The SM4 Block Cipher Algorithm And Its Modes Of Operations
draft-ribose-cfgm-sm4-02

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).

This document is a product of the Crypto Forum Research Group (CFRG).

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 https://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 April 21, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(https://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
   1.1. Purpose ............................................. 3
   1.2. History ............................................. 4
   1.3. Applications ....................................... 5
   1.4. Cryptanalysis ....................................... 5
2. Terms and Definitions ....................................... 5
3. Symbols And Abbreviations .................................. 6
4. Compute Structure ......................................... 6
5. Key And Key Parameters .................................... 6
6. Functions .................................................. 7
   6.1. Round Function F ..................................... 7
   6.2. Permutation T and T' ................................ 7
      6.2.1. Non-linear Transformation tau .................. 7
      6.2.2. Linear Transformation L and L' ................ 8
      6.2.3. S-box S ......................................... 8
7. Algorithm ................................................... 9
   7.1. Encryption ........................................... 9
   7.2. Decryption ........................................... 9
   7.3. Key Schedule ......................................... 10
      7.3.1. Family Key FK .................................. 10
      7.3.2. Constant Key CK ................................. 10
8. Modes of Operation ......................................... 11
   8.1. Variables And Primitives .............................. 11
   8.2. Initialization Vectors ............................... 12
   8.3. SM4-ECB ............................................. 12
      8.3.1. SM4-ECB Encryption .............................. 12
      8.3.2. SM4-ECB Decryption ............................. 13
   8.4. SM4-CBC ............................................. 13
      8.4.1. SM4-CBC Encryption .............................. 13
      8.4.2. SM4-CBC Decryption ............................. 14
   8.5. SM4-CFB ............................................. 15
      8.5.1. SM4-CFB Encryption .............................. 15
      8.5.2. SM4-CFB Decryption ............................. 15
      8.5.3. SM4-CFB Decryption ............................. 16
   8.6. SM4-OFB ............................................. 17
      8.6.1. SM4-OFB Encryption .............................. 17
      8.6.2. SM4-OFB Decryption ............................. 18
   8.7. SM4-CTR ............................................. 19
      8.7.1. SM4-CTR Encryption .............................. 19
      8.7.2. SM4-CTR Decryption ............................. 20
9. Object Identifier .......................................... 21
   9.1. GM/T OID ............................................ 21
   9.2. ISO OID ............................................. 21
1. Introduction

SM4 [GBT.32907-2016] [ISO.IEC.18033-3.AMD2] is a cryptographic standard issued by the Organization of State Commercial Administration of China [OSCCA] as an authorized cryptographic algorithm for the use within China. The algorithm is published in public.

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

1.1. Purpose

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 the "SMS4" algorithm [SM4] published in 2006, while this document follows the updated description and structure of [GBT.32907-2016] published in 2016. Sections 1 to 7 of this document directly map to the corresponding sections numbers of the [GBT.32907-2016] standard for convenience of the reader.

This document also provides additional information on the practical usage and implementation of SM4, specifying multiple modes of operations that are known to be used with SM4 and providing the SM4 OIDs.
1.2. 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 the OSCCA, officially renamed to "SM4" in 2012 in [GMT-0002-2012] published by the OSCCA, and finally standardized in 2016 as a Chinese National Standard (GB Standard) [GBT.32907-2016]. SM4 is also standardized in [ISO.IEC.18033-3.AMD2] by the International Organization for Standardization in 2017.

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 the 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
1.3. Applications

SM4 (and SM54) 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).

1.4. 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 hardware 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 round keys and data mask, is able to recover the round key successfully. When the SM4 algorithm is implemented in hardware, the parameters and keys SHOULD be randomly generated without fixed correlation.

There have been improvements to the hardware embodiment design for 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 S(.)

3. Symbols And Abbreviations

   S xor T
   bitwise exclusive-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 32-bits wide.

