SM4 **基础实现** +SIMD+T-table

2022年7月30日

学 校: 山东大学

学院: 网络空间安全学院 (研究院)

姓 名: 张起萌 202000460118

目录

1	SM4	基础实现	3
	1.1	实验目的	3
	1.2	实验过程	3
	1.3	实验结果	4
2		D+T-table 实现	5
	2.1	实验目的	
	2.2	实验过程	1
	2.3	实验结果	6
3	参考	文献 1	LC
	3.1	基础实现	10
	3.2	SIMD+T-table 实现	LC
4	附录	1	LC
	4.1	SM4 基础代码实现	10
	4.2	SIMD+T-table 代码实现	

1 SM4 基础实现

1.1 实验目的

实现 SM4 基础算法。

1.2 实验过程

• 首先根据 SM4 算法描述, SM4 算法对 4 个输入字 (128bit) 进行操作, SM4 算法需要 32 次轮 迭代和 1 次反序变换。首先根据密钥生成算法生成每一轮的轮密钥。

密钥生成算法: 将初始密钥 MK 和 FK 的每个字 (32bit) 分别进行异或得到 K。

接着利用公式

$$rk_{i} = K_{i+4} = K_{i} \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_{i}), i = 0, 1, \dots, 31$$

计算轮密钥。其中 T2 和加密算法中合成置换 T1 相同。

• 得到轮密钥后,对明文字进行加密。利用 F 函数,输入四个字和每轮的轮密钥得到一个新的字,迭代 32 轮,最终得到 36 个字,将后四个字反序输出。其中每一轮的 F 函数根据公式

$$F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i)$$

计算得到。

在上述加密函数中引用了 T 函数, T 函数主要包括非线性变化 t 和线性变换 L 两个过程。
 非线性变化 t 是指对输入的每一个字节进行过 S-box 运算。线性变换 L 是指将过 S-box 后得到的结果循环左移, 左移公式如下:

```
/***** 非线性变换t函数实现*******/
   string NLTransform(string str) {
           string res = "";
4
5
           for (int i = 0; i < 4; i++) {
                  res = res + Sbox[HexToDec(str[2 * i])][HexToDec(str[2 * i + 1])];
6
           }
           return res;
  }
10
   /****** 用于加解密算法中的合成置换T函数实现*******/
11
  string T1(string str)
12
13
       string str1 = "";
14
15
       string str2 = "";
16
       str1 = NLTransform(str);
       str2 = XOR(XOR(XOR(XOR(Str1, LeftShift(str1, 2)), LeftShift(str1, 10)), LeftShift(str1, 18)), LeftShift
17
       return str2;
18
19
  }
```

1.3 实验结果

在做实验的过程中,我发现了一个很有趣的事情。我分别在 CodeBlocks 和 VS 上实现了该代码,但是它们的运行时间相差很多,但我并不知道为什么。在两个编译环境下的实验结果如下图:

```
■ DACodeBlocks\Tesf\bin\Debug\Test.exe

明文: 0123456789ABCDEFFEDCBA9876543210
加密密钥: 0123456789ABCDEFFEDCBA9876543210
密文: 681EDF34D206965E86B3E94F536E4246
解密密钥: 0123456789ABCDEFFEDCBA9876543210
明文: 0123456789ABCDEFFEDCBA9876543210
加解密时间: 8ms

Process returned 0 (0x0) execution time: 0.108 s

Press any key to continue.
```

图 1: CodeBlocks 上 SM4 基础实现

```
Microsoft Visual Studio 调试控制台

明 文: 0123456789ABCDEFFEDCBA9876543210

加密密钥: 0123456789ABCDEFFEDCBA9876543210

V密文: 681EDF34D206965E86B3E94F536E4246

即 文: 0123456789ABCDEFFEDCBA9876543210

即 文: 0123456789ABCDEFFEDCBA9876543210

a 加解密时间: 267ms

a C:\Uisers\86188\Source\Repos\Test\Debug\Test. exe (进程 8392)已退出,代码为 0。

思要在调试停止时自动关闭控制台,请启用"工具"->"选项"->"调试"->"调试停止时自动关闭控制台"。

e 按任意键关闭此窗口...
```

