密　级	
编　号	

浙江大學

Zhejiang University

网络与通信安全

AES 加解密和 AES-MAC 的 C 语言实现

C language implementation of AES-MAC and AES

学　院　信息与电子工程学院
专　业　信息工程
姓　名
学　号　3200105426

2023 年 6 月 14 日

目 录

第 1 章 题目要求

1.1 题目

用 MATLAB 或 C 实现加解密算法及其应用：AES 和基于 AES 的 CMAC。

1.2 要求

1. 加密密钥：自己的姓名汉字用 UNICODE 编码后扩展成需要的位数；

2. 加密与认证消息：

Information Security is a multidisciplinaryarea of study and professional activity which is concerned with the developmentand implementation of security mechanisms of all available types to keepinformation in all its locations and, consequently, information systems, whereinformation is created, processed, stored, transmitted and destroyed, free fromthreats. This project is finished by ZHANGQINGMING

第 2 章 AES 算法原理

2.1 AES 总体结构

AES 加密密钥的长度可以使用 128 位、192 位或 256 位。密钥的长度不同，推荐加密轮数也不同，如下表所示：

表 2-1 密钥长度与推荐加密轮数

AES	密钥长度	分组长度（32bit）	加密轮数
AES-128	128	4	10
AES-192	192	4	12
AES-256	256	4	14

以 AES-128 为例进行说明。AES 的整体结构如图 2-1 所示，输入明文首先经过一个初始变换再进行 10 轮的循环运算。其中初始变换就是轮密钥加，每一轮循环运算包括了字节代换、行移位、列混淆、轮密钥加。AES 解密过程即对加密过程做一个逆运算即可。下面将具体分析字节代换、行移位、列混淆、轮密钥加过程。

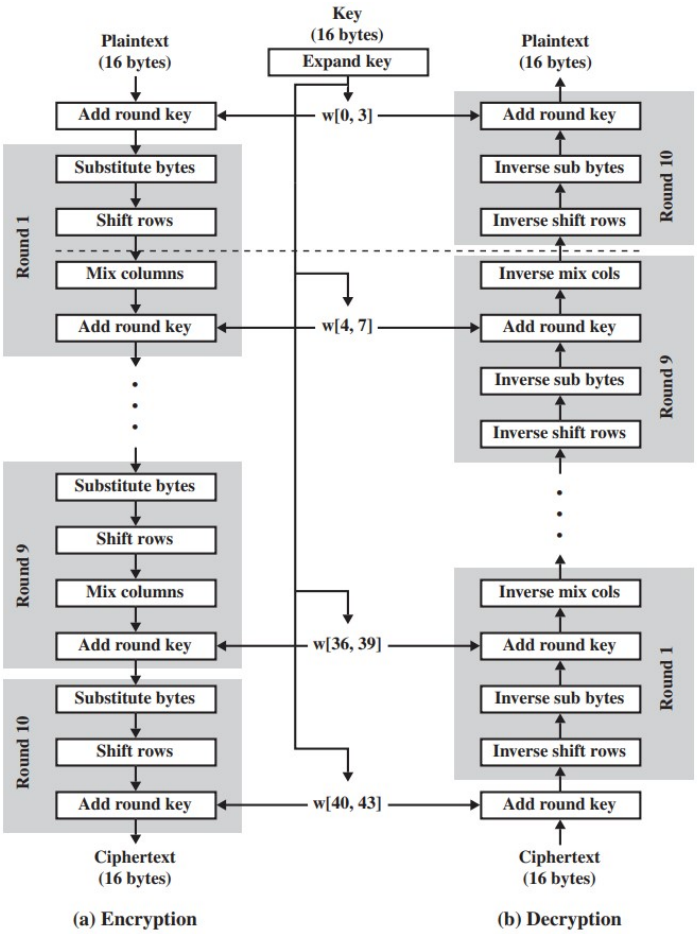


图 2-1 AES 加密和解密

2.2 字节代换

字节代换即 SubBytes，是一种简单的表查找如图 2-2 所示。AES 定义了一个字节值矩阵，称为 S 盒，其中包含所有可能的 256 个 8 位值的排列。State 的每个单独字节都以以下方式映射到一个新字节：字节最左边的 4 位用作行值，最右边的 4 位用于列值。这些行和列值用作 S 框的索引，以选择唯一的 8 位输出值。

