Q/GK

深圳壁虎新能源汽车科技有限公司企业标准

Q/GK TD035-2024

信息安全密码算法规范

Cybersecurity Cryptographic Algorithm Specification

2024-12-27 发布　　2025-01-01 实施

深圳壁虎新能源汽车科技有限公司发布

目　录

前　言

本标准由深圳壁虎新能源汽车科技有限公司提出。

本标准由深圳壁虎新能源汽车科技有限公司研发中心归口。

本标准起草单位：深圳壁虎新能源汽车科技有限公司研发中心智能网联部。

本标准起草人： 严一鑫、王尚俊。

本标准审核人： 钟幸原。

本标准批准人： 余欣。

本文件首次发布。

This standard was proposed by Shenzhen Gecko New Energy Vehicle Technologies Company Limited.

This standard was under management of Shenzhen Gecko New Energy Vehicle Technologies Company Limited.

This standard was drafted by Intelligent and Connectivity Department of R&D center at Shenzhen Gecko New Energy Vehicle Technologies Company Limited

This standard was drafted by Yixin Yan, Shangjun Wang.

This standard was reviewed by Xingyuan Zhong.

This standard was approved by Xin Yu.

This standard was the first published.

信息安全密码算法规范
Cybersecurity Cryptographic Algorithm Specification

1 范围/Scope

本文档旨在建立车内电子控制单元 ECU 和外部设备进行身份校验的算法和流程说明。

The purpose of this document is to establish the algorithm and process description for identity verification of in-vehicle ECUs and external devices.

本文档适用于深圳壁虎新能源汽车科技有限公司（简称“壁虎科技”）全新开发的所有车型。

This document applies to all the vehicles which are newly developed by Shenzhen Gecko New Energy Vehicle Technologies Company Limited (simplified as Gecko Technologies).

2 规范性引用文件/References

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

The following documents are essential for the application of this document. In the case of a dated reference, only the dated version applies to this document. For undated references, the most recent version of the document (including all change orders) applies to this document.

Ref.1　　RFC4493

3 术语和定义/Terms and Definitions

3.1 术语/Terms

3.1.1 种子/Seed

种子数组/Seed array：Unsigned Seed[16] ={S0,S1,S2,…,S14,S15}；S0~S15：1 byte。

随机数组必须由ECU硬件安全模块随机生成。

The seed array shall be randomly generated by the security hardware module.

3.1.2 原始密钥/Key

原始密钥数组/Original key array：Unsigned Key[16] ={K0,K1,K2,…,K14,K15}；K0~K15：1 byte。

原始密钥应由壁虎科技定义，应存储在ECU内HSM、TEE等安全区域内。若没有硬件安全模块，应对密钥进行安全存储，如存储在OTP区域，对密钥的存储、使用以及读取有相应的保护措施。

The original key should be defined by Gecko Technology and should be stored in a secure area such as HSM and TEE within the ECU. If it is stored in the OTP area, there are corresponding protection measures for the storage, use and reading of the key.

3.1.3 计算密钥/Calculating Key

计算密钥数组/Calculating key array：Unsigned W[44] ={W0,W1,W2,…,W42,W43}； W0~W43:4 bytes，初始值/initial value=0x00000000。

3.1.4 计算结果/Calculation Result

计算结果数组/Calculation result array：Unsigned CR[16] ={CR0,CR1,CR2,…,CR14,CR15}；CR0~CR15：1 byte, 初始值/initial value=0x00。

3.1.5 过程数组/Process Array

Sa, Sb, Sc和Sd是计算过程中数组。
Sa, Sb, Sc and Sd are the arrays used in the calculating process.
Unsigned Sa[16] ={Sa0,Sa1,Sa2,…,Sa14,Sa15}；Sa0~Sa15：1 byte, 初始值/initial value=0x00。
Unsigned Sb[16] ={Sb0,Sb1,Sb2,…,Sb14,Sb15}；Sb0~Sb15：1 byte, 初始值/initial value=0x00。
Unsigned Sc[16] ={Sc0,Sc1,Sc2,…,Sc14,Sc15}；Sc0~Sc15：1 byte, 初始值/initial value=0x00。
Unsigned Sd[16] ={Sd0,Sd1,Sd2,…,Sd14,Sd15}；Sd0~Sd15：1 byte, 初始值/initial value=0x00。

3.2 缩写/Abbreviations

表/Table 1 缩写/Abbreviations

缩写/Abbreviation	定义/Definition
AES	Advanced Encryption Standard，高级加密标准
CAN	Controller Area Network，控制器局域网
CAN FD	CAN with Flexible Data rate，可变数据速率的 CAN
DID	Data IDentifier，数据 ID
ECU	Electronic Control Unit，电子控制单元
EOL	End of Line，下线
ID	IDentifier，标识符
OBD	On-Board Dianostic，车载自动诊断系统
UDS	Unified Diagnostic Service，统一诊断服务
CMAC	Cipher-based Message Authentication Code，基于密码的消息认证码
MSB(x)	The most-significant bit of the string x，字符串 x 的最高有效位

4 信息安全算法/CyberSecurity Algorithms

4.1 对称算法 AES

4.1.1 概述/Overview

信息安全建议采用国际认可的安全密码算法，如AES-128，RSA-2048，SM4等。

It is recommended to use internationally recognized secure cryptographic algorithms, such as AES-128, RSA-2048, SM4, etc.

