

The round function F is defined as:

F(X_0, X_1, X_2, X_3, rk) = X_0 xor T(X_1 xor X_2 xor X_3 xor rk)

6.2. Mixer Substitution T

T is a reversible substitution function that outputs 32 bits from an input of 32 bits.

It consists of a non-linear transform tau and linear transform L.

T(.) = L(tau(.))

6.2.1. Non-linear Transformation tau

tau is composed of four parallel S-boxes.

Given a 32-bit input of A, where each a_i is a 8-bit string:

A = (a_0, a_1, a_2, a_3)

The output is a 32-bit B, where each b_i is a 8-bit string:

B = (b_0, b_1, b_2, b_3)

B is calculated as follows:

(b_0, b_1, b_2, b_3) = tau(A)

tau(A) = (Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3))

The Sbox lookup table is shown here:

	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

For example, input "EF" will produce an output read from the S-box table row E and column F, giving the result Sbox(EF) = 84.

6.3. Linear Substitution L

The output of non-linear transformation function tau is used as input to linear transformation function L.

Given B, a 32-bit input:

L produces a 32-bit output C:

C = L(B)

L(B) = B xor (B <<< 2) xor (B <<< 10) xor (B <<< 18) xor (B <<< 24)

7. Calculation

7.1. SM4 Encryption

The encryption algorithm consists of 32 rounds and 1 reverse transform R.

Given a 128-bit plaintext input, where each X_i is a 32-bit word:

(X_0, X_1, X_2, X_3)

The output is a 128-bit ciphertext, where each Y_i is a 32-bit word:

(Y_0, Y_1, Y_2, Y_3)

Each round key is designated as rk_i, where each rk_i is a 32-bit word and i = 0, 1, 2, ..., 31.

a. 32 rounds of calculation

i = 0, 1, ..., 31

X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i)

b. reverse transformation

(Y_0, Y_1, Y_2, Y_3) = R(X_32, X_33, X_34, X_35)

R(X_32, X_33, X_34, X_35) = (X_35, X_34, X_33, X_32)

Please refer to Section 12 for a sample calculation.

7.2. SM4 Decryption

Decryption takes an identical process as encryption, with the only difference the order of the round key sequence.

During decryption, the round key sequence is:

(rk_31, rk_30, ..., rk_0)

7.3. SM4 Key Expansion

Round keys used during encryption are derived from the encryption key.

Specifically, given the encryption key MK, where each MK_i is a 32-bit word:

MK = (MK_0, MK_1, MK_2, MK_3)

Each round key rk_i is created as follows, where i = 0, 1, ..., 31.

(K_0, K_1, K_2, K_3) = (MK_0 xor FK_0, MK_1 xor FK_1, MK_2 xor FK_2, MK_3 xor FK_3)

rk_i = K_{i + 4}

K_{i + 4} = K_i xor T' (K_{i + 1} xor K_{i + 2} xor K_{i + 3} xor CK_i)

Since the decryption key is identical to the encryption key, the round keys used in the decryption process are derived from the decryption key through the identical process to that of during encryption.

7.3.1. Transformation Function T'

The transformation function T' is created from T by replacing the linear transform function L with L'.

L'(B) = B xor (B <<< 13) xor (B <<< 23)

7.3.2. System Parameter FK

System parameter FK given in hexadecimal notation, is:

FK_0 = A3B1BAC6 FK_1 = 56AA3350 FK_2 = 677D9197 FK_3 = B27022DC

7.3.3. Constant Parameter CK

The method to retrieve values from the constant parameter CK is as follows.

Let ck_{i, j} be the j-th byte (i = 0, 1, ..., 31; j = 0, 1, 2, 3) of CK_i.

Therefore, each ck_{i, j} is a 8-bit string, and each CK_i a 32-bit word.