图 2: VS 上 SM4 基础实现

2 SIMD+T-table 实现

2.1 实验目的

最大限度加速 SM4 算法。

2.2 实验过程

首先构造 T-table, 将线性变换 L 和过 S-box 这两个过程合并为一个过程, 即直接过 4 个 256x32bit 的查找表 (T-table)。这里我将 T-tables 直接列出, 方便后续查找。

```
1 static const uint32_t Table0[] =
      {
 2
              0xD55B5B8E, 0x924242D0, 0xEAA7A74D, 0xFDFBFB06, 0xCF3333FC, 0xE2878765,
              0x3DF4F4C9, 0xB5DEDE6B, 0x1658584E, 0xB4DADA6E, 0x14505044, 0xC10B0BCA,
              0x28A0A088, 0xF8EFEF17, 0x2CB0B09C, 0x05141411, 0x2BACAC87, 0x669D9DFB,
              0x986A6AF2, 0x77D9D9AE, 0x2AA8A82, 0xBCFAFA46, 0x04101014, 0xC00F0FCF,
 6
              0xA8AAAA02, \ 0x45111154 \,, \ 0x134C4C5F, \ 0x269898BE \,, \ 0x4825256D \,, \ 0x841A1A9E \,,
              0 \\ x \\ 934646 \\ D5 \,, \ 0 \\ x \\ ECBFBF53 \,, \ 0 \\ x \\ 9A6262 \\ E8 \,, \ 0 \\ x \\ 7BE9E992 \,, \ 0 \\ x \\ 33CCCCFF \,, \ 0 \\ x \\ 55515104 \,, \\
              0x0B2C2C27, 0x420D0D4F, 0xEEB7B759, 0xCC3F3FF3, 0xAEB2B21C, 0x638989EA,
              0 \\ \\ x \\ E \\ 7939374 \;,\;\; 0 \\ x \\ B \\ 1 \\ C \\ E \\ C \\ E \\ F \\ ,\;\; 0 \\ x \\ 1 \\ C \\ 70706 \\ C \;,\;\; 0 \\ x \\ A \\ B \\ A \\ 6 \\ A \\ 60 \\ D \;,\;\; 0 \\ x \\ C \\ A \\ 2727 \\ E \\ D \;,\;\; 0 \\ x \\ 0 \\ 820202028 \;,\;\; 0 \\ x \\ 0 \\ x
11
              0xEBA3A348, 0x975656C1, 0x82020280, 0xDC7F7FA3, 0x965252C4, 0xF9EBEB12,
12
              0x74D5D5A1. 0x8D3E3EB3. 0x3FFCFCC3. 0xA49A9A3E. 0x461D1D5B. 0x071C1C1B.
13
              0xA59E9E3B, 0xFFF3F30C, 0xF0CFCF3F, 0x72CDCDBF, 0x175C5C4B, 0xB8EAEA52,
14
              0x810E0E8F, 0x5865653D, 0x3CF0F0CC, 0x1964647D, 0xE59B9B7E, 0x87161691,
15
              0x4E3D3D73, 0xAAA2A208, 0x69A1A1C8, 0x6AADADC7, 0x83060685, 0xB0CACA7A,
17
              0x70C5C5B5, 0x659191F4, 0xD96B6BB2, 0x892E2EA7, 0xFBE3E318, 0xE8AFAF47,
              0x0F3C3C33, 0x4A2D2D67, 0x71C1C1B0, 0x5759590E, 0x9F7676E9, 0x35D4D4E1,
18
              0x1E787866, 0x249090B4, 0x0E383836, 0x5F797926, 0x628D8DEF, 0x59616138,
19
              0xD2474795, 0xA08A8A2A, 0x259494B1, 0x228888AA, 0x7DF1F18C, 0x3BECECD7,
20
              0 \\ x \\ 01040405 \,, \ 0 \\ x \\ 218484A5 \,, \ 0 \\ x \\ 79E1E198 \,, \ 0 \\ x \\ 851E1E9B \,, \ 0 \\ x \\ D7535384 \,, \ 0 \\ x \\ 0000000000 \,,
21
              0x4719195E, 0x565D5D0B, 0x9D7E7EE3, 0xD04F4F9F, 0x279C9CBB, 0x5349491A,
              0x4D31317C, 0x36D8D8EE, 0x0208080A, 0xE49F9F7B, 0xA2828220, 0xC71313D4,
23
              0xCB2323E8, 0x9C7A7AE6, 0xE9ABAB42, 0xBDFEFE43, 0x882A2AA2, 0xD14B4B9A,
24
              0x41010140, 0xC41F1FDB, 0x38E0E0D8, 0xB7D6D661, 0xA18E8E2F, 0xF4DFDF2B,
25
              0xF1CBCB3A, 0xCD3B3BF6, 0xFAE7E71D, 0x608585E5, 0x15545441, 0xA3868625,
26
              0xE3838360, 0xACBABA16, 0x5C757529, 0xA6929234, 0x996E6EF7, 0x34D0D0E4,
27
              0x1A686872, 0x54555501, 0xAFB6B619, 0x914E4EDF, 0x32C8C8FA, 0x30C0C0F0,
              0xF6D7D721, 0x8E3232BC, 0xB3C6C675, 0xE08F8F6F, 0x1D747469, 0xF5DBDB2E,
29
              0xE18B8B6A, 0x2EB8B896, 0x800A0A8A, 0x679999FE, 0xC92B2BE2, 0x618181E0,
30
              0xC30303C0, 0x29A4A48D, 0x238C8CAF, 0xA9AEAE07, 0x0D343439, 0x524D4D1F,
31
32
              0x4F393976\,,\;\;0x6EBDBDD3,\;\;0xD6575781\,,\;\;0xD86F6FB7,\;\;0x37DCDCEB,\;\;0x44151551\,,
              0xDD7B7BA6, 0xFEF7F709, 0x8C3A3AB6, 0x2FBCBC93, 0x030C0C0F, 0xFCFFFF03,
              0x6BA9A9C2, \ 0x73C9C9BA, \ 0x6CB5B5D9, \ 0x6DB1B1DC, \ 0x5A6D6D37, \ 0x50454515\,,
              0x8F3636B9, 0x1B6C6C77, 0xADBEBE13, 0x904A4ADA, 0xB9EEEE57, 0xDE7777A9,
35
              0xBEF2F24C, 0x7EFDFD83, 0x11444455, 0xDA6767BD, 0x5D71712C, 0x40050545,
36
              0x1F7C7C63, 0x10404050, 0x5B696932, 0xDB6363B8, 0x0A282822, 0xC20707C5,
37
              0x31C4C4F5, 0x8A2222A8, 0xA7969631, 0xCE3737F9, 0x7AEDED97, 0xBFF6F649,
38
              0x2DB4B499, \ 0x75D1D1A4, \ 0xD3434390, \ 0x1248485A, \ 0xBAE2E258, \ 0xE6979771, \\
39
              0xB6D2D264, 0xB2C2C270, 0x8B2626AD, 0x68A5A5CD, 0x955E5ECB, 0x4B292962,
40
              0x0C30303C, 0x945A5ACE, 0x76DDDDAB, 0x7FF9F986, 0x649595F1, 0xBBE6E65D,
41
              0xF2C7C735, 0x0924242D, 0xC61717D1, 0x6FB9B9D6, 0xC51B1BDE, 0x86121294,
42
              0x18606078, 0xF3C3C330, 0x7CF5F589, 0xEFB3B35C, 0x3AE8E8D2, 0xDF7373AC,
43
              0x4C353579, 0x208080A0, 0x78E5E59D, 0xEDBBBB56, 0x5E7D7D23, 0x3EF8F8C6,
44
              0xD45F5F8B, 0xC82F2FE7, 0x39E4E4DD, 0x49212168
45
46
      };
      static const uint32_t Table1[] =
```

```
48
        0x5B5B8ED5, 0x4242D092, 0xA7A74DEA, 0xFBFB06FD, 0x3333FCCF, 0x878765E2,
49
        0xF4F4C93D, 0xDEDE6BB5, 0x58584E16, 0xDADA6EB4, 0x50504414, 0x0B0BCAC1,
50
        51
        0x6A6AF298, \ 0xD9D9AE77, \ 0xA8A8822A, \ 0xFAFA46BC, \ 0x10101404, \ 0x0F0FCFC0,
52
        0xAAAA02A8, 0x11115445, 0x4C4C5F13, 0x9898BE26, 0x25256D48, 0x1A1A9E84,
53
        0x18181E06\,,\ 0x6666FD9B\,,\ 0x7272EC9E\,,\ 0x09094A43\,,\ 0x41411051\,,\ 0xD3D324F7\,,
54
        0x4646D593, 0xBFBF53EC, 0x6262F89A, 0xE9E9927B, 0xCCCCFF33, 0x51510455,
55
56
        0x2C2C270B, 0x0D0D4F42, 0xB7B759EE, 0x3F3FF3CC, 0xB2B21CAE, 0x8989EA63,
        0x939374E7, 0xCECE7FB1, 0x70706C1C, 0xA6A60DAB, 0x2727EDCA, 0x20202808,
57
        0xA3A348EB, 0x5656C197, 0x02028082, 0x7F7FA3DC, 0x5252C496, 0xEBEB12F9,
58
        0xD5D5A174, 0x3E3EB38D, 0xFCFCC33F, 0x9A9A3EA4, 0x1D1D5B46, 0x1C1C1B07,
59
        0x9E9E3BA5, 0xF3F30CFF, 0xCFCF3FF0, 0xCDCDBF72, 0x5C5C4B17, 0xEAEA52B8,
60
61
        0x0E0E8F81, 0x65653D58, 0xF0F0CC3C, 0x64647D19, 0x9B9B7EE5, 0x16169187,
        0x3D3D734E, 0xA2A208AA, 0xA1A1C869, 0xADADC76A, 0x06068583, 0xCACA7AB0,
62
        0xC5C5B570, 0x9191F465, 0x6B6BB2D9, 0x2E2EA789, 0xE3E318FB, 0xAFAF47E8,
63
        0x3C3C330F, \ 0x2D2D674A, \ 0xC1C1B071, \ 0x59590E57, \ 0x7676E99F, \ 0xD4D4E135, \\
64
65
        0x7878661E, 0x9090B424, 0x3838360E, 0x7979265F, 0x8D8DEF62, 0x61613859,
        0x474795D2\,,\ 0x8A8A2AA0,\ 0x9494B125\,,\ 0x8888AA22\,,\ 0xF1F18C7D\,,\ 0xECECD73B,
66
        0x04040501, 0x8484A521, 0xE1E19879, 0x1E1E9B85, 0x535384D7, 0x00000000,
67
        0x19195E47, 0x5D5D0B56, 0x7E7EE39D, 0x4F4F9FD0, 0x9C9CBB27, 0x49491A53,
68
        0x31317C4D, 0xD8D8EE36, 0x08080A02, 0x9F9F7BE4, 0x828220A2, 0x1313D4C7,
69
        0x2323E8CB, 0x7A7AE69C, 0xABAB42E9, 0xFEFE43BD, 0x2A2AA288, 0x4B4B9AD1,
70
        0x01014041, 0x1F1FDBC4, 0xE0E0D838, 0xD6D661B7, 0x8E8E2FA1, 0xDFDF2BF4,
71
72
        0xCBCB3AF1, 0x3B3BF6CD, 0xE7E71DFA, 0x8585E560, 0x54544115, 0x868625A3,
        0x838360E3, 0xBABA16AC, 0x7575295C, 0x929234A6, 0x6E6EF799, 0xD0D0E434,
73
        0x6868721A, 0x55550154, 0xB6B619AF, 0x4E4EDF91, 0xC8C8FA32, 0xC0C0F030,
74
        0 \\ x \\ D7 \\ D721 \\ F6, \ 0 \\ x \\ 3232 \\ BC8E, \ 0 \\ x \\ C6C675 \\ B3, \ 0 \\ x \\ 8F8F6 \\ FE0, \ 0 \\ x \\ 7474691 \\ D, \ 0 \\ x \\ DB \\ DB2EF5,
75
        0x8B8B6AE1, 0xB8B8962E, 0x0A0A8A80, 0x9999FE67, 0x2B2BE2C9, 0x8181E061,
76
        0x0303C0C3, 0xA4A48D29, 0x8C8CAF23, 0xAEAE07A9, 0x3434390D, 0x4D4D1F52,
77
        0x3939764F, 0xBDBDD36E, 0x575781D6, 0x6F6FB7D8, 0xDCDCEB37, 0x15155144,
78
        0x7B7BA6DD, 0xF7F709FE, 0x3A3AB68C, 0xBCBC932F, 0x0C0C0F03, 0xFFFF03FC,
79
        0xA9A9C26B, 0xC9C9BA73, 0xB5B5D96C, 0xB1B1DC6D, 0x6D6D375A, 0x45451550,
80
        0x3636B98F, 0x6C6C771B, 0xBEBE13AD, 0x4A4ADA90, 0xEEEE57B9, 0x7777A9DE,
81
        0xF2F24CBE, \ 0xFDFD837E, \ 0x444445511 \,, \ 0x6767BDDA, \ 0x71712C5D \,, \ 0x05054540 \,,
82
        0x7C7C631F, 0x40405010, 0x6969325B, 0x6363B8DB, 0x2828220A, 0x0707C5C2,
83
        0xC4C4F531, 0x2222A88A, 0x969631A7, 0x3737F9CE, 0xEDED977A, 0xF6F649BF,
84
        0xB4B4992D, 0xD1D1A475, 0x434390D3, 0x48485A12, 0xE2E258BA, 0x979771E6,
85
        0xD2D264B6, 0xC2C270B2, 0x2626AD8B, 0xA5A5CD68, 0x5E5ECB95, 0x2929624B,
86
        0x30303C0C\,,\;\;0x5A5ACE94,\;\;0xDDDDAB76,\;\;0xF9F9867F\,,\;\;0x9595F164\,,\;\;0xE6E65DBB,
87
88
        0xC7C735F2, 0x24242D09, 0x1717D1C6, 0xB9B9D66F, 0x1B1BDEC5, 0x12129486,
        0x60607818, 0xC3C330F3, 0xF5F5897C, 0xB3B35CEF, 0xE8E8D23A, 0x7373ACDF,
89
        0x3535794C, 0x8080A020, 0xE5E59D78, 0xBBBB56ED, 0x7D7D235E, 0xF8F8C63E,
90