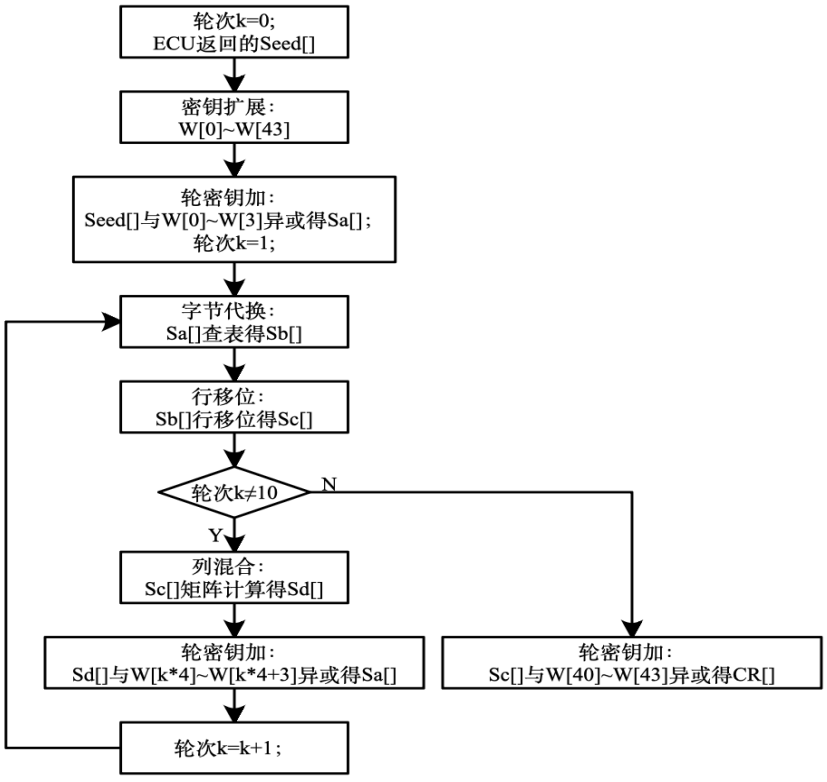
壁虎科技采用AES-128算法去计算CMAC作为校验的依据，AES的运算都是以4X4字节表示的二维数组矩阵为基础，使用16 bytes/128 bits的密钥对数据进行加密和解密。

Gecko Technology uses the AES-128 algorithm to calculate the CMAC as the basis for verification, and the AES operation is based on a two-dimensional array matrix represented by 4X4 bytes, and the data is encrypted and decrypted with a 16 bytes/128-bits key.

4.1.2 AES 加密基本流程/Basic process of AES encryption

AES-128算法总共需要计算11轮次，计算基本流程包括密钥扩展、字节代换、行移位、列混合、轮密钥加，如图1所示。

The AES-128 algorithm requires a total of 11 rounds of computation, and the basic computation process includes key expansion, byte substitution, row shifting, column mixing, and round key addition, as shown in Figure 1.



图/Figure 1　AES-128 算法计算流程/Calculating Process of AES-128 Algorithm

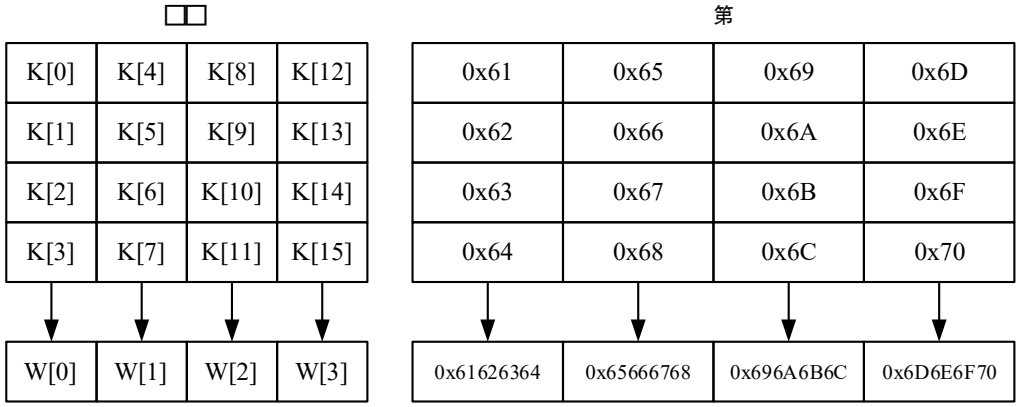
1） 密钥扩展

AES-128中原始密钥key为16 bytes/128 bits，运算中需要11个矩阵大小的密钥，所以在开始加密前，需要进行密钥扩展。

In AES-128, the original key is 16 bytes/128 bits, and 11 matrix-sized keys are required for the operation, so the key needs to be expanded before starting encryption.