CK_i = (ck_{i, 0}, ck_{i, 1}, ck_{i, 2}, ck_{i, 3})

ck_{i, j} = (4i + j) x 7 (mod 256)

The constant parameter CK_i, (i = 0, 1, ..., 31) values, in hexadecimal, are:

```
CK_0  = 00070E15
CK_1  = 1C232A31
CK_2  = 383F464D
CK_3  = 545B6269
CK_4  = 70777E85
CK_5  = 8C939AA1
CK_6  = A8AFB6BD
CK_7  = C4CBD2D9
CK_8  = E0E7EEF5
CK_9  = FC030A11
CK_10 = 181F262D
CK_11 = 343B4249
CK_12 = 50575E65
CK_13 = 6C737A81
CK_14 = 888F969D
CK_15 = A4ABB2B9
CK_16 = C0C7CED5
CK_17 = DCE3EAF1
CK_18 = F8FF060D
CK_19 = 141B2229
CK_20 = 30373E45
CK_21 = 4C535A61
CK_22 = 686F767D
CK_23 = 848B9299
CK_24 = A0A7AEB5
CK_25 = BCC3CAD1
CK_26 = D8DFE6ED
CK_27 = F4FB0209
CK_28 = 10171E25
CK_29 = 2C333A41
CK_30 = 484F565D
CK_31 = 646B7279
```

8. Modes of Operation

This document defines multiple modes of operation for the SM4 blockcipher algorithm.

The CBC (Cipher Block Chaining), ECB (Electronic CodeBook), CFB (Cipher FeedBack), OFB (Output FeedBack) and CTR (Counter) modes are defined in [NIST.SP.800-38A] and utilized with the SM4 algorithm in the following sections.

8.1. Variables And Primitives

Hereinafter we define:

SM4Encrypt(P, K)

The SM4 algorithm that encrypts plaintext P with key K, described in Section 7.1

SM4Decrypt(C, K)
The SM4 algorithm that decrypts ciphertext C with key K, described in Section 7.2

b
block size in bits, defined as 128 for SM4

P_j
block j of ciphertext bitstring P

C_j
block j of ciphertext bitstring C

NBlocks(B, b)
Number of blocks of size b-bits in bitstring B

IV
Initialization vector

LSB(b, S)
Least significant b bits of the bitstring S

MSB(b, S)
Most significant b bits of the bitstring S

8.2. Initialization Vector

The CBC, CFB and OFB modes require an additional input to the encryption process, called the initialization vector (IV). The identical IV is used in the input of encryption as well as the decryption of the corresponding ciphertext.

The IV MUST fulfill the following requirements for security:

- CBC, CFB modes. The IV for a particular execution must be unpredictable.

- OFB mode. Each execution must be given a unique IV.

8.3. SM4-ECB

In SM4-ECB, the same key is utilized to create a fixed assignment for a plaintext block with a ciphertext block, meaning that a given plaintext block always gets encrypted to the same ciphertext block.

As described in [NIST.SP.800-38A], this mode should be avoided if this property is undesirable.

This mode requires input plaintext to be a multiple of the block size, which in this case of SM4 it is 128-bits. It also allows multiple blocks to be computed in parallel.

8.3.1. SM4-ECB Encryption

Inputs:

- P, plaintext, length MUST be multiple of b
- K, SM4 128-bit encryption key

Output:

- C, ciphertext, length is a multiple of b

C is defined as follows.

```
n = NBlocks(P, b)

for i = 1 to n
  C_i = SM4Encrypt(P_i, K)
end for

C = C_1 || ... || C_n
```

8.3.2. SM4-ECB Decryption

Inputs:

- C, ciphertext, length MUST be multiple of b
- K, SM4 128-bit encryption key

Output:

- P, plaintext, length is a multiple of b

P is defined as follows.

```
n = NBlocks(C, b)

for i = 1 to n
  P_i = SM4Decrypt(C_i, K)
end for

P = P_1 || ... || P_n
```

8.4. SM4-CBC

SM4-CBC is similar to SM4-ECB that the input plaintext MUST be a multiple of the block size, which is 128-bits in SM4. SM4-CBC requires an additional input, the IV, that is unpredictable for a particular execution of the encryption process.

Since CBC encryption relies on a foward cipher operation that depend on results of the previous operation, it cannot be parallelized. However, for decryption, since ciphertext blocks are already available, CBC parallel decryption is possible.