        0x5F5F8BD4, 0x2F2FE7C8, 0xE4E4DD39, 0x21216849
91
92
    };
    {\tt static \ const \ uint 32\_t \ Table 2 \, \lceil \, \rceil} \ = \\
93
94
        0x5B8ED55B, \ 0x42D09242\,, \ 0xA74DEAA7, \ 0xFB06FDFB, \ 0x33FCCF33\,, \ 0x8765E287\,,
95
        0xF4C93DF4, 0xDE6BB5DE, 0x584E1658, 0xDA6EB4DA, 0x50441450, 0x0BCAC10B,
96
97
        0xA08828A0, 0xEF17F8EF, 0xB09C2CB0, 0x14110514, 0xAC872BAC, 0x9DFB669D,
        0x6AF2986A, 0xD9AE77D9, 0xA8822AA8, 0xFA46BCFA, 0x10140410, 0x0FCFC00F,
98
        0xAA02A8AA, 0x11544511, 0x4C5F134C, 0x98BE2698, 0x256D4825, 0x1A9E841A,
99
        0x181E0618, 0x66FD9B66, 0x72EC9E72, 0x094A4309, 0x41105141, 0xD324F7D3,
00
        0x46D59346, 0xBF53ECBF, 0x62F89A62, 0xE9927BE9, 0xCCFF33CC, 0x51045551,
01
        0x2C270B2C, 0x0D4F420D, 0xB759EEB7, 0x3FF3CC3F, 0xB21CAEB2, 0x89EA6389,
02
        0x9374E793, 0xCE7FB1CE, 0x706C1C70, 0xA60DABA6, 0x27EDCA27, 0x20280820,
03
        0xA348EBA3, 0x56C19756, 0x02808202, 0x7FA3DC7F, 0x52C49652, 0xEB12F9EB,
04
        0xD5A174D5, \ 0x3EB38D3E, \ 0xFCC33FFC, \ 0x9A3EA49A, \ 0x1D5B461D, \ 0x1C1B071C, \\
05
        0x9E3BA59E, 0xF30CFFF3, 0xCF3FF0CF, 0xCDBF72CD, 0x5C4B175C, 0xEA52B8EA,
06
```

```
07
             0x0E8F810E, 0x653D5865, 0xF0CC3CF0, 0x647D1964, 0x9B7EE59B, 0x16918716,
.08
             0x3D734E3D, 0xA208AAA2, 0xA1C869A1, 0xADC76AAD, 0x06858306, 0xCA7AB0CA,
             0xC5B570C5, 0x91F46591, 0x6BB2D96B, 0x2EA7892E, 0xE318FBE3, 0xAF47E8AF,
.09
             0x3C330F3C\,,\ 0x2D674A2D\,,\ 0xC1B071C1\,,\ 0x590E5759\,,\ 0x76E99F76\,,\ 0xD4E135D4\,,
10
             0x78661E78\,,\ 0x90B42490\,,\ 0x38360E38\,,\ 0x79265F79\,,\ 0x8DEF628D\,,\ 0x61385961\,,
11
12
             0x4795D247, 0x8A2AA08A, 0x94B12594, 0x88AA2288, 0xF18C7DF1, 0xECD73BEC,
             0 \\ x 0 4 0 5 0 1 0 4 \;,\;\; 0 \\ x 8 4 A 5 2 1 8 4 \;,\;\; 0 \\ x E 1 9 8 7 9 E 1 \;,\;\; 0 \\ x 1 E 9 B 8 5 1 E \;,\;\; 0 \\ x 5 3 8 4 D 7 5 3 \;,\;\; 0 \\ x 0 0 0 0 0 0 0 0 0 0 \;,\;\;
13
             0x195E4719, 0x5D0B565D, 0x7EE39D7E, 0x4F9FD04F, 0x9CBB279C, 0x491A5349,
14
15
             0x317C4D31, 0xD8EE36D8, 0x080A0208, 0x9F7BE49F, 0x8220A282, 0x13D4C713,
             0x23E8CB23, 0x7AE69C7A, 0xAB42E9AB, 0xFE43BDFE, 0x2AA2882A, 0x4B9AD14B,
16
            0 \times 01404101, 0 \times 1FDBC41F, 0 \times E0D838E0, 0 \times D661B7D6, 0 \times 8E2FA18E, 0 \times DF2BF4DF,
17
             0xCB3AF1CB, 0x3BF6CD3B, 0xE71DFAE7, 0x85E56085, 0x54411554, 0x8625A386,
18
             0x8360E383, 0xBA16ACBA, 0x75295C75, 0x9234A692, 0x6EF7996E, 0xD0E434D0,
19
20
             0x68721A68, 0x55015455, 0xB619AFB6, 0x4EDF914E, 0xC8FA32C8, 0xC0F030C0,
             0xD721F6D7, 0x32BC8E32, 0xC675B3C6, 0x8F6FE08F, 0x74691D74, 0xDB2EF5DB,
21
             0x8B6AE18B, \ 0xB8962EB8, \ 0x0A8A800A, \ 0x99FE6799, \ 0x2BE2C92B, \ 0x81E06181, \\
22
             23
24
             0x39764F39, 0xBDD36EBD, 0x5781D657, 0x6FB7D86F, 0xDCEB37DC, 0x15514415,
             0x7BA6DD7B, \ 0xF709FEF7, \ 0x3AB68C3A, \ 0xBC932FBC, \ 0x0C0F030C, \ 0xFF03FCFF, \\
25
             0xA9C26BA9, 0xC9BA73C9, 0xB5D96CB5, 0xB1DC6DB1, 0x6D375A6D, 0x45155045,
26
             0x36B98F36, 0x6C771B6C, 0xBE13ADBE, 0x4ADA904A, 0xEE57B9EE, 0x77A9DE77,
27
             0xF24CBEF2, 0xFD837EFD, 0x44551144, 0x67BDDA67, 0x712C5D71, 0x05454005,
28
            0x7C631F7C, 0x40501040, 0x69325B69, 0x63B8DB63, 0x28220A28, 0x07C5C207,
29
             0xC4F531C4, 0x22A88A22, 0x9631A796, 0x37F9CE37, 0xED977AED, 0xF649BFF6,
30
31
             0xB4992DB4, 0xD1A475D1, 0x4390D343, 0x485A1248, 0xE258BAE2, 0x9771E697,
             0xD264B6D2, 0xC270B2C2, 0x26AD8B26, 0xA5CD68A5, 0x5ECB955E, 0x29624B29,
32
             0x303C0C30, 0x5ACE945A, 0xDDAB76DD, 0xF9867FF9, 0x95F16495, 0xE65DBBE6,
33
             0xC735F2C7, 0x242D0924, 0x17D1C617, 0xB9D66FB9, 0x1BDEC51B, 0x12948612,
34
             0 \\ x 60781860 \;,\;\; 0 \\ x \\ C330 \\ F3 \\ C3 \;,\;\; 0 \\ x \\ F5897 \\ CF5 \;,\;\; 0 \\ x \\ B35 \\ CEFB3 \;,\;\; 0 \\ x \\ E8D23 \\ AE8 \;,\;\; 0 \\ x73 \\ ACDF73 \;,\;\; 0 \\ x \\ E8D23 \\ AE8 \;,\;\; 0 \\ x \\ E8D23 \\ AE8 \;,\;\; 0 \\ x \\ E8D23 \\ AE8 \;,\;\; 0 \\ x \\ E8D23 \\ 
35
             0x35794C35, 0x80A02080, 0xE59D78E5, 0xBB56EDBB, 0x7D235E7D, 0xF8C63EF8,
36
             0x5F8BD45F, 0x2FE7C82F, 0xE4DD39E4, 0x21684921
37
38
      };
      static const uint32_t Table3[] =
39
40
             0x8ED55B5B, \ 0xD0924242\,, \ 0x4DEAA7A7, \ 0x06FDFBFB, \ 0xFCCF3333, \ 0x65E28787\,,
41
42
             0xC93DF4F4, 0x6BB5DEDE, 0x4E165858, 0x6EB4DADA, 0x44145050, 0xCAC10B0B,
             0x8828A0A0, 0x17F8EFEF, 0x9C2CB0B0, 0x11051414, 0x872BACAC, 0xFB669D9D,
43
             0xF2986A6A, 0xAE77D9D9, 0x822AA8A8, 0x46BCFAFA, 0x14041010, 0xCFC00F0F,
44
             0x02A8AAAA, 0x54451111, 0x5F134C4C, 0xBE269898, 0x6D482525, 0x9E841A1A,
45
             0x1E061818\,,\ 0xFD9B6666\,,\ 0xEC9E7272\,,\ 0x4A430909\,,\ 0x10514141\,,\ 0x24F7D3D3\,,
46
47
             0xD5934646, 0x53ECBFBF, 0xF89A6262, 0x927BE9E9, 0xFF33CCCC, 0x04555151,
             0x270B2C2C, 0x4F420D0D, 0x59EEB7B7, 0xF3CC3F3F, 0x1CAEB2B2, 0xEA638989,
48
             0x74E79393, 0x7FB1CECE, 0x6C1C7070, 0x0DABA6A6, 0xEDCA2727, 0x28082020,
49
             0x48EBA3A3, 0xC1975656, 0x80820202, 0xA3DC7F7F, 0xC4965252, 0x12F9EBEB,
.50
             0xA174D5D5, 0xB38D3E3E, 0xC33FFCFC, 0x3EA49A9A, 0x5B461D1D, 0x1B071C1C,
51
             0x3BA59E9E, 0x0CFFF3F3, 0x3FF0CFCF, 0xBF72CDCD, 0x4B175C5C, 0x52B8EAEA,
.52
            0x8F810E0E, \ 0x3D586565, \ 0xCC3CF0F0, \ 0x7D196464, \ 0x7EE59B9B, \ 0x91871616,
53
             0x734E3D3D, \ 0x08AAA2A2, \ 0xC869A1A1, \ 0xC76AADAD, \ 0x85830606, \ 0x7AB0CACA, \\
54
             0xB570C5C5, 0xF4659191, 0xB2D96B6B, 0xA7892E2E, 0x18FBE3E3, 0x47E8AFAF,
55
56
             0x330F3C3C, 0x674A2D2D, 0xB071C1C1, 0x0E575959, 0xE99F7676, 0xE135D4D4,
.57
             0x661E7878, 0xB4249090, 0x360E3838, 0x265F7979, 0xEF628D8D, 0x38596161,
             0x95D24747, 0x2AA08A8A, 0xB1259494, 0xAA228888, 0x8C7DF1F1, 0xD73BECEC,
58
            0x05010404, 0xA5218484, 0x9879E1E1, 0x9B851E1E, 0x84D75353, 0x00000000,
59
            0x5E471919, 0x0B565D5D, 0xE39D7E7E, 0x9FD04F4F, 0xBB279C9C, 0x1A534949,
60
             0x7C4D3131, 0xEE36D8D8, 0x0A020808, 0x7BE49F9F, 0x20A28282, 0xD4C71313,
61
             0xE8CB2323, 0xE69C7A7A, 0x42E9ABAB, 0x43BDFEFE, 0xA2882A2A, 0x9AD14B4B,
62
            0x40410101, 0xDBC41F1F, 0xD838E0E0, 0x61B7D6D6, 0x2FA18E8E, 0x2BF4DFDF,
63
             0x3AF1CBCB, \ 0xF6CD3B3B, \ 0x1DFAE7E7, \ 0xE5608585 \, , \ 0x41155454 \, , \ 0x25A38686 \, , \\
64
             0x60E38383, 0x16ACBABA, 0x295C7575, 0x34A69292, 0xF7996E6E, 0xE434D0D0,
65
```

```
0x721A6868, 0x01545555, 0x19AFB6B6, 0xDF914E4E, 0xFA32C8C8, 0xF030C0C0,
166
67
        0x21F6D7D7, 0xBC8E3232, 0x75B3C6C6, 0x6FE08F8F, 0x691D7474, 0x2EF5DBDB,
        0x6AE18B8B, 0x962EB8B8, 0x8A800A0A, 0xFE679999, 0xE2C92B2B, 0xE0618181,
68
        0xC0C30303, 0x8D29A4A4, 0xAF238C8C, 0x07A9AEAE, 0x390D3434, 0x1F524D4D,
69
        70
        0xA6DD7B7B, 0x09FEF7F7, 0xB68C3A3A, 0x932FBCBC, 0x0F030C0C, 0x03FCFFFF,
71
        0xC26BA9A9, \ 0xBA73C9C9, \ 0xD96CB5B5, \ 0xDC6DB1B1, \ 0x375A6D6D, \ 0x15504545 \ , \\
72
        0xB98F3636, 0x771B6C6C, 0x13ADBEBE, 0xDA904A4A, 0x57B9EEEE, 0xA9DE7777,
.73
74
        0x4CBEF2F2, 0x837EFDFD, 0x55114444, 0xBDDA6767, 0x2C5D7171, 0x45400505,
        0x631F7C7C, 0x50104040, 0x325B6969, 0xB8DB6363, 0x220A2828, 0xC5C20707,
75
        0xF531C4C4, 0xA88A2222, 0x31A79696, 0xF9CE3737, 0x977AEDED, 0x49BFF6F6,
76
        0x992DB4B4, 0xA475D1D1, 0x90D34343, 0x5A124848, 0x58BAE2E2, 0x71E69797,
77
        0x64B6D2D2, 0x70B2C2C2, 0xAD8B2626, 0xCD68A5A5, 0xCB955E5E, 0x624B2929,
78
79
        0x3C0C3030, 0xCE945A5A, 0xAB76DDDD, 0x867FF9F9, 0xF1649595, 0x5DBBE6E6,
        0x35F2C7C7, 0x2D092424, 0xD1C61717, 0xD66FB9B9, 0xDEC51B1B, 0x94861212,
80
        0x78186060, 0x30F3C3C3, 0x897CF5F5, 0x5CEFB3B3, 0xD23AE8E8, 0xACDF7373,
81
       0x794C3535\,,\;\;0xA0208080\,,\;\;0x9D78E5E5\,,\;\;0x56EDBBBB,\;\;0x235E7D7D\,,\;\;0xC63EF8F8\,,
82
       0x8BD45F5F, 0xE7C82F2F, 0xDD39E4E4, 0x68492121
83
84
   };
```