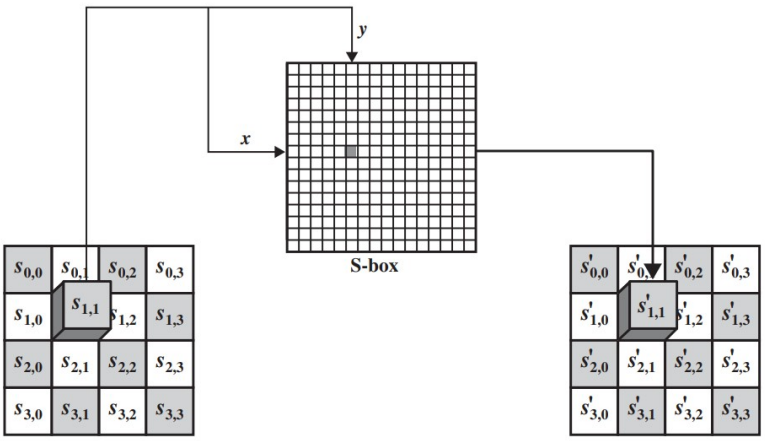


图 2-2 字节代换

2.3 行移位变换

行移位变换即 ShiftRows，如图 2-3 所示。State 的第一行没有改变。对于第二行，执行 1 字节的循环左移位。对于第三行，执行 2 字节的循环左移位。对于第四行，执行 3 字节的循环左移位。

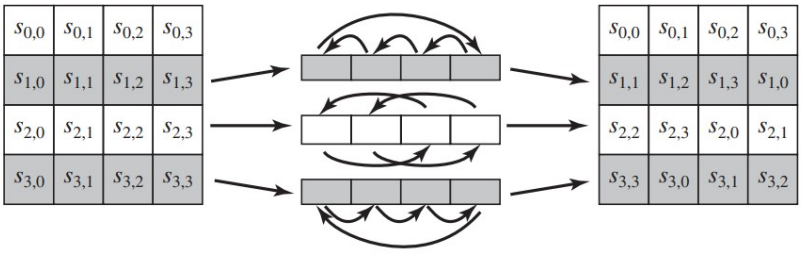


图 2-3 行移位

2.4 列混淆

列混淆即 MixColumns，对每个列单独操作。列的每个字节都映射到一个新值中，该值是该列中所有四个字节的函数。变换可以通过以下状态上的矩阵乘法来定义

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s'_{0,2} & s_{0,3} \\ s'_{1,0} & s_{1,1} & s'_{1,2} & s'_{1,3} \\ s_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{2-1}$$

乘积矩阵中的每个元素是一行和一列元素的乘积之和。在这里乘法和加法都是定义在 GF(2^8) 上的，状态中单列的列混淆变换表示为：

$$
\begin{aligned}
s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\
s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdots_{2,j}) \oplus s_{3,j} \\
s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdots_{3,j}) \\
s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})
\end{aligned}
\tag{2-2}
$$

2.5 轮密钥加

这个操作相对简单，其依据的原理是“任何数和自身的异或结果为 0”。加密过程中，每轮的输入与轮子密钥异或一次；因此，解密时再异或上该轮的轮子密钥即可恢复。下面是轮密钥加的一个例子：