以原始密钥Key=“abcdefghijklmnop”={0x61, 0x62,…,0x6F,0x70}为例，首先按图2所示的过程获得第一轮加密用到的计算密钥，即W[0]=0x61626364，W[1]=0x65666768，W[2]=0x696A6B6C，W[3]=0x6D6E6F70。

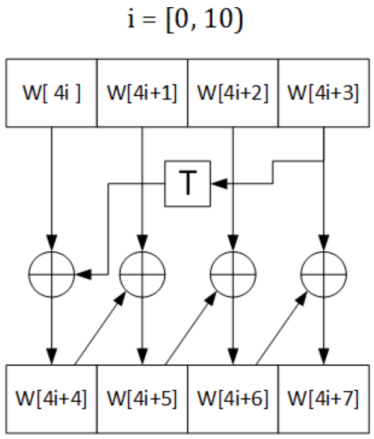
Taking the original key Key= "abcdefghijklmnop"={0x61, 0x62,...,0x6F,0x70} as an example, firstly, the calculation key used in the first round of encryption is obtained according to the process shown in Figure 2, that is, W[0]=0x61626364, W[1]=0x65666768, W[2]=0x696A6B6C, and W[3]=0x6D6E6F70.



图/Figure 2　计算密钥 W[0]~W[3]的获取方法/Method to getting W[0]~W[3]

然后，按照图3的所示的过程和公式1获取剩余的计算密钥W[4]~W[43]。

Then, follow the process shown in Figure 3 and Equation 1 to obtain the remaining computational keys W[4]~W[43].



图/Figure 3　计算密钥 W[4]~W[43]的获取方法/Method to getting W[4]~W[43]

$$W[n]=\begin{cases} W[n-4]\oplus W[n-1],\ \text{if n! = 4的倍数；} \\ W[N-4]\oplus Mix(W[n-1])\oplus Rcon[(n/4)-1],\ \text{if n = 4的倍数。} \end{cases}$$ 式1

其中：

a) Mix(x)=SubWord(RotWord(x))；

b) RotWord(x)为循环左移一位，如输入0x12345678，输出0x34567812；

c) SubWord(x)为字节替换，可以参考图4的字节替换，；

d) Rcon(i)为轮常量异或，可通过表2查询获得。

表/Table 2　Rcon(i)取值表/Rcon(i) value table

i	Rcon(i)（Hex）	i	Rcon(i)（Hex）
0	0x01000000	5	0x20000000

i	Rcon(i)（Hex）	i	Rcon(i)（Hex）
1	0x02000000	6	0x40000000
2	0x04000000	7	0x80000000
3	0x08000000	8	0x1B000000
4	0x10000000	9	0x36000000

2） 字节代换/Byte Substitution

将k=0~9轮次轮密钥加的计算结果赋值给Sa数组；将Sa数组通过图4所示的S盒按字节代换得到数组Sb。

The calculation result of k=0~9 rounds of key addition is assigned to the Sa array. The Sa array is byte-swapped by the S-box shown in Figure 4 to obtain the array Sb.

具体代换过程为：以对应字节16进制数的高位为行编号，以对应字节16进制数的低位为列编号，所对应的单元格中的数据即为代换后的数值。例如，0x12代换后为0xC9；0xF6代换后为0x42。

The specific substitution process is as follows: the row number is numbered by the high digit of the corresponding byte hexadecimal number, and the column number is taken by the low digit of the corresponding byte hexadecimal number, and the data in the corresponding cell is the value after substitution. For example, 0x12 is 0xC9 after substitution; 0xF6 replaced by 0x42.