利用 SIMD 进行加速的实质是使其多线程化,即我们可以将原本需要串行的操作并行化,这里 我选择利用 256bit 的向量寄存器。

因为 SM4 是以字节为单位进行加密的,即每 32bit 就需要进行移位等操作,因此利用 256bit 的向量寄存器即可存储 8 组 32bit 的字,这样就可以使得这 8 组字同时进行操作。

首先我们在 main 函数中实现了密钥扩展方案,实现方法和 SM4 基础实现相同,这里不多加赘述。

接着我们实现加密函数。

首先我们需要将数据 load 进寄存器中 (SIMD 指令特点,需要 load 和 store,相当于计组中的从存储器像寄存器中存取数的指令)。

接下来我们定义一个数组变量 x, 分别对应第 i 个分组 (每组 8 个 32bit 字)。

然后就像实现 SM4 基础实现一样, 我们需要经过 32 轮迭代 (和 SM4 基础实现中公式相同)。

在迭代过程中,我利用 SSE 中的 shuffle_epi8 指令替换掉了非线性变换 t,线性变换 L 和过 S-box 也替换成了查 T-table,代码段中 for(0-32) 循环即为查表阶段。

最后从寄存器中 store 出数据到存储器中。

```
void SM4_Encrypt(uint8_t* cin , uint8_t* out , uint32_t* sm4_key)
 2
    {
         _{\text{m256i}} \times [4];
         ___m256i temp[4];
         m256i ff;
          m256i* cin = (m256i*) cin;
         ff = \underline{mm256}\underline{set1}\underline{epi32(0xFF)};
         temp[0] = _mm256_loadu_si256(cin_ + 0);
         temp[1] = _mm256_loadu_si256(cin_ + 1);
         temp[2] = _mm256_loadu_si256(cin_ + 2);
11
         temp[3] = _mm256_loadu_si256(cin_ + 3);
         x[0] = MM256\_EPI32\_0(temp[0], temp[1], temp[2], temp[3]);
12
         x\,[\,1\,] \;=\; MM256\_EPI32\_1(\,temp\,[\,0\,]\;,\;\; temp\,[\,1\,]\;,\;\; temp\,[\,2\,]\;,\;\; temp\,[\,3\,]\,)\,;
13
14
         x[2] = MM256\_EPI32\_2(temp[0], temp[1], temp[2], temp[3]);
15
         x\,[\,3\,] \ = \ MM256\_EPI32\_3(\,temp\,[\,0\,]\,\,,\,\,\,temp\,[\,1\,]\,\,,\,\,\,temp\,[\,2\,]\,\,,\,\,\,temp\,[\,3\,]\,)\,;
         17
        x[0] = \underline{mm256}_{shuffle} epi8(x[0], vindex);
        x[1] = _mm256_shuffle_epi8(x[1], vindex);
18
         x[2] = mm256 \text{ shuffle epi8}(x[2], \text{ vindex});
19
```

```
x[3] = _mm256\_shuffle\_epi8(x[3], vindex);
20
21
                                             for (int i = 0; i < 32; i++)
22
                                                                      _{m256i \ k} = _{mm256\_set1\_epi32(sm4\_key[i]);}
23
                                                                    temp[0] = \_mm256\_xor\_si256(\_mm256\_xor\_si256(x[1], x[2]), \_mm256\_xor\_si256(x[3], k));
24
                                                                    temp [1] = \underline{\phantom{a}} mm256\underline{\phantom{a}} si256(x [0], \underline{\phantom{a}} mm256\underline{\phantom{a}} i32gather\underline{\phantom{a}} epi32((\underline{\phantom{a}} const\underline{\phantom{a}} int*) Table0, \underline{\phantom{a}} mm256\underline{\phantom{a}} and\underline{\phantom{a}} si256(temp [0], \underline{\phantom{a}} function for all the constants for al
25
                                                                     temp[0] = \underline{mm256\_srli\_epi32(temp[0], 8)};
 26
                                                                     temp[1] = mm256 xor si256(temp[1], mm256 i32gather epi32((const int*)Table1, mm256 and si256(temp[0]
27
                                                                     temp[0] = \underline{mm256}\_srli\_epi32(temp[0], 8);
28
                                                                     temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}sor\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}i32gather\underline{\phantom{0}}epi32((const\;\; int*) \\ Table 2 \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}and\underline{\phantom{0}}si256(temp [0]) \\ temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [0]) \\ temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}si25
29
                                                                    temp[0] = \underline{mm256\_srli\_epi32(temp[0], 8)};
 30
                                                                     temp [1] = \underline{\phantom{0}} mm256\underline{\phantom{0}} si256(temp [1] \; , \; \underline{\phantom{0}} mm256\underline{\phantom{0}} i32gather\underline{\phantom{0}} epi32((\underline{\phantom{0}} const \; int*) \\ Table 3 \; , \; \underline{\phantom{0}} mm256\underline{\phantom{0}} and\underline{\phantom{0}} si256(temp [0] \; )
 31
 32
                                                                    x[0] = x[1];
                                                                    x[1] = x[2];
 33
                                                                    x[2] = x[3];
34
                                                                    x[3] = temp[1];
35
36
37
                                             x[0] = _mm256\_shuffle\_epi8(x[0], vindex);
                                            x[1] = _mm256\_shuffle\_epi8(x[1], vindex);
38
                                            x[2] = _mm256_shuffle_epi8(x[2], vindex);
39
                                            x\,[\,3\,]\,=\,\underline{\phantom{a}}mm256\underline{\phantom{a}}shuffle\underline{\phantom{a}}epi8\,(\,x\,[\,3\,]\,\,,\  \, vindex\,)\,;
40
                                             _{mm256\_storeu\_si256((\__{m256i*})out + 0, MM256\_EPI32\_0(x[3], x[2], x[1], x[0]));}
41
                                             _{mm256\_storeu\_si256((\__m256i*)out + 1, MM256\_EPI32\_1(x[3], x[2], x[1], x[0]));}
42
                                               _{mm256\_storeu\_si256((\__m256i*)out + 2, MM256\_EPI32\_2(x[3], x[2], x[1], x[0]));}
 43
 44
                                             _{mm256\_storeu\_si256((\__{m256i*})out + 3, MM256\_EPI32\_3(x[3], x[2], x[1], x[0]));}
 45
                   }
```