8.4.1. SM4-CBC Encryption

Inputs:

- P, plaintext, length MUST be multiple of b
- K, SM4 128-bit encryption key
- IV, 128-bit, unpredictable, initialization vector

Output:

- C, ciphertext, length is a multiple of b

C is defined as follows.

```
n = NBlocks(P, b)

C_1 = SM4Encrypt(P_1 xor IV, K)

for i = 2 to n
  C_i = SM4Encrypt(P_i xor C_{i - 1}, K)
end for

C = C_1 || ... || C_n
```

8.4.2. SM4-CBC Decryption

Inputs:

- C, ciphertext, length MUST be a multiple of b
- K, SM4 128-bit encryption key
- IV, 128-bit, unpredictable, initialization vector

Output:

- P, plaintext, length is multiple of b

P is defined as follows.

```
n = NBlocks(C, b)

P_1 = SM4Decrypt(C_1, K) xor IV

for i = 2 to n
  P_i = SM4Decrypt(C_i, K) xor C_{i - 1}
end for

P = P_1 || ... || P_n
```

8.5. SM4-CFB

SM4-CFB relies on feedback provided by successive ciphertext segments to generate output blocks. The plaintext given must be a multiple of the block size.

Similar to SM4-CBC, SM4-CFB requires an IV that is unpredictable for a particular execution of the encryption process.

SM4-CFB further allows setting a positive integer parameter s, that is less than or equal to the block size, to specify the size of each data segment. The same segment size must be used in encryption and decryption.

In SM4-CFB, since the input block to each forward cipher function depends on the output of the previous block (except the first that depends on the IV), encryption is not parallizable. Decryption, however, can be parallelized.

8.5.1. SM4-CFB Variants

SM4-CFB takes an integer s to determine segment size in its encryption and decryption routines. We define the following variants of SM4-CFB for various s:

- SM4-CFB-1, the 1-bit SM4-CFB mode, where s is set to 1.
- SM4-CFB-8, the 8-bit SM4-CFB mode, where s is set to 8.
- SM4-CFB-64, the 64-bit SM4-CFB mode, where s is set to 64.
- SM4-CFB-128, the 128-bit SM4-CFB mode, where s is set to 128.

8.5.2. SM4-CFB Encryption

Inputs:

- P#, plaintext, length MUST be multiple of s
- K, SM4 128-bit encryption key
- IV, 128-bit, unpredictable, initialization vector
- s, an integer 1 <= s <= b that defines segment size

Output:

- C#, ciphertext, length is a multiple of s

C# is defined as follows.

```
n = NBlocks(P#, s)

I_1 = IV
for i = 2 to n
  I_i = LSB(b - s, I_{i - 1}) || C#_{j - 1}
end for

for i = 1 to n
  O_j = SM4Encrypt(I_i, K)
end for

for i = 1 to n
  C#_i = P#_1 xor MSB(s, O_j)
end for

C# = C#_1 || ... || C#_n
```

8.5.3. SM4-CFB Decryption

Inputs:

- C#, ciphertext, length MUST be a multiple of s
- K, SM4 128-bit encryption key
- IV, 128-bit, unpredictable, initialization vector
- s, an integer 1 <= s <= b that defines segment size

Output:

- P#, plaintext, length is multiple of s

P is defined as follows.

```
n = NBlocks(P#, s)

I_1 = IV
for i = 2 to n
  I_i = LSB(b - s, I_{i - 1}) || C#_{j - 1}
end for

for i = 1 to n
  O_j = SM4Encrypt(I_i, K)
end for

for i = 1 to n
  P#_i = C#_1 xor MSB(s, O_j)
end for

P# = P#_1 || ... || P#_n
```

8.6. SM4-OFB

SM4-OFB is the application of SM4 through the Output Feedback mode. This mode requires that the IV is a nonce, meaning that the IV MUST be unique for each execution for an input key. OFB does not require the input plaintext to be a multiple of the block size.

In OFB, the routines for encryption and decryption are identical. As each forward cipher function (except the first) depends on previous results, both routines cannot be parallelized. However given a known IV, output blocks could be generated prior to the input of plaintext (encryption) or ciphertext (decryption).