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

$$FK = (FK_0, FK_1, FK_2, FK_3)$$

The constant key 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. Functions

6.1. Round Function $F$

The round function $F$ is defined as:

$$F(X_0, X_1, X_2, X_3, rk) = X_0 \text{ xor } T(X_1 \text{ xor } X_2 \text{ xor } X_3 \text{ xor } rk)$$

Where:

- Each $X_i$ is 32 bits wide.
- The round key $rk$ is 32 bits wide.

6.2. Permutation $T$ and $T'$

$T$ is a reversible permutation that outputs 32 bits from an input of 32 bits.

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

$$T(.) = L(\tau(.))$$

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

$$T'(.) = L'(\tau(.))$$

6.2.1. Non-linear Transformation $\tau$

$\tau$ is composed of four parallel S-boxes.

Given a 32-bit input $A$, 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:

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

$$\tau(A) = (S(a_0), S(a_1), S(a_2), S(a_3))$$

### 6.2.2. Linear Transformation $L$ and $L'$

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

Given $B$, a 32-bit input.

The linear transformation $L'$ is defined as follows.

$$L(B) = B \oplus (B \ll 2) \oplus (B \ll 10) \oplus (B \ll 18) \oplus (B \ll 24)$$

The linear transformation $L'$ is defined as follows.