2.3 实验结果

由于 CodeBlocks 的编译环境,我未能成功导入 immintrin.h 头文件,导致无法实现 SIMD 指令集的加速,所以我仅仅实现了在 VS 环境中的,请注意结果的时间是加解密 1000 次以后的结果。



图 3: VS 上 SM4 基础实现

3 参考文献

3.1 基础实现

国密 SM4 算法 SM4 算法的 C++ 实现

3.2 SIMD+T-table 实现

SM4 SIMD 指令集优化 (intel) SM4 的快速软件实现技术

4 附录

4.1 SM4 基础代码实现

```
#include <iostream>
   2 #include <string>
   3 #include <cmath>
                #include <time.h>
                  using namespace std;
   7
                 string Sbox[16][16] = { "D6", "90", "E9", "FE", "CC", "E1", "3D", "B7", "16", "B6", "14", "C2", "28", "FB", "2C", "05"},
   8
                                                                                                                                                                                {"2B", "67", "9A", "76", "2A", "BE", "04", "C3", "AA", "44", "13", "26", "49", "86", "06",
   9
                                                                                                                                                                                {"9C", "42", "50", "F4", "91", "EF", "98", "7A", "33", "54", "0B", "43", "ED", "CF", "AC",
  10
                                                                                                                                                                                {"E4", "B3", "1C", "A9", "C9", "08", "E8", "95", "80", "DF", "94", "FA", "75", "8F", "3F"
  11
                                                                                                                                                                                 \{ "47","07","A7","FC","F3","73","17","BA","83","59","3C","19", "E6","85","4F", "E6","85","4F", "E6","85","17", "E6","85","17", "E6","85","17", "E6","85","17", "E6","85","17", "E6","17", "E6","17"
  12
  13
                                                                                                                                                                                {"68", "6B", "81", "B2", "71", "64", "DA", "8B", "F8", "EB", "0F", "4B", "70", "56", "9D",
                                                                                                                                                                                 \{ \text{"1E"}, \text{"24"}, \text{"0E"}, \text{"5E"}, \text{"63"}, \text{"58"}, \text{"D1"}, \text{"A2"}, \text{"25"}, \text{"22"}, \text{"7C"}, \text{"3B"}, \text{"01"}, \text{"21"}, \text{"78"}, \text{"78"}, \text{"78"}, \text{"100}, \text{"100}
  14
                                                                                                                                                                                {"D4", "00", "46", "57", "9F", "D3", "27", "52", "4C", "36", "02", "E7", "A0", "C4", "C8",
  15
                                                                                                                                                                                {"EA", "BF", "8A", "D2", "40", "C7", "38", "B5", "A3", "F7", "F2", "CE", "F9", "61", "15",
  16
                                                                                                                                                                                {"E0", "AE", "5D", "A4", "9B", "34", "1A", "55", "AD", "93", "32", "30", "F5", "8C", "B1",
  17
                                                                                                                                                                                {"1D", "F6", "E2", "2E", "82", "66", "CA", "60", "C0", "29", "23", "AB", "0D", "53", "4E",
                                                                                                                                                                                {"D5", "DB", "37", "45", "DE", "FD", "8E", "2F", "03", "FF", "6A", "72", "6D", "6C", "5B",
  19
                                                                                                                                                                                {"8D", "1B", "AF", "92", "BB", "DD", "BC", "7F", "11", "D9", "5C", "41", "1F", "10", "5A",
 20
                                                                                                                                                                                {"0A", "C1", "31", "88", "A5", "CD", "7B", "BD", "2D", "74", "D0", "12", "B8", "E5", "B4",
 21
                                                                                                                                                                                {"89", "69", "97", "4A", "0C", "96", "77", "7E", "65", "B9", "F1", "09", "C5", "6E", "C6",
 22
                                                                                                                                                                                 \{ "18", "F0", "7D", "EC", "3A", "DC", "4D", "20", "79", "EE", "5F", "3E", "D7", "CB", "39", "18", "18", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "19", "
 23
                  string FK[4] = { "A3B1BAC6", "56AA3350", "677D9197", "B27022DC" };
                  string CK[32] = { "00070E15", "1C232A31", "383F464D", "545B6269"
 25
                                                                                                                                        "70777E85", "8C939AA1", "A8AFB6BD", "C4CBD2D9",
 26
                                                                                                                                        "E0E7EEF5", "FC030A11", "181F262D", "343B4249",
 27
 28
                                                                                                                                       "50575E65", "6C737A81", "888F969D", "A4ABB2B9",
                                                                                                                                       "C0C7CED5", "DCE3EAF1", "F8FF060D", "141B2229",
 29
                                                                                                                                       "30373E45", "4C535A61", "686F767D", "848B9299",
 30
 31
                                                                                                                                       "A0A7AEB5", "BCC3CAD1", "D8DFE6ED", "F4FB0209",
                                                                                                                                       "10171E25", "2C333A41", "484F565D", "646B7279" };
32
33
                   34
                  string BinToHex(string str)
 35
 36
                                                      string hex = "";
 37
                                                     int temp = 0;
 38
39
                                                      while(str.size() % 4 != 0)
```

```
40
                      {
                                                      str = "0" + str;
 41
 42
                                }
                                for (int i = 0; i < str.size(); i += 4)
 43
                      {
 44
                                                      temp = (str[i] - '0') * 8 + (str[i+1] - '0') * 4 + (str[i+2] - '0') * 2 + (str[i+3] - '0')
 45
 46
 47
                                                       {
                                                                            hex += to_string(temp);
 48
                                                       }
 49
 50
                                                       else
 51
                                                       {
 52
                                                                            hex += 'A' + (temp - 10);
 53
                                                       }
54
                                }
                                 return hex;
55
56
57
58
           {\tt string HexToBin(string str)}
59
60
                                 string bin = "";
61
                                 string table [16] = { "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000", "1001", "1010", "1011", "1010", "1011", "1010", "1011", "1010", "1011", "1010", "1011", "1010", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", "1011", 
62
 63
                                 for (int i = 0; i < str.size(); i++)
 64
                      {
                                                       if (str[i] >= 'A'&&str[i] <= 'F')
 65
 66
                                                       {
                                                                             bin += table[str[i] - 'A' + 10];
67
                                                       }
 68
 69
                                                       else
 70
                                                       {
                                                                            bin += table[str[i] - '0'];
71
72
                                }
73
74
                                 return bin;
75
 76
           77
          int HexToDec(char str)
78
79
           {
80
                                 int dec = 0;
81
                                 if (str >= 'A' && str <= 'F')
                      {
82
                                                      dec += (str - 'A' + 10);
83
                                }
84
                                 else
 85
 86
                                 {
 87
                                                       dec += (str - '0');
 88
                                 }
 89
                                 return dec;
90
91
          /******* 循环左移 len 位函数实现 ********/
92
          string LeftShift(string str, int len) {
                                string res = HexToBin(str);
94
                                res = res.substr(len) + res.substr(0, len);
95
                                res = BinToHex(res);
96
97
                                 return res;
98 }
```

```
99
    /************ 字符串异或函数实现***********/
00
   string XOR(string str1, string str2)
101
02
           string res1 = HexToBin(str1);
103
           string res2 = HexToBin(str2);
           string res = "";
05
           for (int i = 0; i < res1.size(); i++)
06
107
                   if (res1[i] == res2[i])
08
09
           {
                           res += "0";
10
                   }
11
                   else
12
                   {
13
                           res += "1";
14
15
           res = BinToHex(res);
17
           return res;
118
119
20
    /***** 非线性变换t函数实现********/
21
22
   string NLTransform(string str) {
23
           string res = "";
24
           for (int i = 0; i < 4; i++) {
25
                  res = res + Sbox[HexToDec(str[2 * i])][HexToDec(str[2 * i + 1])];
26
27
           return res;
28
29 }
130
   /******用于加解密算法中的合成置换T函数实现*******/
131
   string T1(string str)
32
33
34
       string str1 = "";
       string str2 = "";
35
       str1 = NLTransform(str);
36
       str2 = XOR(XOR(XOR(XOR(Str1, LeftShift(str1, 2)), LeftShift(str1, 10)), LeftShift(str1, 18)), LeftShift(str1
37
138
       return str2;
39
   /****** 用于密钥拓展算法中的合成置换T函数实现*******/
141
   string T2(string str)
142
43
       string \ str1 = "";
44
       string str2 = "";
45
       str1 = NLTransform(str);
46
       str2 = XOR(XOR(str1, LeftShift(str1, 13)), LeftShift(str1, 23));
47
       return str2;
48
49 }
50
151 /******************密钥拓展实现**************/
string KeySet(string MK)
53 {
           string K[36] = { XOR(MK. substr (0,8), FK[0]), XOR(MK. substr (8,8), FK[1]), XOR(MK. substr (16,8), FK[2]), XOR(MK.
154
           string rkey = "";
155
           for (int i = 0; i < 32; i++)
56
57
       {
```

```
K[i + 4] = XOR(K[i], T2(XOR(XOR(K[i + 1], K[i + 2]), K[i + 3]), CK[i]));
58
 59
                                            rkey += K[i + 4];
                          }
60
                          return rkey;
61
62
63
         64
65
        string Encryption(string plain, string key)
66
                          string \ cipher [36] = \{ \ plain.substr(0,8), plain.substr(8,8), plain.substr(16,8), plain.substr(24) \};
67
68
                          string rkey = KeySet(key);
 69
                           for (int i = 0; i < 32; i++)
 70
                                            cipher[i+4] = XOR(cipher[i], T1(XOR(XOR(XOR(cipher[i+1], cipher[i+2]), cipher[i+3]), rkwinter(i+1), cipher[i+2]), cipher[i+3]), rkwinter(i+1), cipher[i+2], cipher[i+3]), rkwinter(i+1), cipher[i+3], cipher[i+3]), cipher[i+3]), rkwinter(i+1), cipher[i+3]), cipher[i+3]
 71
72
                          }
                          73
74
75
         76
        string Decryption(string cipher, string key)
177
178
                          string plain [36] = \{ \text{ cipher.substr}(0,8), \text{ cipher.substr}(8,8), \text{ cipher.substr}(16,8), \text{ cipher.substr}(24,8) \};
79
80
                          string rkey = KeySet(key);
 81
                           for (int i = 0; i < 32; i++)
                    {
 82
                                            plain[i+4] = XOR(plain[i], T1(XOR(XOR(XOR(plain[i+1], plain[i+2]), plain[i+3]), rkey.su
 .83
 84
                          return plain [35] + plain [34] + plain [33] + plain [32];
85
86
87
88
        int main()
189
        {
                          string str = "0123456789ABCDEFFEDCBA9876543210";
190
                          cout << "明文: " << str << endl;
91
                          string key = "0123456789ABCDEFFEDCBA9876543210";
92
                          cout << "加密密钥: " << key << endl;
 93
94
                          string cipher;
                          string plain;
95
96
                          double start = clock();
97
                    cipher = Encryption(str, key);
98
                    plain = Decryption(cipher, key);
                          double finish = clock();
                          cout << "密文: " << cipher << endl;
200
                          cout << "解密密钥: " << key << endl;
201
                          cout << "明文: " << plain << endl;
202
                          cout << "加解密时间: " << finish - start << "ms" << endl;
203
204
                           return 0;
05
        }
```