8.6.1. SM4-OFB Encryption

Inputs:

- P, plaintext, composed of (n - 1) blocks of size b, with the last block P_n of size 1 <= u <= b
- K, SM4 128-bit encryption key
- IV, a nonce (a unique value for each execution per given key)

Output:

- C, ciphertext, composed of (n - 1) blocks of size b, with the last block C_n of size 1 <= u <= b

C is defined as follows.

```
n = NBlocks(P, b)

I_1 = IV
for i = 1 to (n - 1)
  O_i = SM4Encrypt(I_i)
  I_{i + 1} = O_i
end for

for i = 1 to (n - 1)
  C_i = P_i xor O_i
end for

C_n = P_n xor MSB(u, O_n)

C = C_1 || ... || C_n
```

8.6.2. SM4-OFB Decryption

Inputs:

- C, ciphertext, composed of (n - 1) blocks of size b, with the last block C_n of size 1 <= u <= b
- K, SM4 128-bit encryption key
- IV, the nonce used during encryption

Output:

- P, plaintext, composed of (n - 1) blocks of size b, with the last block P_n of size 1 <= u <= b

C is defined as follows.

```
n = NBlocks(C, b)

I_1 = IV
for i = 1 to (n - 1)
  O_i = SM4Encrypt(I_i)
  I_{i + 1} = O_i
end for

for i = 1 to (n - 1)
  P_i = C_i xor O_i
end for

P_n = C_n xor MSB(u, O_n)

P = P_1 || ... || P_n
```

8.7. SM4-CTR

SM4-CTR is an implementation of a stream cipher through a block cipher primitive. It generates a "keystream" of keys that are used to encrypt successive blocks, with the keystream created from the input key, a nonce (the IV) and an incremental counter. The counter could be any sequence that does not repeat within the block size.

Both SM4-CTR encryption and decryption routines could be parallelized, and random access is also possible.

8.7.1. SM4-CTR Encryption

Inputs:

- P, plaintext, composed of (n - 1) blocks of size b, with the last block P_n of size 1 <= u <= b
- K, SM4 128-bit encryption key
- IV, a nonce (a unique value for each execution per given key)
- T, a sequence of counters from T_1 to T_n

Output:

- C, ciphertext, composed of (n - 1) blocks of size b, with the last block C_n of size 1 <= u <= b

C is defined as follows.

```
n = NBlocks(P, b)

for i = 1 to n
  O_i = SM4Encrypt(T_i)
end for

for i = 1 to (n - 1)
  C_i = P_i xor O_i
end for

C_n = P_n xor MSB(u, O_n)

C = C_1 || ... || C_n
```

8.7.2. SM4-CTR Encryption

Inputs:

- C, ciphertext, composed of (n - 1) blocks of size b, with the last block C_n of size 1 <= u <= b
- K, SM4 128-bit encryption key
- IV, a nonce (a unique value for each execution per given key)
- T, a sequence of counters from T_1 to T_n

Output:

- P, plaintext, composed of (n - 1) blocks of size b, with the last block P_n of size 1 <= u <= b

P is defined as follows.

```
n = NBlocks(C, b)

for i = 1 to n
  O_i = SM4Encrypt(T_i)
end for

for i = 1 to (n - 1)
  P_i = C_i xor O_i
end for

P_n = C_n xor MSB(u, O_n)

C = C_1 || ... || C_n
```

9. Object Identifier

The Object Identifier for SM4 is the value "1.2.156.10197.1.104", specified in [GMT-0006-2012].

10. Security Considerations

- Products and services that utilize cryptography are regulated by OSCCA [OSCCA]; they must be explicitly approved or certified by OSCCA before being allowed to be sold or used in China.

- SM4 [GBT.32907-2016] is a blockcipher certified by OSCCA [OSCCA]. No formal proof of security is provided. There are no known feasible attacks against SM4 algorithm by the time of publishing this document. On the other hand, there are security concerns with regards to side-channel attacks, when the SM4 algorithm is implemented in a device [SM4-Power]. For instance, [SM4-Power] illustrated an attack by measuring the power consumption of the device. A chosen ciphertext attack, assuming a fixed correlation between the sub-keys and data mask, is able to recover the round key successfully. When the SM4 algorithm is implemented in hardware, the parameters/keys SHOULD be randomly generated without fixed correlation.

- SM4 is a blockcipher symmetric algorithm with key length of 128 bits. It is considered as an alternative to AES-128 [NIST.FIPS.197].