$$L'(B) = B \oplus (B \ll 13) \oplus (B \ll 23)$$

### 6.2.3. S-box S

The S-box $S$ used in $\tau$ is given in this lookup table in hexadecimal form:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>D6</td>
<td>90</td>
<td>E9</td>
<td>FE</td>
<td>CC</td>
<td>E1</td>
<td>3D</td>
<td>B7</td>
<td>16</td>
<td>B6</td>
<td>14</td>
<td>C2</td>
<td>28</td>
<td>FB</td>
<td>2C</td>
</tr>
<tr>
<td>1</td>
<td>2B</td>
<td>67</td>
<td>9A</td>
<td>76</td>
<td>2A</td>
<td>BE</td>
<td>04</td>
<td>C3</td>
<td>A4</td>
<td>43</td>
<td>26</td>
<td>49</td>
<td>86</td>
<td>06</td>
<td>99</td>
</tr>
<tr>
<td>2</td>
<td>9C</td>
<td>42</td>
<td>50</td>
<td>F4</td>
<td>91</td>
<td>EF</td>
<td>98</td>
<td>7A</td>
<td>33</td>
<td>54</td>
<td>0B</td>
<td>43</td>
<td>ED</td>
<td>CF</td>
<td>AC</td>
</tr>
<tr>
<td>3</td>
<td>E4</td>
<td>B3</td>
<td>1C</td>
<td>A9</td>
<td>C9</td>
<td>08</td>
<td>E8</td>
<td>95</td>
<td>80</td>
<td>DF</td>
<td>94</td>
<td>FA</td>
<td>75</td>
<td>8F</td>
<td>3F</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>07</td>
<td>A7</td>
<td>FC</td>
<td>F3</td>
<td>73</td>
<td>17</td>
<td>BA</td>
<td>83</td>
<td>59</td>
<td>3C</td>
<td>19</td>
<td>E6</td>
<td>85</td>
<td>4F</td>
</tr>
<tr>
<td>5</td>
<td>68</td>
<td>6B</td>
<td>81</td>
<td>B2</td>
<td>71</td>
<td>64</td>
<td>DA</td>
<td>8B</td>
<td>EE</td>
<td>0F</td>
<td>22</td>
<td>56</td>
<td>95</td>
<td>37</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>1E</td>
<td>24</td>
<td>0E</td>
<td>5E</td>
<td>63</td>
<td>58</td>
<td>D1</td>
<td>A2</td>
<td>94</td>
<td>85</td>
<td>3B</td>
<td>24</td>
<td>7C</td>
<td>32</td>
<td>61</td>
</tr>
<tr>
<td>7</td>
<td>D4</td>
<td>00</td>
<td>46</td>
<td>57</td>
<td>9F</td>
<td>D3</td>
<td>27</td>
<td>52</td>
<td>4C</td>
<td>36</td>
<td>02</td>
<td>A7</td>
<td>0A</td>
<td>C4</td>
<td>E8</td>
</tr>
<tr>
<td>8</td>
<td>EA</td>
<td>BF</td>
<td>8A</td>
<td>D2</td>
<td>40</td>
<td>C7</td>
<td>38</td>
<td>B5</td>
<td>A3</td>
<td>F7</td>
<td>2F</td>
<td>CE</td>
<td>F9</td>
<td>61</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>E0</td>
<td>AE</td>
<td>5D</td>
<td>A4</td>
<td>9B</td>
<td>34</td>
<td>1A</td>
<td>55</td>
<td>AD</td>
<td>93</td>
<td>32</td>
<td>30</td>
<td>F5</td>
<td>8C</td>
<td>B1</td>
</tr>
<tr>
<td>A</td>
<td>1D</td>
<td>F6</td>
<td>E2</td>
<td>2E</td>
<td>82</td>
<td>66</td>
<td>CA</td>
<td>60</td>
<td>C0</td>
<td>29</td>
<td>C9</td>
<td>AB</td>
<td>D0</td>
<td>53</td>
<td>4E</td>
</tr>
<tr>
<td>B</td>
<td>D5</td>
<td>DB</td>
<td>37</td>
<td>45</td>
<td>DE</td>
<td>FD</td>
<td>8E</td>
<td>2F</td>
<td>03</td>
<td>FF</td>
<td>6A</td>
<td>72</td>
<td>6D</td>
<td>6C</td>
<td>5B</td>
</tr>
<tr>
<td>C</td>
<td>8D</td>
<td>1B</td>
<td>AF</td>
<td>92</td>
<td>BB</td>
<td>DD</td>
<td>BC</td>
<td>7F</td>
<td>11</td>
<td>D9</td>
<td>5C</td>
<td>41</td>
<td>1F</td>
<td>10</td>
<td>5A</td>
</tr>
<tr>
<td>D</td>
<td>0A</td>
<td>C1</td>
<td>31</td>
<td>88</td>
<td>A5</td>
<td>CD</td>
<td>7B</td>
<td>BD</td>
<td>2D</td>
<td>74</td>
<td>D0</td>
<td>12</td>
<td>B8</td>
<td>E5</td>
<td>B4</td>
</tr>
<tr>
<td>E</td>
<td>89</td>
<td>69</td>
<td>97</td>
<td>4A</td>
<td>0C</td>
<td>96</td>
<td>77</td>
<td>7E</td>
<td>65</td>
<td>B9</td>
<td>F1</td>
<td>09</td>
<td>C5</td>
<td>6E</td>
<td>C6</td>
</tr>
<tr>
<td>F</td>
<td>18</td>
<td>F0</td>
<td>7D</td>
<td>EC</td>
<td>3A</td>
<td>DC</td>
<td>4D</td>
<td>20</td>
<td>79</td>
<td>EE</td>
<td>5F</td>
<td>3E</td>
<td>D7</td>
<td>CB</td>
<td>39</td>
</tr>
</tbody>
</table>
For example, input "EF" will produce an output read from the S-box table row E and column F, giving the result S(EF) = 84.

7. Algorithm

7.1. 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 sample calculations.

7.2. 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. Key Schedule

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

Specifically, given the encryption key MK, where each MK_i is 32 bits wide:

\[ MK = (MK_0, MK_1, MK_2, MK_3) \]

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. Family Key FK

Family key FK given in hexadecimal notation, is:

\[ FK_0 = A3B1BAC6 \]
\[ FK_1 = 56AA3350 \]
\[ FK_2 = 677D9197 \]
\[ FK_3 = B27022DC \]

7.3.2. Constant Key CK

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

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

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

\[ CK_i = (c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3}) \]

\[ c_{i,j} = (4i + j) \times 7 \mod 256 \]

The values of the constant key CK_i, where (i = 0, 1, ..., 31), in hexadecimal, are:
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 Vectors

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.