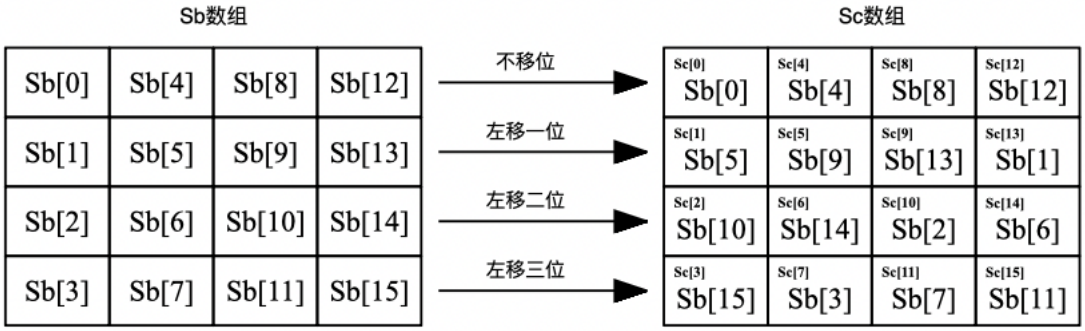
行/列	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x63	0x7c	0x77	0x7b	0xf2	0x6b	0x6f	0xc5	0x30	0x01	0x67	0x2b	0xfe	0xd7	0xab	0x76
1	0xca	0x82	0xc9	0x7d	0xfa	0x59	0x47	0xf0	0xad	0xd4	0xa2	0xaf	0x9c	0xa4	0x72	0xc0
2	0xb7	0xfd	0x93	0x26	0x36	0x3f	0xf7	0xcc	0x34	0xa5	0xe5	0xf1	0x71	0xd8	0x31	0x15
3	0x04	0xc7	0x23	0xc3	0x18	0x96	0x05	0x9a	0x07	0x12	0x80	0xe2	0xeb	0x27	0xb2	0x75
4	0x09	0x83	0x2c	0x1a	0x1b	0x6e	0x5a	0xa0	0x52	0x3b	0xd6	0xb3	0x29	0xe3	0x2f	0x84
5	0x53	0xd1	0x00	0xed	0x20	0xfc	0xb1	0x5b	0x6a	0xcb	0xbe	0x39	0x4a	0x4c	0x58	0xcf
6	0xd0	0xef	0xaa	0xfb	0x43	0x4d	0x33	0x85	0x45	0xf9	0x02	0x7f	0x50	0x3c	0x9f	0xa8
7	0x51	0xa3	0x40	0x8f	0x92	0x9d	0x38	0xf5	0xbc	0xb6	0xda	0x21	0x10	0xff	0xf3	0xd2
8	0xcd	0x0c	0x13	0xec	0x5f	0x97	0x44	0x17	0xc4	0xa7	0x7e	0x3d	0x64	0x5d	0x19	0x73
9	0x60	0x81	0x4f	0xdc	0x22	0x2a	0x90	0x88	0x46	0xee	0xb8	0x14	0xde	0x5e	0x0b	0xdb
A	0xe0	0x32	0x3a	0x0a	0x49	0x06	0x24	0x5c	0xc2	0xd3	0xac	0x62	0x91	0x95	0xe4	0x79
B	0xe7	0xc8	0x37	0x6d	0x8d	0xd5	0x4e	0xa9	0x6c	0x56	0xf4	0xea	0x65	0x7a	0xae	0x08
C	0xba	0x78	0x25	0x2e	0x1c	0xa6	0xb4	0xc6	0xe8	0xdd	0x74	0x1f	0x4b	0xbd	0x8b	0x8a
D	0x70	0x3e	0xb5	0x66	0x48	0x03	0xf6	0x0e	0x61	0x35	0x57	0xb9	0x86	0xc1	0x1d	0x9e
E	0xe1	0xf8	0x98	0x11	0x69	0xd9	0x8e	0x94	0x9b	0x1e	0x87	0xe9	0xce	0x55	0x28	0xdf
F	0x8c	0xa1	0x89	0x0d	0xbf	0xe6	0x42	0x68	0x41	0x99	0x2d	0x0f	0xb0	0x54	0xbb	0x16

图/Figure 4　字节代换数值表/Value Table of Byte Substitution

3） 行位移/ShiftRows

将Sb数组通过图5所示的行位移得到Sc数组。

The Sb array is obtained by displacing the rows as shown in Figure 5.



图/Figure 5　行位移过程/Process of ShiftRows

4） 列混合/MixColumns

将k=1~9轮次行位移得到的Sc数组通过公式2进行列混合得到Sd数组。

The Sc array obtained by k=1~9 rounds of row displacement is mixed by equation 2 to obtain the Sd array.

$$\begin{bmatrix} Sd0 & Sd4 & Sd8 & Sd12 \\ Sd1 & Sd5 & Sd9 & Sd13 \\ Sd2 & Sd6 & Sd10 & Sd14 \\ Sd3 & Sd7 & Sd11 & Sd15 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} Sc0 & Sc4 & Sc8 & Sc12 \\ Sc1 & Sc5 & Sc9 & Sc13 \\ Sc2 & Sc6 & Sc10 & Sc14 \\ Sc3 & Sc7 & Sc11 & Sc15 \end{bmatrix} \quad \text{式2}$$

式2定义的矩阵乘法是基于GF(2^8)有限域的二元运算,可通过式3~式6进行等效计算。

The matrix multiplication defined in Equation 2 is a binary operation based on the finite field of GF(28), which can be equivalent by Equation 3~Equation 6.