- SM4-CFB: The OFB mode requires a unique IV for every message that is ever encrypted under the given key. If, contrary to this requirement, the same IV is used for the encryption of more than one message, then the confidentiality of those messages may be compromised. In particular, if a plaintext block of any of these messages is known, say, the jth plaintext block, then the jth

output of the forward cipher function can be determined easily from the jth ciphertext block of the message. This information allows the jth plaintext block of any other message that is encrypted using the same IV to be easily recovered from the jth ciphertext block of that message. Confidentiality may similarly be compromised if any of the input blocks to the forward cipher function for the encryption of a message is designated as the IV for the encryption of another message under the given key.

11. IANA Considerations

This document does not require any action by IANA.

12. Appendix A: Example Calculations

12.1. Example 1.

This example demonstrates encryption of a plaintext.

Plaintext: 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

Encryption key: 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

Status of the round key (rk_i) and round output (X_i) per round:

```
rk_0  = F12186F9   X_4  = 27FAD345
rk_1  = 41662B61   X_5  = A18B4CB2
rk_2  = 5A6AB19A   X_6  = 11C1E22A
rk_3  = 7BA92077   X_7  = CC13E2EE
rk_4  = 367360F4   X_8  = F87C5BD5
rk_5  = 776A0C61   X_9  = 33220757
rk_6  = B6BB89B3   X_10 = 77F4C297
rk_7  = 24763151   X_11 = 7A96F2EB
rk_8  = A520307C   X_12 = 27DAC07F
rk_9  = B7584DBD   X_13 = 42DD0F19
rk_10 = C30753ED   X_14 = B8A5DA02
rk_11 = 7EE55B57   X_15 = 907127FA
rk_12 = 6988608C   X_16 = 8B952B83
rk_13 = 30D895B7   X_17 = D42B7C59
rk_14 = 44BA14AF   X_18 = 2FFC5831
rk_15 = 104495A1   X_19 = F69E6888
rk_16 = D120B428   X_20 = AF2432C4
rk_17 = 73B55FA3   X_21 = ED1EC85E
rk_18 = CC874966   X_22 = 55A3BA22
rk_19 = 92244439   X_23 = 124B18AA
rk_20 = E89E641F   X_24 = 6AE7725F
rk_21 = 98CA015A   X_25 = F4CBA1F9
rk_22 = C7159060   X_26 = 1DCDFA10
rk_23 = 99E1FD2E   X_27 = 2FF60603
rk_24 = B79BD80C   X_28 = EFF24FDC
rk_25 = 1D2115B0   X_29 = 6FE46B75
rk_26 = 0E228AEB   X_30 = 893450AD
rk_27 = F1780C81   X_31 = 7B938F4C
rk_28 = 428D3654   X_32 = 536E4246
rk_29 = 62293496   X_33 = 86B3E94F
rk_30 = 01CF72E5   X_34 = D206965E
rk_31 = 9124A012   X_35 = 681EDF34
```

Ciphertext: 68 1E DF 34 D2 06 96 5E 86 B3 E9 4F 53 6E 42 46

12.2. Example 2

This example demonstrates encryption of a plaintext 1,000,000 times repeatedly using a fixed encryption key.

Plaintext: 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

Encryption Key: 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10

Ciphertext: 59 52 98 C7 C6 FD 27 1F 04 02 F8 04 C3 3D 3F 66

13. References

13.1. Normative References

[GBT.32907-2016]
Standardization Administration of the People's Republic of China, "GB/T 32907-2016: Information security technology --- SM4 block cipher algorithm", August 2016, <http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=7803DE42D3BC5E80B0C3E5D8E873D56A>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

13.2. Informative References

[GB.15629.11-2003]
Standardization Administration of the People's Republic of China, "Information technology -- Telecommunications and information exchange between systems -- Local and metropolitan area networks -- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", May 2003, <http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=74B9DD11287E72408C19C4D3A360D1BD>.

[GMT-0002-2012]
Organization of State Commercial Administration of China, "GM/T 0002-2012: SM4 block cipher algorithm", March 2012, <http://www.oscca.gov.cn/Column/Column_32.htm>.