4.2 SIMD+T-table 代码实现

```
#include <iostream>
#include <string>
#include <cmath>
#include <thread>
#include <immintrin.h>
#include <time.h>
```

```
s #define MM256_EPI32_0(a, b, c, d) _mm256_unpacklo_epi64(_mm256_unpacklo_epi32(a, b),_mm256_unpacklo_epi32(c, d)
 9 #define MM256_EPI32_1(a, b, c, d) _mm256_unpackhi_epi64(_mm256_unpacklo_epi32(a, b),_mm256_unpacklo_epi32(c, d)
10 \quad \# define \quad MM256\_EPI32\_2(a, b, c, d) \\ \_mm256\_unpacklo\_epi64(\_mm256\_unpackhi\_epi32(a, b), \_mm256\_unpackhi\_epi32(c, d)
       11
12
        using namespace std;
14
        static uint32 t FK[4] = { 0xa3b1bac6, 0x56aa3350, 0x677d9197, 0xb27022dc };
15
        static uint32\_t CK[32] =
16
17
                 0x00070e15, 0x1c232a31, 0x383f464d, 0x545b6269, 0x70777e85, 0x8c939aa1,
18
                 0xa8afb6bd, 0xc4cbd2d9, 0xe0e7eef5, 0xfc030a11, 0x181f262d, 0x343b4249,
19
                 0x50575e65, 0x6c737a81, 0x888f969d, 0xa4abb2b9, 0xc0c7ced5, 0xdce3eaf1,
20
21
                 0 \\ xf8ff060 \\ d \ , \ 0 \\ x141 \\ b2229 \ , \ 0 \\ x30373 \\ e45 \ , \ 0 \\ x4c535a61 \ , \ 0 \\ x686f767 \\ d \ , \ 0 \\ x848 \\ b9299 \\ x84
                 0 x a 0 a 7 a e b 5 \;,\;\; 0 x b c c 3 c a d 1 \;,\;\; 0 x d 8 d f e 6 e d \;,\;\; 0 x f 4 f b 0 2 0 9 \;,\;\; 0 x 1 0 1 7 1 e 2 5 \;,\;\; 0 x 2 c 3 3 3 a 4 1 \;,\;\;
22
                 0x484f565d, 0x646b7279
23
24
        };
        static uint8_t S_box[256] =
25
26
        {
27
                 0xD6, 0x90, 0xE9, 0xFE, 0xCC, 0xE1, 0x3D, 0xB7, 0x16, 0xB6, 0x14, 0xC2,
                 0x28, 0xFB, 0x2C, 0x05, 0x2B, 0x67, 0x9A, 0x76, 0x2A, 0xBE, 0x04, 0xC3,
28
                 0xAA, 0x44, 0x13, 0x26, 0x49, 0x86, 0x06, 0x99, 0x9C, 0x42, 0x50, 0xF4,
29
                 0x91, 0xEF, 0x98, 0x7A, 0x33, 0x54, 0x0B, 0x43, 0xED, 0xCF, 0xAC, 0x62,
30
31
                 0xE4, 0xB3, 0x1C, 0xA9, 0xC9, 0x08, 0xE8, 0x95, 0x80, 0xDF, 0x94, 0xFA,
32
                 0x75, 0x8F, 0x3F, 0xA6, 0x47, 0x07, 0xA7, 0xFC, 0xF3, 0x73, 0x17, 0xBA,
                 0x83, 0x59, 0x3C, 0x19, 0xE6, 0x85, 0x4F, 0xA8, 0x68, 0x6B, 0x81, 0xB2,
33
                 0x71\,,\ 0x64\,,\ 0xDA,\ 0x8B\,,\ 0xF8\,,\ 0xEB,\ 0x0F\,,\ 0x4B\,,\ 0x70\,,\ 0x56\,,\ 0x9D\,,\ 0x35\,,
34
                 0x1E,\ 0x24\,,\ 0x0E,\ 0x5E,\ 0x63\,,\ 0x58\,,\ 0xD1,\ 0xA2,\ 0x25\,,\ 0x22\,,\ 0x7C\,,\ 0x3B\,,
35
                 0x01, 0x21, 0x78, 0x87, 0xD4, 0x00, 0x46, 0x57, 0x9F, 0xD3, 0x27, 0x52,
36
                 0x4C,\ 0x36\,,\ 0x02\,,\ 0xE7,\ 0xA0,\ 0xC4,\ 0xC8,\ 0x9E,\ 0xEA,\ 0xBF,\ 0x8A,\ 0xD2,
37
                 0x40, 0xC7, 0x38, 0xB5, 0xA3, 0xF7, 0xF2, 0xCE, 0xF9, 0x61, 0x15, 0xA1,
38
39
                 0xE0, 0xAE, 0x5D, 0xA4, 0x9B, 0x34, 0x1A, 0x55, 0xAD, 0x93, 0x32, 0x30,
                 0xF5, 0x8C, 0xB1, 0xE3, 0x1D, 0xF6, 0xE2, 0x2E, 0x82, 0x66, 0xCA, 0x60,
40
                 0xC0, 0x29, 0x23, 0xAB, 0x0D, 0x53, 0x4E, 0x6F, 0xD5, 0xDB, 0x37, 0x45,
41
                 0xDE, \ 0xFD, \ 0x8E, \ 0x2F, \ 0x03\,, \ 0xFF, \ 0x6A, \ 0x72\,, \ 0x6D, \ 0x6C, \ 0x5B, \ 0x51\,,
42
                 0x8D, 0x1B, 0xAF, 0x92, 0xBB, 0xDD, 0xBC, 0x7F, 0x11, 0xD9, 0x5C, 0x41,
43
                 0x1F, 0x10, 0x5A, 0xD8, 0x0A, 0xC1, 0x31, 0x88, 0xA5, 0xCD, 0x7B, 0xBD,
44
                 0x2D, 0x74, 0xD0, 0x12, 0xB8, 0xE5, 0xB4, 0xB0, 0x89, 0x69, 0x97, 0x4A,
45
                 0x0C, 0x96, 0x77, 0x7E, 0x65, 0xB9, 0xF1, 0x09, 0xC5, 0x6E, 0xC6, 0x84,
46
                 0x18\,,\ 0xF0\,,\ 0x7D\,,\ 0xEC,\ 0x3A\,,\ 0xDC,\ 0x4D\,,\ 0x20\,,\ 0x79\,,\ 0xEE,\ 0x5F\,,\ 0x3E\,,
47
48
                 0xD7, 0xCB, 0x39, 0x48
49
        };
        static const uint32_t Table0[] =
50
51
        {
                 0xD55B5B8E, 0x924242D0, 0xEAA7A74D, 0xFDFBFB06, 0xCF3333FC, 0xE2878765,
52
                 0x3DF4F4C9, 0xB5DEDE6B, 0x1658584E, 0xB4DADA6E, 0x14505044, 0xC10B0BCA,
53
                 0x28A0A088\,,\ 0xF8EFEF17,\ 0x2CB0B09C,\ 0x05141411\,,\ 0x2BACAC87,\ 0x669D9DFB,
54
                 0 \\ x986 \\ A6 \\ AF2, \ 0 \\ x77 \\ D9 \\ D9 \\ AE, \ 0 \\ x2A \\ A8 \\ A882, \ 0 \\ xB \\ CFAFA46, \ 0 \\ x04101014 \ , \ 0 \\ xC00 \\ F0 \\ FCF, \ 0 \\ x04101014 \ , \ 0 \\ xC00 \\ x0410101014 \ , \ 0 \\ xC00 \\ x0410101014 \ , \ 0 \\ x041010101014 \ , \ 0 \\ x041010101014 \ , \ 0 \\ x0410101014 \ , 
55
                 0xA8AAAA02, 0x45111154, 0x134C4C5F, 0x269898BE, 0x4825256D, 0x841A1A9E,
56
57
                 0 \\ x 0 6 1 8 1 8 1 \\ E \ , \ 0 \\ x 9 B 6 6 6 6 FD \ , \ 0 \\ x 9 E 7 2 7 2 EC \ , \ 0 \\ x 4 3 0 9 0 9 4 \\ A \ , \ 0 \\ x 5 1 4 1 4 1 1 0 \ , \ 0 \\ x F 7 D 3 D 3 2 4 \ ,
                 0x934646D5, 0xECBFBF53, 0x9A6262F8, 0x7BE9E992, 0x33CCCCFF, 0x55515104,
58
                 0x0B2C2C27, 0x420D0D4F, 0xEEB7B759, 0xCC3F3FF3, 0xAEB2B21C, 0x638989EA,
59
                 0xE7939374, 0xB1CECE7F, 0x1C70706C, 0xABA6A60D, 0xCA2727ED, 0x08202028,
60
                 0xEBA3A348, 0x975656C1, 0x82020280, 0xDC7F7FA3, 0x965252C4, 0xF9EBEB12,
61
                 0x74D5D5A1, 0x8D3E3EB3, 0x3FFCFCC3, 0xA49A9A3E, 0x461D1D5B, 0x071C1C1B,
62
                 0xA59E9E3B, 0xFFF3F30C, 0xF0CFCF3F, 0x72CDCDBF, 0x175C5C4B, 0xB8EAEA52,
63
                 0x810E0E8F, 0x5865653D, 0x3CF0F0CC, 0x1964647D, 0xE59B9B7E, 0x87161691,
64
                 0x4E3D3D73, \ 0xAAA2A208, \ 0x69A1A1C8, \ 0x6AADADC7, \ 0x83060685 \,, \ 0xB0CACA7A, \\
65
66
                 0x70C5C5B5, 0x659191F4, 0xD96B6BB2, 0x892E2EA7, 0xFBE3E318, 0xE8AFAF47,
```

```
0x0F3C3C33, 0x4A2D2D67, 0x71C1C1B0, 0x5759590E, 0x9F7676E9, 0x35D4D4E1,
                      0x1E787866, 0x249090B4, 0x0E383836, 0x5F797926, 0x628D8DEF, 0x59616138,
 68
                      0xD2474795, 0xA08A8A2A, 0x259494B1, 0x228888AA, 0x7DF1F18C, 0x3BECECD7,
69
                      0 \\ x \\ 01040405 \,, \ 0 \\ x \\ 218484 \\ A5 \,, \ 0 \\ x \\ 79 \\ E1E198 \,, \ 0 \\ x \\ 851 \\ E1E9B \,, \ 0 \\ x \\ D7535384 \,, \ 0 \\ x \\ 0000000000 \,,
 70
                      0x4719195E\,,\;\;0x565D5D0B\,,\;\;0x9D7E7EE3\,,\;\;0xD04F4F9F\,,\;\;0x279C9CBB\,,\;\;0x5349491A\,,
71
                      0x4D31317C,\ 0x36D8D8EE,\ 0x0208080A,\ 0xE49F9F7B,\ 0xA2828220,\ 0xC71313D4,
72
                      0xCB2323E8, \ 0x9C7A7AE6, \ 0xE9ABAB42, \ 0xBDFEFE43, \ 0x882A2AA2, \ 0xD14B4B9A, \\
73
                      0x41010140, 0xC41F1FDB, 0x38E0E0D8, 0xB7D6D661, 0xA18E8E2F, 0xF4DFDF2B,
74
75
                      0xF1CBCB3A, 0xCD3B3BF6, 0xFAE7E71D, 0x608585E5, 0x15545441, 0xA3868625,
                      0xE3838360, 0xACBABA16, 0x5C757529, 0xA6929234, 0x996E6EF7, 0x34D0D0E4,
76
                      0x1A686872, 0x54555501, 0xAFB6B619, 0x914E4EDF, 0x32C8C8FA, 0x30C0C0F0,
 77
                      0xF6D7D721, 0x8E3232BC, 0xB3C6C675, 0xE08F8F6F, 0x1D747469, 0xF5DBDB2E,
                      0xE18B8B6A, 0x2EB8B896, 0x800A0A8A, 0x679999FE, 0xC92B2BE2, 0x618181E0,
 79
                      0xC30303C0, 0x29A4A48D, 0x238C8CAF, 0xA9AEAE07, 0x0D343439, 0x524D4D1F,
 80
                      0x4F393976\,,\;\;0x6EBDBDD3,\;\;0xD6575781\,,\;\;0xD86F6FB7\,,\;\;0x37DCDCEB,\;\;0x44151551\,,
81
                      0xDD7B7BA6, 0xFEF7F709, 0x8C3A3AB6, 0x2FBCBC93, 0x030C0C0F, 0xFCFFFF03,
82
                      0x6BA9A9C2, \ 0x73C9C9BA, \ 0x6CB5B5D9, \ 0x6DB1B1DC, \ 0x5A6D6D37, \ 0x50454515\,,
83
                      0x8F3636B9, 0x1B6C6C77, 0xADBEBE13, 0x904A4ADA, 0xB9EEEE57, 0xDE7777A9,
84
                      0xBEF2F24C, \ 0x7EFDFD83, \ 0x114444455 \,, \ 0xDA6767BD, \ 0x5D71712C \,, \ 0x40050545 \,,
85
                      0x1F7C7C63, 0x10404050, 0x5B696932, 0xDB6363B8, 0x0A282822, 0xC20707C5,
86
                      0x31C4C4F5, 0x8A2222A8, 0xA7969631, 0xCE3737F9, 0x7AEDED97, 0xBFF6F649,
87
                      0x2DB4B499, 0x75D1D1A4, 0xD3434390, 0x1248485A, 0xBAE2E258, 0xE6979771,
 88
                      0xB6D2D264, 0xB2C2C270, 0x8B2626AD, 0x68A5A5CD, 0x955E5ECB, 0x4B292962,
 89
 90
                      0x0C30303C, 0x945A5ACE, 0x76DDDDAB, 0x7FF9F986, 0x649595F1, 0xBBE6E65D,
 91
                      0xF2C7C735, 0x0924242D, 0xC61717D1, 0x6FB9B9D6, 0xC51B1BDE, 0x86121294,
                      0x18606078, 0xF3C3C330, 0x7CF5F589, 0xEFB3B35C, 0x3AE8E8D2, 0xDF7373AC,
 92
                      0x4C353579, 0x208080A0, 0x78E5E59D, 0xEDBBBB56, 0x5E7D7D23, 0x3EF8F8C6,
93
                      0xD45F5F8B, 0xC82F2FE7, 0x39E4E4DD, 0x49212168
94
          };
95
          static const uint32_t Table1[] =
96
97
          {
98
                      0x5B5B8ED5, 0x4242D092, 0xA7A74DEA, 0xFBFB06FD, 0x3333FCCF, 0x878765E2,
                      0xF4F4C93D, 0xDEDE6BB5, 0x58584E16, 0xDADA6EB4, 0x50504414, 0x0B0BCAC1,
99
                      0 \\ x \\ A \\ 0 \\ A \\ 0 \\ x \\ E \\ F \\ 17 \\ F \\ 8, \quad 0 \\ x \\ B \\ 0 \\ B \\ 0 \\ C \\ 2 \\ C, \quad 0 \\ x \\ 14141105 \ , \quad 0 \\ x \\ A \\ C \\ A \\ C \\ 87 \\ B, \quad 0 \\ x \\ 9 \\ D \\ 9 \\ D \\ F \\ B \\ 6, \quad 0 \\ x \\ 14141105 \ , \quad 0 \\ x \\ A \\ C \\ A \\ C \\ R \\ 2 \\ B, \quad 0 \\ x \\ 9 \\ D \\ 9 \\ D \\ F \\ B \\ 6, \quad 0 \\ x \\ 14141105 \ , \quad 0 \\ x \\ A \\ C \\ A \\ C \\ R \\ 2 \\ B, \quad 0 \\ x \\ 9 \\ D \\ 9 \\ D \\ F \\ B \\ 6, \quad 0 \\ x \\ 14141105 \ , \quad 0 \\ x \\ 14141
.00
                      0x6A6AF298, \ 0xD9D9AE77, \ 0xA8A8822A, \ 0xFAFA46BC, \ 0x10101404, \ 0x0F0FCFC0,
01
                      0xAAAA02A8, 0x11115445, 0x4C4C5F13, 0x9898BE26, 0x25256D48, 0x1A1A9E84,
02
                      0x18181E06, 0x6666FD9B, 0x7272EC9E, 0x09094A43, 0x41411051, 0xD3D324F7,
.03
                      0x4646D593, 0xBFBF53EC, 0x6262F89A, 0xE9E9927B, 0xCCCCFF33, 0x51510455,
04
                      0x2C2C270B, 0x0D0D4F42, 0xB7B759EE, 0x3F3FF3CC, 0xB2B21CAE, 0x8989EA63,
05
                      06
07
                      0xA3A348EB, 0x5656C197, 0x02028082, 0x7F7FA3DC, 0x5252C496, 0xEBEB12F9,
                      0 \\ x \\ D 5 \\ D 5 \\ A 174, \quad 0 \\ x 3 \\ E 3 \\ E B 3 \\ B \\ D, \quad 0 \\ x F \\ C F \\ C C \\ 33 \\ F, \quad 0 \\ x 9 \\ A 9 \\ A 3 \\ E \\ A \\ 4, \quad 0 \\ x 1 \\ D 1 \\ D 5 \\ B \\ 46, \quad 0 \\ x 1 \\ C 1 \\ C 1 \\ B \\ 0 \\ 7, \quad 0 \\ x 1 \\ C 1 \\ C 1 \\ B \\ 1 \\ 7, \quad 0 \\ x 1 \\ C 1
08
                      0x9E9E3BA5, 0xF3F30CFF, 0xCFCF3FF0, 0xCDCDBF72, 0x5C5C4B17, 0xEAEA52B8,
09
                      0x0E0E8F81, 0x65653D58, 0xF0F0CC3C, 0x64647D19, 0x9B9B7EE5, 0x16169187,
10
                      0x3D3D734E, 0xA2A208AA, 0xA1A1C869, 0xADADC76A, 0x06068583, 0xCACA7AB0,
11
                      0xC5C5B570, 0x9191F465, 0x6B6BB2D9, 0x2E2EA789, 0xE3E318FB, 0xAFAF47E8,
12
                      0x3C3C330F, \ 0x2D2D674A, \ 0xC1C1B071, \ 0x59590E57, \ 0x7676E99F, \ 0xD4D4E135, \\
13
                      0x7878661E\,,\ 0x9090B424\,,\ 0x3838360E\,,\ 0x7979265F\,,\ 0x8D8DEF62,\ 0x61613859\,,
 14
                      0x474795D2, 0x8A8A2AA0, 0x9494B125, 0x8888AA22, 0xF1F18C7D, 0xECECD73B,
 15
 16
                      0x04040501, 0x8484A521, 0xE1E19879, 0x1E1E9B85, 0x535384D7, 0x00000000,
 17
                      0x19195E47, 0x5D5D0B56, 0x7E7EE39D, 0x4F4F9FD0, 0x9C9CBB27, 0x49491A53,
                      0x31317C4D\,,\ 0xD8D8EE36,\ 0x08080A02\,,\ 0x9F9F7BE4\,,\ 0x828220A2\,,\ 0x1313D4C7\,,
18
19
                      0x2323E8CB, 0x7A7AE69C, 0xABAB42E9, 0xFEFE43BD, 0x2A2AA288, 0x4B4B9AD1,
                      0 \times 01014041, 0 \times 1F1FDBC4, 0 \times E0E0D838, 0 \times D6D661B7, 0 \times 8E8E2FA1, 0 \times DFDF2BF4,
20
                      0xCBCB3AF1, 0x3B3BF6CD, 0xE7E71DFA, 0x8585E560, 0x54544115, 0x868625A3,
21
                      0x838360E3, 0xBABA16AC, 0x7575295C, 0x929234A6, 0x6E6EF799, 0xD0D0E434,
22
                      0x6868721A, 0x55550154, 0xB6B619AF, 0x4E4EDF91, 0xC8C8FA32, 0xC0C0F030,
23
                      0 \\ x \\ D7 \\ D721 \\ F6, \ 0 \\ x \\ 3232 \\ BC8E, \ 0 \\ x \\ C6C675 \\ B3, \ 0 \\ x \\ 8F8F6 \\ FE0, \ 0 \\ x \\ 7474691 \\ D, \ 0 \\ x \\ DBDB2 \\ EF5, \ 0 \\ x \\ DF5 \\
24
                      0x8B8B6AE1, 0xB8B8962E, 0x0A0A8A80, 0x9999FE67, 0x2B2BE2C9, 0x8181E061,
25
```

```
0x0303C0C3, 0xA4A48D29, 0x8C8CAF23, 0xAEAE07A9, 0x3434390D, 0x4D4D1F52,
26
27
        0x3939764F, 0xBDBDD36E, 0x575781D6, 0x6F6FB7D8, 0xDCDCEB37, 0x15155144,
        0x7B7BA6DD, 0xF7F709FE, 0x3A3AB68C, 0xBCBC932F, 0x0C0C0F03, 0xFFFF03FC,
28
        0 \times A 9 A 9 C 26 B, \ 0 \times C 9 C 9 B A 73, \ 0 \times B 5 B 5 D 9 6 C, \ 0 \times B 1 B 1 D C 6 D, \ 0 \times 6 D 6 D 3 7 5 A, \ 0 \times 4 5 4 5 1 5 5 0 \,,
29
        0x3636B98F, \ 0x6C6C771B, \ 0xBEBE13AD, \ 0x4A4ADA90, \ 0xEEEE57B9, \ 0x7777A9DE, \\
.30
        0xF2F24CBE, 0xFDFD837E, 0x444445511, 0x6767BDDA, 0x71712C5D, 0x05054540,
31
        0x7C7C631F, \ 0x40405010 \,, \ 0x6969325B \,, \ 0x6363B8DB, \ 0x2828220A \,, \ 0x0707C5C2 \,,