$$Sd[0+4i] = (\{02\} \cdot Sc[0+4i]) \oplus (Sc[1+4i]) \oplus (Sc[2+4i]) \oplus (\{03\} \cdot Sc[3+4i]) \quad \text{式3}$$

$$Sd[1+4i] = (Sc[0+4i]) \oplus (Sc[1+4i]) \oplus (\{03\} \cdot Sc[2+4i]) \oplus (\{02\} \cdot Sc[3+4i]) \quad \text{式4}$$

$$Sd[2+4i] = (Sc[0+4i]) \oplus (\{03\} \cdot Sc[1+4i]) \oplus (\{02\} \cdot Sc[2+4i]) \oplus (Sc[3+4i]) \quad \text{式5}$$

$$Sd[3+4i] = (\{03\} \cdot Sc[0+4i]) \oplus (\{02\} \cdot Sc[1+4i]) \oplus (Sc[2+4i]) \oplus (Sc[3+4i]) \quad \text{式6}$$

其中i取值0~3

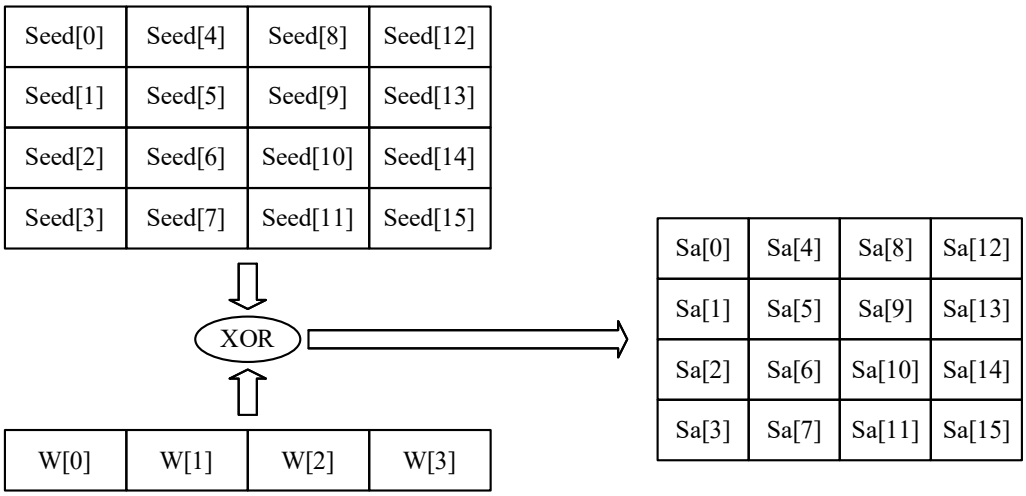
where i takes the value 0~3.

5） 轮密钥加/AddRoundKey

根据轮次不同,轮密钥加分为三种情况：

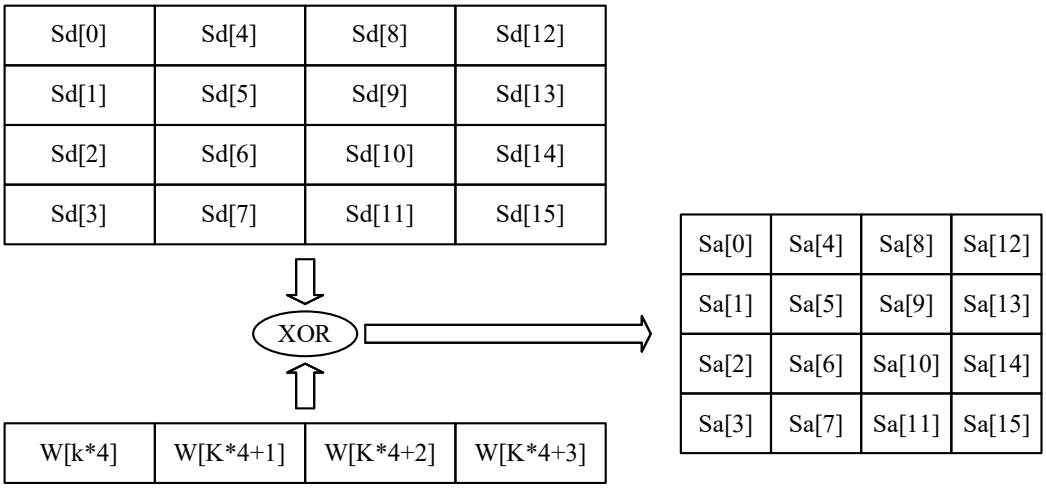
According to the number of rounds, there are three types of round key additions:

a) 轮次k=0时，直接将Seed[]与W[0]~W[3]进行异或运算，并将运算结果赋给Sa[]，如图6所示。
When the round k=0, Seed[] and W[0]~W[3] are directly XOR arithmetic, and the result is assigned to Sa[], as shown in Figure 6.