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

图 2-4 轮密钥加实例

2.6 密钥扩展算法

密钥扩展的原理如图 2-5 所示。

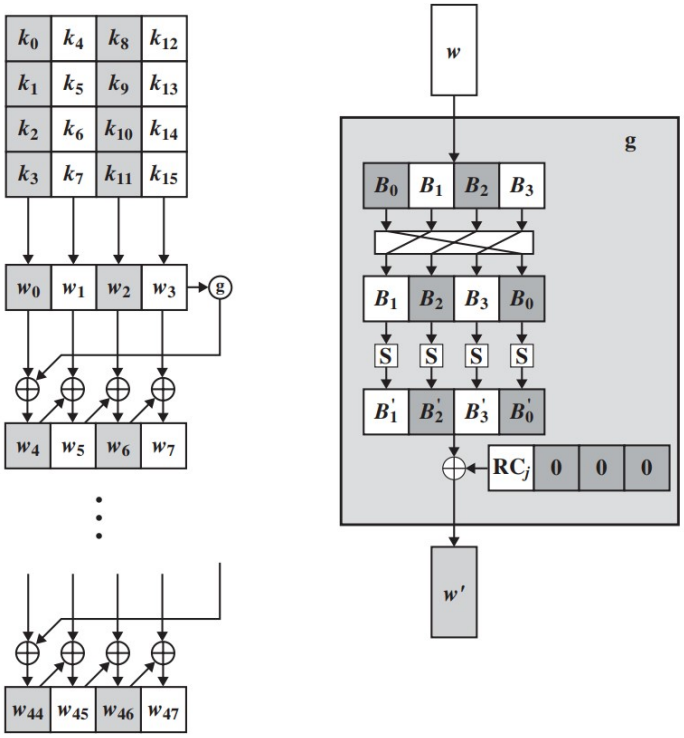


图 2-5 AES 密钥扩展

密钥扩展过程具体说明如下：

1. 将种子密钥按右边的格式排列，其中 k_0、k_1、...、k_{15} 依次表示种子密钥的一个字节；排列后用 4 个 32 比特的字表示，分别记为 $w[0]$、$w[1]$、$w[2]$、$w[3]$；

2. 按照如下方式，依次求解 $w[j]$，其中 j 是整数并且属于 $[4,43]$；

3. 若 $j=0(\mathrm{mod}\,4)$, 则 $w[j]=w[j-4]\oplus g(w[j-1])$, 否则 $w[j]=w[j-4]\oplus w[j-1]$；

函数 g 流程具体说明如下：

1. 将 w 循环左移 8 比特；

2. 分别对每个字节做 S 盒置换；

3. 与 32 比特的常量（$RC[\frac{j}{4}]$,0,0,0）进行异或，RC 是一个一维数组，其值如下。RC 的值只需要有 10 个，而此处用了 11 个，实际上 $RC[0]$ 在运算中没有用到，增加 $RC[0]$ 是为了便于程序中用数组表示。由于 j 的最小取值是 4，$\frac{j}{4}$ 的最小取值则是 1，因此不会产生错误。$RC=\{0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1B,0x36\}$。

第 3 章 AES 算法实现

3.1 密钥扩展实现

该函数实现密钥的扩展。首先将原始密钥拷贝到密钥调度字节表 $w[]$，剩余的 $w[]$ 首先进行一个循环左移一个字节, ; 再分别对每个字节按 S 盒进行映射; 再将其与 32bits 的常量进行异或。如此循环即可实现密钥的扩展，实现伪代码如下:

算法 3-1 Algorithm KeyExpansion

输入: $key[4 \cdot N_k], w[Nb \cdot (N_r + 1)], N_k$

word temp $i = 0$
repeat
  $w[i] = word(Key[4], Key[4 \cdot i + 1], Key[4 \cdot *i + 2], Key[4 \cdot i + 3])$
  $i = i + 1$
until $i < N_k$
$i = N_k$
repeat
  temp = $w[i - 1]$
  if $i = 0(mod\ N_k)$ **then**
    temp = SubByte(RotByte(temp)) $\oplus$ Rcon$[i/N_k]$
  else if $N_k > 6\ and\ i = 4(mod\ N_k)$ **then**
    temp = SubWord(temp)
  end if
  $w[i] = w[i - N_k] \oplus$ temp
until $i < N_b\ (Nr + 1)$

输出: w

3.2 AES 加解密实现

首先定义以下变量，则 AES 的加解密算法伪代码如下:

1. State：对明文加密的中间状态, 是一个 $4 \cdot N_b$ 的矩阵，初始状态等于输入;
2. CipherKey: 密钥, 可以看成一个 $4 \cdot N_k$ 的矩阵;
3. N_b：State 矩阵的列数，$N_b = 4$;
4. N_k：密钥字长 (1 字 =32bits)，$N_k = 4, 6, 8$;
5. N_r：加密的轮数，$N_r = 10, 12, 14$;
6. RoundKey：由 CipherKey 产生，每一轮加密计算对应一个 RoundKey;
7. KeyExpansion：将 CipherKey 转为 RoundKey;
8. AddRoundKey：使用 RoundKey 对 State 进行加密计算 (RoundKey 和 State 异或操作);
9. SubBytes：使用非线性字节替换表 sbox 对 State 进行替换操作;

10. ShiftRows：对 State 矩阵后三行进行偏移操作；

11. MixColumns：对 State 矩阵的所有列进行列混合操作；

算法 3-2 Algorithm AES_Encode

输入： input, K(128-bit key)

Initializatio: state = input
AddRoundkey(state, $K[0,\ N_b - 1]$)
round = 1
repeat
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundkey(state, $K[\text{round} \cdot N_b,\ (\text{round} + 1) \cdot N_b - 1]$)
    round = round + 1
until round = $N_r - 1$
SubBytes(state)
ShiftRows(state)
AddRoundkey(state, $K[N_r \cdot N_b,\ (\text{round} + 1) \cdot N_b - 1]$)

输出： state

算法 3-3 Algorithm AES_Decode

输入： input, K(128-bit key)

Initializatio: state = input
AddRoundkey(state, $K[N_r \cdot N_b,\ (\text{round} + 1) \cdot N_b - 1]$)
round = $N_r - 1$
repeat
    InvShiftRows(state)
    InvSubBytes(state)
    AddRoundkey(state, $K[\mathit{rounde} \cdot N_b,\ (\text{round} + 1) \cdot N_b - 1]$)
    InvMixColumns(state)
    round = round − 1
until round = $N_r - 1$
SubBytes(state)
ShiftRows(state)
AddRoundkey(state, $K[0,\ N_b - 1]$)

输出： state

第 4 章 基于 AES 的 CMAC 算法原理

为简单起见,我们以 AES-128 进行原理说明。首先将明文等长分为 n 组 $M_1, M_2, ..., M_n$，每组长度为 b bit，对第一组明文进行以密钥 K 的 AES 加密得到 C_1，再用 C_1 与 M_2 进行异或，将异或得到结果在进行 AES 加密得到 C_2，如此循环直到得到 C_{n-1}，C_{n-1} 与 M_n 与 K_1 进行异或，再将得到结果进行 AES 加密得到 C_n，取 C_n 前 $Tlen$ 个 bit 即得到消息认证码 T。即进行以下数学运算:

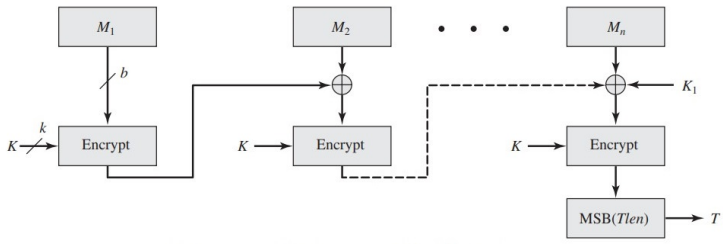
$$\begin{aligned} C_1 &= E\left(K, M_1\right) \\ C_2 &= E\left(K, [M_2 \oplus C_1]\right) \\ C_3 &= E\left(K, [M_3 \oplus C_2]\right) \\ &\cdots \\ C_n &= E\left(K, [M_n \oplus C_{n-1} \oplus K_1]\right) \\ T &= \mathrm{MSB}_{Tlen}(C_n) \end{aligned} \tag{4-1}$$

其中 K_1 和 K_2 的生成由以下公式决定:

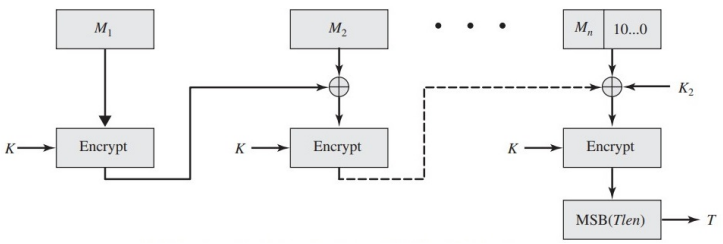
$$\begin{aligned} L &= E(K, 0^n) \\ K_1 &= L \cdot x \\ K_2 &= L \cdot x^2 = (L \cdot x) \cdot x \end{aligned} \tag{4-2}$$

其中乘法表示在 $\mathrm{GF}(2^n)$ 域上的乘法。故基于 AES 的 CMAC 算法的核心还是在于 AES 的计算，利用一个分组循环的思想，得到最后的消息认真码。

基于 AES 的 CMAC 算法框图如图 4-1 所示，其中图 (a) 为消息长度是分组长度的整数倍情况，图 (b) 是消息长度不是分组长度整数倍的情况。



(a) 消息长度是分组长度的整数倍



(b) 消息长度不是分组长度的整数倍

图 4-1 基于 AES 的 CMAC

第 5 章 基于 AES 的 CMAC 实现

5.1 子密钥生成实现

函数 Generate_subkey 输入一个密钥 K 产生子密钥 K_1 或者 K_2。其中 K_1 用于最后一个块的长度等于块长度的情况。K_2 用于最后一个块的长度小于块长度的情况。算法流程如下：

1. 用 K 对一个全零的块进行 AES 加密得到 L；
2. K_1 由以下步骤生成：如果 L 的最高有效位等于 0，则 K_1 是 L 左移 1 位；否则，K_1 是 const_Rb 与 L 左移 1 位的异或；
3. K_2 由以下步骤生成：如果 K_1 的最高有效位等于 0，则 K_2 是 K_1 的左移 1 位；否则，K_2 是 const_Rb 与 K_1 左移 1 位的异或；
4. 返回 K_1 和 K_2；

算法 5-4 Algorithm Generate_Subkey

输入： K(128-bit key)

Initializatio:const_Zeros = 0(128bit),const_Rb = $0x87$(128bit)
L = AES − 128(K, const_Zero)
n = ceil(len/const_Bsize)
if MSB(L) = 0 **then**
    $K_1 = L << 1$
else
    $K_1 = (L << 1) \oplus$ const_Rb
end if
if MSB(K_1) = 0 **then**
    $K_2 = K_1 << 1$
else
    $K_2 = (K_1 << 1) \oplus$ const_Rb
end if

输出： K_1,K_2

5.2 MAC 消息认真码生成实现

MAC 生成算法 AES − CMAC 输入一个密钥、一条消息和以八位字节为单位的消息长度。消息及其以八位字节为单位的长度分别用 M 和 len 表示。消息 M 由 M_i 的序列表示，其中 M_i 是第 i 个消息块。即如果 M 由 n 个块组成，那么 M 可以写成：$M = M_1, M_2, ..., M_{n-1}, M_n$，其中每一个块 M_i 的长度为 128bit，$i = 1, ..., n-1$，最后一个块 M_n 长度小于等于 128bit。MAC 生成算法的输出是一个 128 位的字符串，称为 MAC，也即用于验证输入消息的消息认证码，用 T 表示。当消息长度不能被块长度整除时，需要在最后一个块后先添加一个 1，再一直添 0 直到长度为 128bit。具体算法流程如下：

1. 首先生成子密钥 K_1 和 K_2；
2. 用消息长度 len 除以块长度 const_Bsize，取大于商的最小整数作为分组个数 n；
3. 检查 n 的值，若 n 等于 0，则令 n 为 1，令 $flag$ 为 false。若 n 不为 0，则检查消息长度除以块长度的余数，若余数为 0，则令 $flag$ 为 ture；若余数不为 0，则令 $flag$ 为 false；其中 $flag$ 为真则用子密钥 K_1 进行异或，$flag$ 为假则用子密钥 K_2 进行异或；
4. 计算 M_n，$flag$ 为真则与子密钥 K_1 进行异或，$flag$ 为假则先进行填充，再与子密钥 K_2 进行异或；
5. 初始化变量 X；
6. 对 $M_1, M_2, ..., M_n$ 循环执行异或和 AES 加密操作；
7. 取 128 位的消息认证码，返回 T；