[GMT-0006-2012]
Organization of State Commercial Administration of China, "GM/T 0006-2012: Cryptographic Application Identifier Criterion Specification", March 2012, <http://www.oscca.gov.cn/Column/Column_32.htm>.

[LSW-Bio] Sun, M., "Lv Shu Wang -- A life in cryptography", November 2010, <http://press.ustc.edu.cn/sites/default/files/fujian/field_fujian_multi/20120113/%E5%90%95%E8%BF%B0%E6%9C%9B%20%E5%AF%86%E7%A0%81%E4%B8%80%E6%A0%B7%E7%9A%84%E4%BA%BA%E7%94%9F.pdf>.

[NIST.FIPS.197]
National Institute of Standards and Technology, "NIST FIPS 197: Advanced Encryption Standard (AES)", November 2001, <https://doi.org/10.6028/NIST.FIPS.197>.

[NIST.SP.800-38A]
Dworkin, M., "NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation -- Methods and Techniques", December 2001, <http://dx.doi.org/10.6028/NIST.SP.800-38A>.

[OSCCA] Organization of State Commercial Administration of China, "Organization of State Commercial Administration of China", May 2017, <http://www.oscca.gov.cn>.

[SideChannel]
Lei, Q., Wu, L., Zhang, S., Zhang, X., Li, X., Pan, L., and Z. Dong, "Software Hardware Co-design for Side-Channel Analysis Platform on Security Chips", December 2015, <https://doi.org/10.1109/CIS.2015.102>.

[SM4] Organization of State Commercial Administration of China, "SMS4 Cryptographic Algorithm For Wireless LAN Products", January 2006, <http://www.oscca.gov.cn/UpFile/200621016423197990.pdf>.

[SM4-Analysis]
Kim, T., Kim, J., Kim, S., and J. Sung, "Linear and Differential Cryptanalysis of Reduced SMS4 Block Cipher", June 2008, <https://eprint.iacr.org/2008/281>.

[SM4-En] Diffie, W. and G. Ledin, "SMS4 Encryption Algorithm for Wireless Networks", May 2008, <https://www.iacr.org/cryptodb/data/paper.php?pubkey=18006>.

[SM4-FPGA]
Cheng, H., Zhai, S., Fang, L., Ding, Q., and C. Huang, "Improvements of SM4 Algorithm and Application in Ethernet Encryption System Based on FPGA", July 2014, <https://www.researchgate.net/publication/287081686_Improvements_of_SM4_algorithm_and_application_in_Ethernet_encryption_system_based_on_FPGA>.

[SM4-HiSpeed]
Lv, Q., Li, L., and Y. Cao, "High-speed Encryption & Decryption System Based on SM4", July 2016, <http://dx.doi.org/10.14257/ijsia.2016.10.9.01>.

[SM4-Linear]
Liu, M. and J. Chen, "Improved Linear Attacks on the Chinese Block Cipher Standard", November 2014, <https://doi.org/10.1007/s11390-014-1495-9>.

[SM4-Power]
Du, Z., Wu, Z., Wang, M., and J. Rao, "Improved chosen-plaintext power analysis attack against SM4 at the round-output", October 2015, <http://dx.doi.org/10.6028/NIST.FIPS.180-4>.

[SM4-VLSI]
Yu, S., Li, K., Li, K., Qin, Y., and Z. Tong, "A VLSI implementation of an SM4 algorithm resistant to power analysis", July 2016, <https://doi.org/10.3233/JIFS-169011>.

[SM4-WhiteBox]
Bai, K. and C. Wu, "A secure white-box SM4 implementation", May 2008, <http://dx.doi.org/10.1002/sec.1394>.

Appendix A. Acknowledgements

The authors would like to thank the following persons for their valuable advice and input.

- Jack Lloyd and Daniel Wyatt of the Ribose rnp team for their input and implementation

Authors' Addresses

Ronald Henry Tse
Ribose
Suite 1111, 1 Pedder Street
Central, Hong Kong
Hong Kong

Email: ronald.tse@ribose.com
URI: https://www.ribose.com

Dr. Wai Kit Wong
Hang Seng Management College
Hang Shin Link, Siu Lek Yuen
Shatin, New Territories
Hong Kong

Email: wongwk@hsmc.edu.hk
URI: https://www.hsmc.edu.hk