32
        0xC4C4F531, 0x2222A88A, 0x969631A7, 0x3737F9CE, 0xEDED977A, 0xF6F649BF,
.33
34
        0xB4B4992D, 0xD1D1A475, 0x434390D3, 0x48485A12, 0xE2E258BA, 0x979771E6,
        35
        0x30303C0C, 0x5A5ACE94, 0xDDDDAB76, 0xF9F9867F, 0x9595F164, 0xE6E65DBB,
36
37
        0xC7C735F2, 0x24242D09, 0x1717D1C6, 0xB9B9D66F, 0x1B1BDEC5, 0x12129486,
        0x60607818, 0xC3C330F3, 0xF5F5897C, 0xB3B35CEF, 0xE8E8D23A, 0x7373ACDF,
38
39
        0x3535794C, 0x8080A020, 0xE5E59D78, 0xBBBB56ED, 0x7D7D235E, 0xF8F8C63E,
        0x5F5F8BD4, 0x2F2FE7C8, 0xE4E4DD39, 0x21216849
40
41
    };
42
    static const uint32_t Table2[] =
43
   {
        0x5B8ED55B, \ 0x42D09242\,, \ 0xA74DEAA7, \ 0xFB06FDFB, \ 0x33FCCF33\,, \ 0x8765E287\,,
44
        0xF4C93DF4, 0xDE6BB5DE, 0x584E1658, 0xDA6EB4DA, 0x50441450, 0x0BCAC10B,
45
        0xA08828A0, 0xEF17F8EF, 0xB09C2CB0, 0x14110514, 0xAC872BAC, 0x9DFB669D,
46
        0x6AF2986A, 0xD9AE77D9, 0xA8822AA8, 0xFA46BCFA, 0x10140410, 0x0FCFC00F,
47
        0xAA02A8AA, 0x11544511, 0x4C5F134C, 0x98BE2698, 0x256D4825, 0x1A9E841A,
48
49
        0x181E0618, 0x66FD9B66, 0x72EC9E72, 0x094A4309, 0x41105141, 0xD324F7D3,
50
        0x46D59346, 0xBF53ECBF, 0x62F89A62, 0xE9927BE9, 0xCCFF33CC, 0x51045551,
51
        0x2C270B2C, 0x0D4F420D, 0xB759EEB7, 0x3FF3CC3F, 0xB21CAEB2, 0x89EA6389,
        0x9374E793, 0xCE7FB1CE, 0x706C1C70, 0xA60DABA6, 0x27EDCA27, 0x20280820,
52
        0xA348EBA3, 0x56C19756, 0x02808202, 0x7FA3DC7F, 0x52C49652, 0xEB12F9EB,
53
        0xD5A174D5, 0x3EB38D3E, 0xFCC33FFC, 0x9A3EA49A, 0x1D5B461D, 0x1C1B071C,
54
        0x9E3BA59E, 0xF30CFFF3, 0xCF3FF0CF, 0xCDBF72CD, 0x5C4B175C, 0xEA52B8EA,
55
        0x0E8F810E, 0x653D5865, 0xF0CC3CF0, 0x647D1964, 0x9B7EE59B, 0x16918716,
.56
57
        0x3D734E3D, 0xA208AAA2, 0xA1C869A1, 0xADC76AAD, 0x06858306, 0xCA7AB0CA,
        0xC5B570C5, 0x91F46591, 0x6BB2D96B, 0x2EA7892E, 0xE318FBE3, 0xAF47E8AF,
.58
        0x3C330F3C\,,\;\;0x2D674A2D\,,\;\;0xC1B071C1\,,\;\;0x590E5759\,,\;\;0x76E99F76\,,\;\;0xD4E135D4\,,
.59
        0x78661E78\,,\ 0x90B42490\,,\ 0x38360E38\,,\ 0x79265F79\,,\ 0x8DEF628D\,,\ 0x61385961\,,
60
61
        0x4795D247, 0x8A2AA08A, 0x94B12594, 0x88AA2288, 0xF18C7DF1, 0xECD73BEC,
        0x04050104, 0x84A52184, 0xE19879E1, 0x1E9B851E, 0x5384D753, 0x00000000,
62
        0x195E4719, 0x5D0B565D, 0x7EE39D7E, 0x4F9FD04F, 0x9CBB279C, 0x491A5349,
63
        0x317C4D31, 0xD8EE36D8, 0x080A0208, 0x9F7BE49F, 0x8220A282, 0x13D4C713,
64
        0x23E8CB23\,,\ 0x7AE69C7A\,,\ 0xAB42E9AB\,,\ 0xFE43BDFE\,,\ 0x2AA2882A\,,\ 0x4B9AD14B\,,
65
66
        0x01404101, 0x1FDBC41F, 0xE0D838E0, 0xD661B7D6, 0x8E2FA18E, 0xDF2BF4DF,
67
        0xCB3AF1CB, 0x3BF6CD3B, 0xE71DFAE7, 0x85E56085, 0x54411554, 0x8625A386,
        0x8360E383, 0xBA16ACBA, 0x75295C75, 0x9234A692, 0x6EF7996E, 0xD0E434D0,
68
        0x68721A68, 0x55015455, 0xB619AFB6, 0x4EDF914E, 0xC8FA32C8, 0xC0F030C0,
69
        0xD721F6D7, 0x32BC8E32, 0xC675B3C6, 0x8F6FE08F, 0x74691D74, 0xDB2EF5DB,
70
        0x8B6AE18B, 0xB8962EB8, 0x0A8A800A, 0x99FE6799, 0x2BE2C92B, 0x81E06181,
71
        72
        0x39764F39\,,\ 0xBDD36EBD,\ 0x5781D657\,,\ 0x6FB7D86F\,,\ 0xDCEB37DC,\ 0x15514415\,,
73
        0x7BA6DD7B, 0xF709FEF7, 0x3AB68C3A, 0xBC932FBC, 0x0C0F030C, 0xFF03FCFF,
74
75
        0xA9C26BA9, 0xC9BA73C9, 0xB5D96CB5, 0xB1DC6DB1, 0x6D375A6D, 0x45155045,
        0x36B98F36, 0x6C771B6C, 0xBE13ADBE, 0x4ADA904A, 0xEE57B9EE, 0x77A9DE77,
76
        0xF24CBEF2, 0xFD837EFD, 0x44551144, 0x67BDDA67, 0x712C5D71, 0x05454005,
.77
78
        0x7C631F7C, 0x40501040, 0x69325B69, 0x63B8DB63, 0x28220A28, 0x07C5C207,
        0xC4F531C4, 0x22A88A22, 0x9631A796, 0x37F9CE37, 0xED977AED, 0xF649BFF6,
79
        0xB4992DB4, 0xD1A475D1, 0x4390D343, 0x485A1248, 0xE258BAE2, 0x9771E697,
80
        0xD264B6D2, 0xC270B2C2, 0x26AD8B26, 0xA5CD68A5, 0x5ECB955E, 0x29624B29,
81
        0x303C0C30, 0x5ACE945A, 0xDDAB76DD, 0xF9867FF9, 0x95F16495, 0xE65DBBE6,
82
        0xC735F2C7, \ 0x242D0924, \ 0x17D1C617, \ 0xB9D66FB9, \ 0x1BDEC51B, \ 0x12948612, \\
83
        0x60781860, 0xC330F3C3, 0xF5897CF5, 0xB35CEFB3, 0xE8D23AE8, 0x73ACDF73,
```

```
85
                    0x35794C35, 0x80A02080, 0xE59D78E5, 0xBB56EDBB, 0x7D235E7D, 0xF8C63EF8,
 86
                    0x5F8BD45F, 0x2FE7C82F, 0xE4DD39E4, 0x21684921
 87
          };
          static const uint32_t Table3[] =
 88
 89
          {
 90
                    0x8ED55B5B, 0xD0924242, 0x4DEAA7A7, 0x06FDFBFB, 0xFCCF3333, 0x65E28787,
                    0xC93DF4F4, \ 0x6BB5DEDE, \ 0x4E165858\,, \ 0x6EB4DADA, \ 0x44145050\,, \ 0xCAC10B0B, \\
 91
                    0x8828A0A0, 0x17F8EFEF, 0x9C2CB0B0, 0x11051414, 0x872BACAC, 0xFB669D9D,
 92
                    0xF2986A6A, 0xAE77D9D9, 0x822AA8A8, 0x46BCFAFA, 0x14041010, 0xCFC00F0F,
 93
                    0x02A8AAAA, 0x54451111, 0x5F134C4C, 0xBE269898, 0x6D482525, 0x9E841A1A,
 94
                    0x1E061818, 0xFD9B6666, 0xEC9E7272, 0x4A430909, 0x10514141, 0x24F7D3D3,
 .95
                    0xD5934646, 0x53ECBFBF, 0xF89A6262, 0x927BE9E9, 0xFF33CCCC, 0x04555151,
  96
                    0x270B2C2C, 0x4F420D0D, 0x59EEB7B7, 0xF3CC3F3F, 0x1CAEB2B2, 0xEA638989,
  97
                    0x74E79393, 0x7FB1CECE, 0x6C1C7070, 0x0DABA6A6, 0xEDCA2727, 0x28082020,
 98
                    0x48EBA3A3, \ 0xC1975656\,, \ 0x80820202\,, \ 0xA3DC7F7F, \ 0xC4965252\,, \ 0x12F9EBEB,
 99
                    0xA174D5D5, \ 0xB38D3E3E, \ 0xC33FFCFC, \ 0x3EA49A9A, \ 0x5B461D1D, \ 0x1B071C1C, \\
300
                    0x3BA59E9E, \ 0x0CFFF3F3, \ 0x3FF0CFCF, \ 0xBF72CDCD, \ 0x4B175C5C, \ 0x52B8EAEA, \\
201
                    0x8F810E0E, 0x3D586565, 0xCC3CF0F0, 0x7D196464, 0x7EE59B9B, 0x91871616,
202
                    0 \\ x734 \\ E3D3D \\ , \ 0 \\ x08AAA2A2 \\ , \ 0 \\ xC869A1A1 \\ , \ 0 \\ xC76AADAD \\ , \ 0 \\ x85830606 \\ , \ 0 \\ x7AB0CACA \\ , \ 0 \\ x85830606 \\ , \ 0 \\ x8580606 \\ , \ 0 \\ x858060606 \\ , \ 0 \\ x8580606 \\ , \ 0 \\ x858060606 \\ , \ 0 \\ x8580606 \\ , \ 0 \\ x858060606 \\ , \ 0 \\ x85
203
204
                    0xB570C5C5, 0xF4659191, 0xB2D96B6B, 0xA7892E2E, 0x18FBE3E3, 0x47E8AFAF,
                    0x330F3C3C, 0x674A2D2D, 0xB071C1C1, 0x0E575959, 0xE99F7676, 0xE135D4D4,
205
                    0x661E7878, 0xB4249090, 0x360E3838, 0x265F7979, 0xEF628D8D, 0x38596161,
206
                    0x95D24747, 0x2AA08A8A, 0xB1259494, 0xAA228888, 0x8C7DF1F1, 0xD73BECEC,
207
 809
                    0x05010404, 0xA5218484, 0x9879E1E1, 0x9B851E1E, 0x84D75353, 0x00000000,
 09
                    0x5E471919, 0x0B565D5D, 0xE39D7E7E, 0x9FD04F4F, 0xBB279C9C, 0x1A534949,
210
                    0x7C4D3131, 0xEE36D8D8, 0x0A020808, 0x7BE49F9F, 0x20A28282, 0xD4C71313,
                    0xE8CB2323, 0xE69C7A7A, 0x42E9ABAB, 0x43BDFEFE, 0xA2882A2A, 0x9AD14B4B,
211
                    0 \\ x \\ 40 \\ 41 \\ 01 \\ 101 \\ , \quad 0 \\ x \\ DBC41 \\ F1F, \quad 0 \\ x \\ D838 \\ E0E0, \quad 0 \\ x \\ 61 \\ B7D6D6, \quad 0 \\ x \\ 2FA18 \\ E8E, \quad 0 \\ x \\ 2BF4DFDF, \quad 0 \\ x \\ D838 \\ E0E0, \quad 0 \\ x \\ E0E
212
                    0x3AF1CBCB, 0xF6CD3B3B, 0x1DFAE7E7, 0xE5608585, 0x41155454, 0x25A38686,
213
                    0x60E38383\,,\;\;0x16ACBABA,\;\;0x295C7575\,,\;\;0x34A69292\,,\;\;0xF7996E6E\,,\;\;0xE434D0D0\,,
214
                    0x721A6868, 0x01545555, 0x19AFB6B6, 0xDF914E4E, 0xFA32C8C8, 0xF030C0C0,
215
216
                    0x21F6D7D7, 0xBC8E3232, 0x75B3C6C6, 0x6FE08F8F, 0x691D7474, 0x2EF5DBDB,
217
                    0x6AE18B8B, 0x962EB8B8, 0x8A800A0A, 0xFE679999, 0xE2C92B2B, 0xE0618181,
                    0xC0C30303, 0x8D29A4A4, 0xAF238C8C, 0x07A9AEAE, 0x390D3434, 0x1F524D4D,
218
                    0x764F3939\,,\;\;0xD36EBDBD,\;\;0x81D65757\,,\;\;0xB7D86F6F\,,\;\;0xEB37DCDC,\;\;0x51441515\,,
219
 20
                    0xA6DD7B7B, 0x09FEF7F7, 0xB68C3A3A, 0x932FBCBC, 0x0F030C0C, 0x03FCFFFF,
                    0xC26BA9A9, 0xBA73C9C9, 0xD96CB5B5, 0xDC6DB1B1, 0x375A6D6D, 0x15504545,
 21
                    0xB98F3636, 0x771B6C6C, 0x13ADBEBE, 0xDA904A4A, 0x57B9EEEE, 0xA9DE7777,
222
                    0x4CBEF2F2, 0x837EFDFD, 0x55114444, 0xBDDA6767, 0x2C5D7171, 0x45400505,
223
                    0x631F7C7C\,,\ 0x50104040\,,\ 0x325B6969\,,\ 0xB8DB6363\,,\ 0x220A2828\,,\ 0xC5C20707\,,
224
225
                    0xF531C4C4, 0xA88A2222, 0x31A79696, 0xF9CE3737, 0x977AEDED, 0x49BFF6F6,
226
                    0x992DB4B4, \ 0xA475D1D1, \ 0x90D34343, \ 0x5A124848, \ 0x58BAE2E2, \ 0x71E69797, \\
                    0x64B6D2D2, 0x70B2C2C2, 0xAD8B2626, 0xCD68A5A5, 0xCB955E5E, 0x624B2929,
227
                    0x3C0C3030, 0xCE945A5A, 0xAB76DDDD, 0x867FF9F9, 0xF1649595, 0x5DBBE6E6,
228
                    0x35F2C7C7, 0x2D092424, 0xD1C61717, 0xD66FB9B9, 0xDEC51B1B, 0x94861212,
229
                    0x78186060\,,\;\;0x30F3C3C3\,,\;\;0x897CF5F5\,,\;\;0x5CEFB3B3\,,\;\;0xD23AE8E8\,,\;\;0xACDF7373\,,
230
                    0x794C3535\,,\;\;0xA0208080\,,\;\;0x9D78E5E5\,,\;\;0x56EDBBBB,\;\;0x235E7D7D\,,\;\;0xC63EF8F8\,,
231
                    0x8BD45F5F, \ 0xE7C82F2F, \ 0xDD39E4E4, \ 0x68492121
 32
233
          };
234
235
          void SM4_Encrypt(uint8_t* cin , uint8_t* out , uint32_t* sm4_key)
236
          {
237
                    ___m256i x [4];
                    \underline{\phantom{a}} m256i temp [4];
238
                    \underline{\phantom{a}}m256i ff;
239
                     __m256i* cin_ = (__m256i*)cin;
240
241
                    ff = \underline{mm256}\underline{set1}\underline{epi32(0xFF)};
242
                    temp[0] = \underline{mm256}\underline{loadu}\underline{si256}(cin\underline{+0});
                    temp[1] = \underline{mm256\_loadu\_si256(cin\_ + 1)};
```

```
temp[2] = _mm256_loadu_si256(cin_ + 2);
244
  245
                                                   temp[3] = _mm256_loadu_si256(cin_ + 3);
                                                  x\,[\,0\,] \,=\, MM256\_EPI32\_0\,(\,temp\,[\,0\,]\,\,,\,\,\,temp\,[\,1\,]\,\,,\,\,\,temp\,[\,2\,]\,\,,\,\,\,temp\,[\,3\,]\,)\,;
  246
                                                  \mbox{$x[1]$ = $MM256\_EPI32\_1$(temp[0], temp[1], temp[2], temp[3]);} \label{eq:final_emp}
 247
 248
                                                  x[2] = MM256\_EPI32\_2(temp[0], temp[1], temp[2], temp[3]);
 249
                                                  x\,[\,3\,] \ = \ MM256\_EPI32\_3(\,temp\,[\,0\,]\,\,,\,\,\,temp\,[\,1\,]\,\,,\,\,\,temp\,[\,2\,]\,\,,\,\,\,temp\,[\,3\,]\,\,)\,;
                                                       \underline{\hspace{0.5cm}} m256 i \ vindex = \underline{\hspace{0.5cm}} mm256\underline{\hspace{0.5cm}} setr\underline{\hspace{0.5cm}} epi8(2,\ 3,\ 1,\ 0,\ 7,\ 6,\ 5,\ 4,\ 11,\ 10,\ 9,\ 8,\ 15,\ 14,\ 13,\ 12,\ 2,\ \boxed{3},\ 1,\ 0,\ 7,\ 6,\ 5
 250
 251
                                                  x[0] = mm256 \text{ shuffle epi8}(x[0], \text{ vindex});
 252
                                                  x[1] = \underline{mm256}_{shuffle}(x[1], vindex);
                                                  x[2] = \underline{mm256}_{shuffle}(x[2], vindex);
 253
 254
                                                   x[3] = \underline{mm256}\underline{shuffle}\underline{epi8}(x[3], vindex);
   55
                                                    for (int i = 0; i < 32; i++)
   56
  257
                                                                                     _{m256i \ k} = _{mm256}_{set1}_{epi32}(sm4_{key}[i]);
  258
                                                                            temp \ [0] = \_mm256\_xor\_si256 (\_mm256\_xor\_si256 (x \ [1], \ x \ [2]) \ , \ \ \_mm256\_xor\_si256 (x \ [3], \ k));
                                                                           temp [1] = \underline{\phantom{a}} mm256\underline{\phantom{a}} si256(x[0], \underline{\phantom{a}} mm256\underline{\phantom{a}} i32gather\underline{\phantom{a}} epi32((\underline{\phantom{a}} const\underline{\phantom{a}} int*) Table 0, \underline{\phantom{a}} mm256\underline{\phantom{a}} and\underline{\phantom{a}} si256(temp [0], \underline{\phantom{a}} for equation for each of the constant o
 259
 260
                                                                           temp[0] = \underline{mm256\_srli\_epi32(temp[0], 8)};
 261