算法 5-5 Algorithm AES-CMAC

输入： K(128-bit key),M(message to be authenticated),len(length of the message in octets)

Initializatio:const_Zeros = $0(128\text{bit})$,const_Bsize = 16
(K_1, K_2) = Generate_Subkey(K)
n = ceil$(len/\text{const_Bsize})$
if $N = 0$ **then**
    $n = 1$
    $flag$ = false
else if $len = 0 mod(\text{const})$ **then**
    $flag$ = ture
else
    $flag$ = false
end if
if $flag$ = true **then**
    $M\_last = M_n \oplus K_1$
else
    M_last = padding$(M_n) \oplus K_2$
    X = const_Zero
end if
repeat
    $i = 1$
    $Y = X \oplus M_i$
    $X = \text{AES} - 128(K, Y)$
until $i = n - 1$
$Y = M\_last \oplus X$
$T = \text{AES} - 128(K, Y)$

输出： T

第 6 章 实验结果

6.1 AES 加密

首先生成密钥，将姓名转化为 UNICODE 编码得到：5f20 9752 94ed，在其后添加 0 作为扩展，可以得到密钥 K 为：5f20 9752 94ed 0000 0000 0000 0000 0000。

消息：

Information Security is a multidisciplinaryarea of study and professional activity which is concerned with the developmentand implementation of security mechanisms of all available types to keepinformation in all its locations and, consequently, information systems, whereinformation is created, processed, stored, transmitted and destroyed, free fromthreats. This project is finished by ZHANGQINGMING

对应 ASCII 码：

49 6e 66 6f 72 6d 61 74 69 6f 6e 20 53 65 63 75 72 69 74 79 20 69 73 20 61 20 6d 75 6c 74 69 64 69 73 63 69 70 6c 69 6e 61 72 79 61 72 65 61 20 6f 66 20 73 74 75 64 79 20 61 6e 64 20 70 72 6f 66 65 73 73 69 6f 6e 61 6c 20 61 63 74 69 76 69 74 79 20 77 68 69 63 68 20 69 73 20 63 6f 6e 63 65 72 6e 65 64 20 77 69 74 68 20 74 68 65 20 64 65 76 65 6c 6f 70 6d 65 6e 74 61 6e 64 20 69 6d 70 6c 65 6d 65 6e 74 61 74 69 6f 6e 20 6f 66 20 73 65 63 75 72 69 74 79 20 6d 65 63 68 61 6e 69 73 6d 73 20 6f 66 20 61 6c 6c 20 61 76 61 69 6c 61 62 6c 65 20 74 79 70 65 73 20 74 6f 20 6b 65 65 70 69 6e 66 6f 72 6d 61 74 69 6f 6e 20 69 6e 20 61 6c 6c 20 69 74 73 20 6c 6f 63 61 74 69 6f 6e 73 20 61 6e 64 2c 20 63 6f 6e 73 65 71 75 65 6e 74 6c 79 2c 20 69 6e 66 6f 72 6d 61 74 69 6f 6e 20 73 79 73 74 65 6d 73 2c 20 77 68 65 72 65 69 6e 66 6f 72 6d 61 74 69 6f 6e 20 69 73 20 63 72 65 61 74 65 64 2c 20 70 72 6f 63 65 73 73 65 64 2c 20 73 74 6f 72 65 64 2c 20 74 72 61 6e 73 6d 69 74 74 65 64 20 61 6e 64 20 64 65 73 74 72 6f 79 65 64 2c 20 66 72 65 65 20 66 72 6f 6d 74 68 72 65 61 74 73 2e 20 54 68 69 73 20 70 72 6f 6a 65 63 74 20 69 73 20 66 69 6e 69 73 68 65 64 20 62 79 20 5a 48 41 4e 47 51 49 4e 47 4d 49 4e 47

对消息进行加密得到结果消息认证码为：

1D0604C6A0CDC4287644B5D4D81A2901。