Generation of IV values MUST take into account of the considerations in Section 10 recommended by [BC-EVAL].

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

Output:
   o 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 forward 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
o K, SM4 128-bit encryption key
o IV, 128-bit, unpredictable, initialization vector

Output:
o C, ciphertext, length is a multiple of b

C is defined as follows.

\[ n = \text{NBlocks}(P, b) \]
\[ C_1 = \text{SM4Encrypt}(P_1 \text{xor} IV, K) \]
for \( i = 2 \) to \( n \)
\[ C_i = \text{SM4Encrypt}(P_i \text{xor} C_{i-1}, K) \]
end for

\[ C = C_1 \parallel ... \parallel C_n \]

8.4.2. SM4-CBC Decryption

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

Output:
o P, plaintext, length is multiple of b

P is defined as follows.

\[ n = \text{NBlocks}(C, b) \]
\[ P_1 = \text{SM4Decrypt}(C_1, K) \text{xor} IV \]
for \( i = 2 \) to \( n \)
\[ P_i = \text{SM4Decrypt}(C_i, K) \text{xor} C_{i-1} \]
end for

\[ P = P_1 \parallel ... \parallel P_n \]
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 parallelizable. 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.

\[ n = \text{NBlocks}(P\#, s) \]

\[ I_1 = \text{IV} \]

for i = 2 to n

\[ I_i = \text{LSB}(b - s, I_{i - 1}) \parallel C\#_{j - 1} \]

end for

for i = 1 to n

\[ O_j = \text{SM4Encrypt}(I_i, K) \]

end for

for i = 1 to n

\[ C\#_i = P\#_1 \text{xor} \text{MSB}(s, O_j) \]

end for

\[ C\# = C\#_1 \parallel ... \parallel C\#_n \]

### 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.
\[ n = \text{NBlocks}(P\#, s) \]

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

for \( i = 1 \) to \( n \)
\[ O_j = \text{SM4Encrypt}(I_i, K) \]
end for

for \( i = 1 \) to \( n \)
\[ P\#_i = C\#_1 \text{xor MSB}(s, O_j) \]
end for

\[ P\# = P\#_1 \parallel ... \parallel P\#_n \]

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 \( 1 \leq u \leq b \)
- \( K \), 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 \( 1 \leq u \leq b \)

\( C \) 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 Decryption

Inputs:

- C, ciphertext, composed of $(n - 1)$ blocks of size $b$, with the last block $C_n$ of size $1 \leq u \leq b$
- 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 $1 \leq u \leq b$

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 identified through these OIDs.

9.1. GM/T OID

"1.2.156.10197.1.104" for "SM4 Algorithm" [GMT-0006-2012].

9.2. ISO OID

"1.0.18033.3.2.4" for "id-bc128-sm4" [ISO.IEC.18033-3.AMD2], described below.

```plaintext
is18033-3  OID ::= {iso(1) standard(0) is18033(18033) part3(3)}
id-bc128   OID ::= {is18033-3 block-cipher-128-bit(2)}
id-bc128-sm4 OID ::= {id-bc128 sm4(4)}
```

10. Security Considerations

- Products and services that utilize cryptography are regulated by the OSCCA [OSCCA]; they must be explicitly approved or certified by the OSCCA before being allowed to be sold or used in China.
- 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 [GBT.32907-2016] is a blockcipher certified by the 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, but there are security concerns with regards to side-channel attacks when the SM4 algorithm is implemented in hardware. See Section 1.4 for more details.
- The IV does not have to be secret. The IV itself, or criteria enough to determine it, MAY be transmitted with ciphertext.
- SM4-ECB: ECB is one of the four original modes defined for DES. With its problem well known to "leak quite a large amount of information" [BC-EVAL], it SHOULD NOT be used in most cases.
- SM4-CBC, SM4-CFB, SM4-OFB: CBC, CFB and OFB are IV-based modes of operation originally defined for DES.