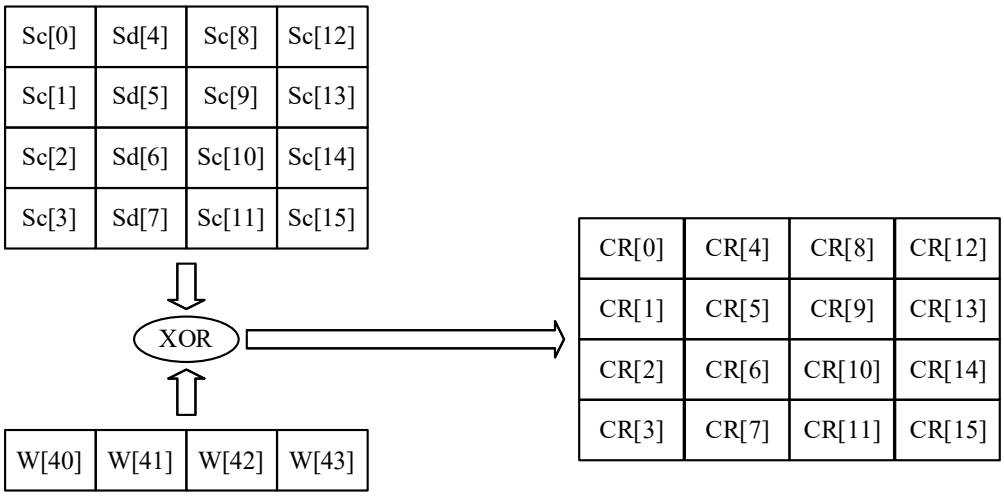
图/Figure 6　轮密钥加-k=0/AddRoundKey-k=0

b) 轮次k=1~9时，将对应轮次列混合的结果Sd[]与W[k*4]~W[k*4+3]进行异或运算，并将运算结果赋给Sa[]，如图7所示。When the round k=1~9, the result of the corresponding round column mixing is XOR operation is performed with W[k*4]~W[k*4+3], and the result is assigned to Sa[], as shown in Figure 7.



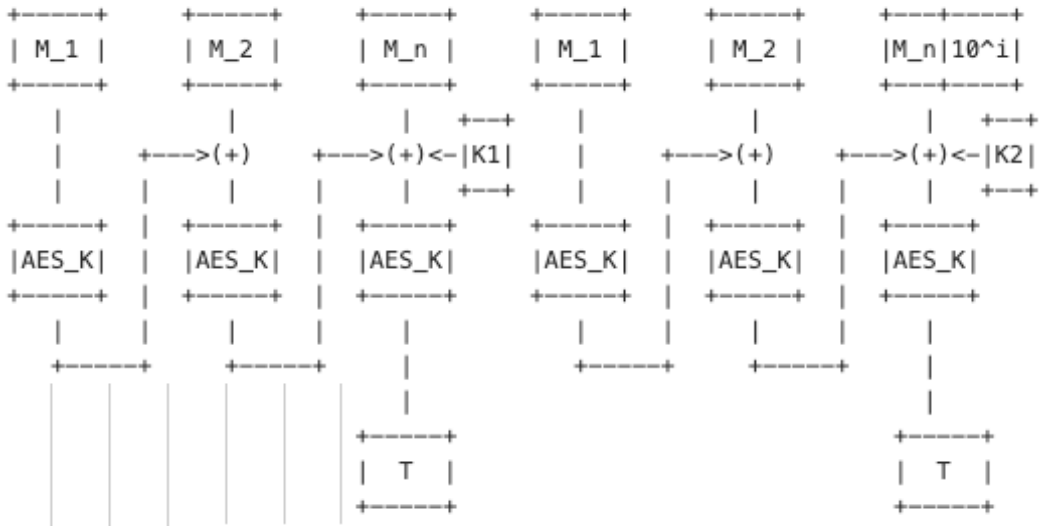
图/Figure 7　轮密钥加-k=1~9/AddRoundKey-k=1~9

c) 轮次=10时，将轮次10行位移的结果Sc[]与W[40]~W[43]进行异或运算，并将运算结果赋给CR[]，即为最终的结算结果，如图8所示。When the round = 10, the result of the displacement of the 10 rows of the round Sc[] is XOR operated with W[40]~W[43], and the result is assigned to CR[], which is the final settlement result, as shown in Figure 8.



图/Figure 8　轮密钥加-k=10/AddRoundKey-k=10

4.1.3 CMAC 计算/CMAC calculations



图/Figure 9　AES-CMAC 的两种情况/ Illustration of the two cases of AES-CMAC

AES_K是AES-128的密钥，消息M被分成块M_1，…M_n，其中M_i是第i个消息块。M_i的长度为16 bytes/128 bits，当i = 1，…，n-1，并且最后一个块M_n的长度小于或等于16 bytes/128 bits。K1是情况(a)的子密钥，K2是情况(b)的子密钥。K1和K2是通过子密钥生成算法生成的。如果计算的消息分组最后一个数据块是16 bytes/128 bits，则最后一个数据块和K1进行计算；如果最后一个数据块不足16 bytes/128 bits，则要先进行填充，填充方法为先添加1bit的1，其余bit填充0，直到填充的数据块达到16 bytes/128 bits。K1和K2参与最后一个数据块的计算。

AES_K is the key of AES-128 and message M is divided into chunks M_1,... M_n, where M_i is the ith message block. The length of the M_i is 16 bytes /128 bits when i = 1,...,n-1 and the length of the last block M_n is less than or equal to 16 bytes/128 bits. K1 is the sub-key of case (a) and K2 is the sub-key of case (b). K1 and K2 are generated by a sub-key generation algorithm. If the last data block of the calculated message group is 16 bytes/128 bits, the last data block is calculated with K1. If the last data block is less than 16 bytes/128 bits, it needs to be filled first, by adding 1 bit of 1 bit first, and the rest of the bits are

filled with 0 until the filled data block reaches 16 bytes/128 bits. K1 and K2 are involved in the computation of the last data block.

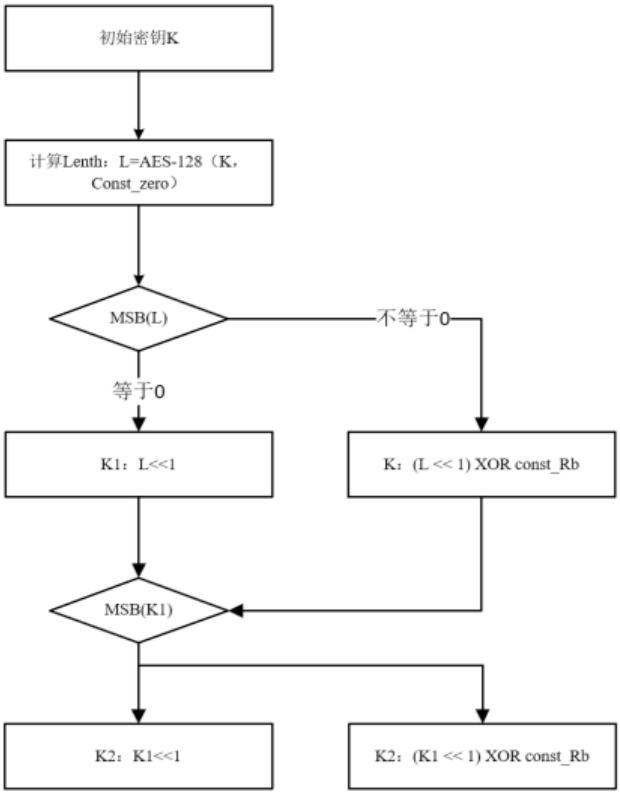
1） 生成子密钥/Generate Subkeys

在ECU中存储的16 bytes/128 bits的初始密钥K，通过子密钥生成算法生成两个16 bytes/128 bits的子密钥，初始定义两个数值块：const_Zero：0x00000000000000000000000000000000和const_Rb：0x00000000000000000000000000000087。

The initial 16 bytes/128 bits key K stored in the ECU generates two 16 bytes/128 bits subkeys through the sub-key generation algorithm, and initially defines two numerical blocks: const_Zero:0x00000000000000000000000000000000 and const_Rb:0x00000000000000000000000000000087.

首先用密钥K对全零输入块通过AES-128加密计算得到L，如果MSB(L)即L的最高有效位为0，则K1=L左移1位，否则K1等于const_Rb与左移1位的L异或；如果MSB(K1)最高有效位为0，则K2=K1左移1位，否则K2等于const_Rb与左移1位的K1异或。

Firstly, L is calculated by AES-128 encryption with key K for the all-zero input block, if MSB(L), that is, the most significant bit of L is 0, then K1=L is shifted to the left by 1 bit, otherwise K1 is equal to const_Rb L is XOR with the left shift of 1 bit; If the most significant bit of MSB(K1) is 0, then K2 = K1 is shifted 1 bit to the left, otherwise K2 is equal to const_Rb is XOR with K1 shifted 1 bit to the left.



图/Figure 10　子密钥生成/ Subkey generation

2） 生成MAC值/Generate MAC value

MAC 生成算法 AES-CMAC（） 采用三个输入、一个密钥、一条消息和消息长度（以八位字节为单位）。密钥（用 K 表示）就是 AES-128 的密钥。消息及其长度（以八位字节为单位）分别用 M 和 len 表示。消息 M 由 M_i 序列表示，其中 M_i 是第 i 个消息块。也就是说，如果 M 由 n 个块组成，则 M 写为M = M_1 ||M_2 ||... ||M_{n-1} ||M_n。

The MAC generation algorithm AES-CMAC() takes three inputs, a key, a message, and the message length (in octet). The key (denoted by K) is the key to AES-128. Messages and their lengths (in octet) are denoted by M and len, respectively. Message M is represented by a M_i sequence, where M_i is the i-th message block. That is, if M consists of n blocks, then M is written as M = M_1 ||M_2 ||... ||M_{n-1} ||M_n。