When using these modes of operation, the IV SHOULD be random to preserve message confidentiality [BC-EVAL]. It is shown in the same document that CBC, CFB, OFB, the variants #CBC, #CFB that utilize the
recommendation of [NIST.SP.800-38A] to make CBC and CFB nonce-based, are SemCPA secure as probabilistic encryption schemes.

Various attack scenarios have been described in [BC-EVAL] and these modes SHOULD NOT be used unless for compatibility reasons.

- SM4-CTR: CTR is considered to be the "best" mode of operation within [NIST.SP.800-38A] as it is considered SemCPA secure as a nonce-based encryption scheme, providing provable-security guarantees as good as the classic modes of operation (ECB, CBC, CFB, OFB) [BC-EVAL].

Users with no need of authenticity, non-malleability and chosen-ciphertext (CCA) security MAY utilize this mode of operation [BC-EVAL].

11. IANA Considerations

This document does not require any action by IANA.

12. Appendix A: Example Calculations

12.1. Examples From GB/T 32907-2016

12.1.1. Example 1

This is example 1 provided by [GBT.32907-2016] to demonstrate 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:
12.1.2. Example 2

This example is provided by [GBT.32907-2016] to demonstrate 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
```

12.2. Examples For Various Modes Of Operations

The following examples can be verified using open-source cryptographic libraries including:

- the Botan cryptographic library [BOTAN] with SM4 support, and
- the OpenSSL Cryptography and SSL/TLS Toolkit [OPENSSL] with SM4 support

12.2.1. SM4-ECB Example

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:

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

12.2.2. SM4-CBC Example

Plaintext:

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

IV:

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

Ciphertext:

```
26 77 F4 6B 09 C1 22 CC 97 55 33 10 5B D4 A2 2A
F6 12 5F 72 75 CE 55 2C 3A 2B BC F5 33 DE 8A 3B
```
12.2.3. SM4-OFB Example

Plaintext:

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

IV:

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

Ciphertext:

```
69 3D 9A 53 5B AD 5B B1 78 6F 53 D7 25 3A 70 56
F2 07 5D 2B B5 23 5F 58 D5 00 27 E4 17 7D 2B CE
```

12.2.4. SM4-CFB Example

Plaintext:

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

IV:

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

Ciphertext:

```
69 3D 9A 53 5B AD 5B B1 78 6F 53 D7 25 3A 70 56
9E D2 58 A8 5A 04 67 CC 92 AA B3 93 DD 97 89 95
```

12.2.5. SM4-CTR Example

Plaintext:

```
AA AA AA AA AA AA AA AA BB BB BB BB BB BB BB BB
CC CC CC CC CC CC CC CC DD DD DD DD DD DD DD
EE EE EE EE EE EE EE EE FF FF FF FF FF FF FF FF
EE EE EE EE EE EE EE EE AA AA AA AA AA AA AA AA
```
Encryption Key:

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

IV:

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

Ciphertext:

C2 B4 75 9E 78 AC 3C F4 3D 08 52 F4 E8 D5 F9 FD
72 56 E8 A5 FC B6 5A 35 0E E0 06 91 2E 44 49
2A 0B 17 E1 B8 5B 06 0D 0F BA 61 2D 8A 95 83 16
38 B3 61 FD 5F FA CD 94 2F 08 14 85 A8 3C A3 5D

13. References

13.1. Normative References


13.2. Informative References


[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%A F%86%E7%A0%81%E4%B8%80%E6%A0%B7%E7%9A%84%E4%BA%BA%E7%94%9F.pdf>.


Appendix A. Acknowledgements

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

- Erick Borsboom for assisting the lengthy review of this document
- 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

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