MAC生成算法的输出是一个 16 bytes/128 bits字符串，用于验证输入消息。MAC 用 T 表示，T ：= AES-CMAC（K，M，len）。

The output of the MAC generation algorithm is a 16 bytes/128 bits string that validates the input message. MAC is denoted by T，T ：= AES-CMAC（K，M，len）。

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                              +
+    Input          : K          ( 128-bit key )                               +
+                   : M          ( message to be authenticated )               +
+                   : len   ( length of the message in octets )                +
+    Output         : T          ( message authentication code )               +
+                                                                              +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+    Constants: const_Zero is 0x00000000000000000000000000000000              +
+                       const_Bsize is 16                                      +
+                                                                              +
+    Variables: K1, K2 for 128-bit subkeys                                     +
+                       M_i is the i-th block (i=1..ceil(len/const_Bsize))     +
+                       M_last is the last block xor-ed with K1 or K2          +
+                              n             for number of blocks to be processed +
+                              r             for number of octets of last block +
+                       flag          for denoting if last block is complete or not +
+                                                                              +
+    Step 1.  (K1,K2) := Generate_Subkey(K);                                   +
+    Step 2.  n := ceil(len/const_Bsize);                                      +
+    Step 3.  if n = 0                                                         +
+                       then                                                   +
+                              n := 1;                                         +
+                              flag := false;                                  +
+                       else                                                   +
+                              if len mod const_Bsize is 0                     +
+                              then flag := true;                              +
+                              else flag := false;                             +
+                                                                              +
+    Step 4.  if flag is true                                                  +
+                       then M_last := M_n XOR K1;                             +
+                       else M_last := padding(M_n) XOR K2;                    +
+    Step 5.  X := const_Zero;                                                 +
+    Step 6.  for i := 1 to n-1 do                                             +
+                              begin                                           +
+                                 Y := X XOR M_i;                              +
+                                 X := AES-128(K,Y);                           +
+                              end                                             +
+                       Y := M_last XOR X;                                     +
+                       T := AES-128(K,Y);                                     +
+    Step 7.  return T;                                                        +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

步骤1，子密钥K1和K2是通过子密钥生成算法从初始密钥K计算生成的。

In step 1, the sub-keys K1 and K2 are generated from the initial key K calculation by the sub-key generation algorithm.

步骤2，计算区块数n。块数是大于或等于通过将 length 参数除以块长度16个八位字节确定的商的最小整数值。

Step 2: Calculate the number of blocks n. The number of blocks is greater than or equal to the smallest integer value of the quotient determined by dividing the length parameter by the block length of 16 octets.

步骤3，检查输入消息的长度。如果输入长度为0（null），则要处理的块数应为1，并且标志应标记为 not-complete-block （false）。否则，如果最后一个块长度为16 bytes/128 bits位，则该标志将标记为 complete-block（true）;否则，将标志标记为 not-complete-block （false）。

Step 3, check the length of the input message. If the input length is 0 (null), the number of blocks to be processed should be 1 and the flag should be marked as not-complete-block (false). Otherwise, if the last block is 16 bytes/128 bits long, the flag will be marked as complete-block(true); Otherwise, mark the flag as not-complete-block (false).

步骤4，M_last 是通过对M_n 和之前计算的子密钥之一进行异或计算的。如果最后一个区块是完整区块 （true），则M_last是M_n和K1的异或。否则，M_last是padding（M_n）和K2的异或。

Step 4, M_last is XOR by XOR calculation of M_n and one of the previously computed subkeys. If the last block is a full block (true), then the M_last is XOR for M_n and K1. Otherwise, M_last is POR (M_n) and K2. Step 5, initialize variable X, which is used for subsequent calculations.

步骤5，初始化变量 X，用于后续的计算。

Step 5, initialize variable X, which is used for subsequent calculations.

步骤6，基于CBC-MAC 应用于 M_1,...,M_{n-1}，M_last。

Step 6, based on CBC-MAC applied to M_1,...,M_{n-1}, M_last.

步骤7，返回16 bytes/128 bits MAC。

STEP 7, GO BACK TO THE 16 bytes/128 bits MAC.

4.2 算法示例/Algorithm Examples

表3给出了2组不同长度的输入，密钥和计算结果。

Table 3 shows 2 sets of inputs, keys and calculation results of different lengths.

表/Table 3 计算结果示例/Examples of Calculation Results

序号	输入（Hex）	密钥（Hex）	计算结果（Hex）
1	6bc1bee22e409f96e93d7e117393172a	2b7e151628aed2a6abf7158809cf4f3c	070a16b46b4d4144f79bdd9dd04a287c
2	6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411	2b7e151628aed2a6abf7158809cf4f3c	dfa66747de9ae63030ca32611497c827