                                                                            temp [1] = \underline{\phantom{a}}mm256\underline{\phantom{a}}si256(temp [1] \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}i32gather\underline{\phantom{a}}epi32((const \;\; int*) \\ Table 1 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 1 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 2 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 3 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 4 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 5 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 6 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 7 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 7 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 8 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 9 \;,\; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(t
 262
                                                                            temp[0] = \underline{-mm256\_srli\_epi32(temp[0], 8)};
                                                                            temp [1] = \underline{\phantom{}}mm256\underline{\phantom{}}sor\underline{\phantom{}}si256(temp [1] \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}i32gather\underline{\phantom{}}epi32((const \; int*) \\ Table 2 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 2 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 3 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 4 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 5 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 6 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 7 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 7 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 8 \;,\; \underline{\phantom{}}mm256\underline{\phantom{}}and\underline{\phantom{}}si256(temp [0] \\ Table 9 
 263
                                                                           temp \left[ \begin{smallmatrix} 0 \end{smallmatrix} \right] \; = \; \_mm256\_srli\_epi32 \\ (temp \left[ \begin{smallmatrix} 0 \end{smallmatrix} \right] \,, \;\; 8);
 264
                                                                            temp [1] = \underline{\phantom{a}}mm256\underline{\phantom{a}}si256(temp [1] \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}i32gather\underline{\phantom{a}}epi32((\underline{\phantom{a}}const \; int*) \\ Table 3 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 3 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 4 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 5 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 6 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 7 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 8 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline{\phantom{a}}si256(temp [0]) \\ Table 9 \; , \; \underline{\phantom{a}}mm256\underline{\phantom{a}}and\underline
 265
 266
                                                                            x[0] = x[1];
   67
                                                                           x[1] = x[2];
   68
                                                                           x[2] = x[3];
                                                                           x[3] = temp[1];
   69
  270
                                                  }
 271
                                                  x[0] = \underline{mm256}\underline{shuffle}\underline{epi8}(x[0], vindex);
 272
                                                  x[1] = \underline{mm256}_{shuffle}(x[1], vindex);
 273
                                                  x[2] = \underline{mm256}\underline{shuffle}\underline{epi8}(x[2], vindex);
 274
                                                  x[3] = \underline{mm256}_{shuffle}(x[3], vindex);
                                                   _{mm256\_storeu\_si256((\__{m256i*})out + 0, MM256\_EPI32\_0(x[3], x[2], x[1], x[0]));}
 275
                                                   \underline{\ \ } mm256\underline{\ \ }storeu\underline{\ \ }si256((\underline{\ \ \ }m256i*)out\ +\ 1,\ MM256\underline{\ \ }EPI32\underline{\ \ }1(x[3]\ ,\ x[2]\ ,\ x[1]\ ,\ x[0]));
 276
 277
                                                    _{mm256\_storeu\_si256((\__{m256i*})out + 2, MM256\_EPI32\_2(x[3], x[2], x[1], x[0]));}
 278
                                                   _{mm256\_storeu\_si256((\__{m256i*})out + 3, MM256\_EPI32\_3(x[3], x[2], x[1], x[0]));}
 279
                         }
 280
 281
                         void SM4_Decrypt(uint8_t* out, uint8_t* cin, uint32_t* sm4_key)
 282
                         {
 283
                                                   ___m256i x [4];
 284
                                                   ___m256i temp[4];
 285
                                                   ___m256i ff;
                                                       __m256i* out_ = (__m256i*)out;
 286
                                                   ff = mm256 set1 epi32(0xFF):
 \frac{1}{2}87
   88
                                                  temp[0] = \underline{mm256}\underline{loadu}\underline{si256}(out\underline{+0});
  289
                                                  temp[1] = \underline{mm256\_loadu\_si256(out\_ + 1)};
  290
                                                  temp[2] = \underline{mm256}\underline{loadu}\underline{si256}(out\underline{+2});
                                                   temp[3] = _mm256_loadu_si256(out_ + 3);
   91
  292
                                                  x[0] = MM256\_EPI32\_0(temp[0], temp[1], temp[2], temp[3]);
  293
                                                  x[1] = MM256\_EPI32\_1(temp[0], temp[1], temp[2], temp[3]);
                                                  x\,[\,2\,] \ = \ MM256\_EPI32\_2(\,temp\,[\,0\,]\;,\;\; temp\,[\,1\,]\;,\;\; temp\,[\,2\,]\;,\;\; temp\,[\,3\,]\,)\,;
 294
 295
                                                  x[3] = MM256\_EPI32\_3(temp[0], temp[1], temp[2], temp[3]);
                                                      \underline{\hspace{0.5cm}} m256 i \ vindex = \underline{\hspace{0.5cm}} m256\underline{\hspace{0.5cm}} setr\underline{\hspace{0.5cm}} epi8(2,\ 3,\ 1,\ 0,\ 7,\ 6,\ 5,\ 4,\ 11,\ 10,\ 9,\ 8,\ 15,\ 14,\ 13,\ 12,\ 2,\ \boxed{3},\ 1,\ 0,\ 7,\ 6,\ 5
  296
                                                  x[0] = _mm256_shuffle_epi8(x[0], vindex);
 297
 298
                                                  x[1] = \underline{mm256}_{shuffle}(x[1], vindex);
                                                  x[2] = \underline{mm256}_{shuffle}(x[2], vindex);
 299
 300
                                                   x[3] = \underline{mm256}\underline{shuffle}\underline{epi8}(x[3], vindex);
 301
                                                   for (int i = 0; i < 32; i++)
 302
                                                   {
```

```
303
                                                                                          _{m256i} k = _{mm256} set1_{epi32} (sm4_{key}[31 - i]);
  304
                                                                                  temp[0] = _{mm256\_xor\_si256(\_mm256\_xor\_si256(x[1], x[2]), \_mm256\_xor\_si256(x[3], k))};
                                                                                  temp [1] = \underline{\phantom{a}} mm256\underline{\phantom{a}} xor\underline{\phantom{a}} si256 (x [0], \underline{\phantom{a}} mm256\underline{\phantom{a}} i32 gather\underline{\phantom{a}} epi32 ((\underline{\phantom{a}} const\underline{\phantom{a}} int*) Table 0, \underline{\phantom{a}} mm256\underline{\phantom{a}} and\underline{\phantom{a}} si256 (temp [0], \underline{\phantom{a}} factor for a function of the constant o
  305
                                                                                  temp \left[ \begin{smallmatrix} 0 \end{smallmatrix} \right] \; = \; \underline{\phantom{a}} mm256\underline{\phantom{a}} srli\underline{\phantom{a}} epi32 \left( temp \left[ \begin{smallmatrix} 0 \end{smallmatrix} \right], \;\; 8 \right);
  306
                                                                                 temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}i32gather\underline{\phantom{0}}epi32((\underline{\phantom{0}}const\;\; int*) \\ Table 1 \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}and\underline{\phantom{0}}si256(temp [0]) \\ = \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [0]) \\ = \underline{\phantom{0}}mm256\underline{\phantom{0}si256(temp [0]) \\ 
  307
                                                                                 temp \left[ \begin{smallmatrix} 0 \end{smallmatrix} \right] \; = \; \_mm256\_srli\_epi32 \\ (temp \left[ \begin{smallmatrix} 0 \end{smallmatrix} \right] \,, \;\; 8);
  308
                                                                                  temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}sor\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}i32gather\underline{\phantom{0}}epi32((const\;\; int*) \\ Table 2 \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}and\underline{\phantom{0}}si256(temp [0]) \\ temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [0]) \\ temp [1] = \underline{\phantom{0}}mm256\underline{\phantom{0}}si256(temp [1] \;,\; \underline{\phantom{0}}mm256\underline{\phantom{0}}si25
  309
 310
                                                                                 temp[0] = mm256 \text{ srli epi} 32(temp[0], 8);
 311
                                                                                 temp[1] = _mm256_xor_si256(temp[1], _mm256_i32gather_epi32((const_int*)Table3, _mm256_and_si256(temp[0]
                                                                                 x\,[\,0\,] \;=\; x\,[\,1\,]\,;
  312
  313
                                                                                 x[1] = x[2];
  314
                                                                                 x[2] = x[3];
  315
                                                                                 x[3] = temp[1];
 316
                                                       }
 317
                                                       x\,[\,0\,]\,=\,\underline{\phantom{a}}mm256\underline{\phantom{a}}shuffle\underline{\phantom{a}}epi8\,(x\,[\,0\,]\,\,,\  \, vindex\,)\,;
                                                       x[1] = _mm256\_shuffle\_epi8(x[1], vindex);
 318
319
                                                       x[2] = \underline{mm256}\underline{shuffle}\underline{epi8}(x[2], vindex);
320
                                                       x\,[\,3\,]\,=\,\underline{\phantom{a}}mm256\underline{\phantom{a}}shuffle\underline{\phantom{a}}epi8\,(x\,[\,3\,]\,\,,\  \, vindex\,)\,;
                                                        _{mm256\_storeu\_si256((\__{m256i*})cin + 0, MM256\_EPI32\_0(x[3], x[2], x[1], x[0]));
321
                                                        _{mm256\_storeu\_si256((\__{m256i*})cin + 1, MM256\_EPI32\_1(x[3], x[2], x[1], x[0]));
322
                                                        _{mm256\_storeu\_si256((\__m256i*)cin + 2, MM256\_EPI32\_2(x[3], x[2], x[1], x[0]));}
  323
                                                        _{mm256\_storeu\_si256((\__{m256i*})cin + 3, MM256\_EPI32\_3(x[3], x[2], x[1], x[0]));
  324
  325
                            }
  326
  327
                            int main()
  328
                            {
                                                        unsigned char key [128] = { 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xbb, 0x
  329
                                                        330
  331
                                                        uint32_t* sm4_key = (uint32_t*) malloc(32 * sizeof(uint32_t));
 332
                                                         if (sm4_key)
  333
                                                        {
 334
                                                                                  uint32_t k[4];
  335
                                                                                  uint32_t tmp;
                                                                                  \label{eq:control_state} \mbox{uint8$\_$t*} \mbox{t*} \mbox{tmp}\_8 = \mbox{(uint8$\_$t*}) \& \mbox{tmp};
  336
  337
                                                                                   for (int i = 0; i < 4; i++)
   338
  339
                                                                                                             int j = 4 * i;
                                                                                                            k[i] = (key[j + 0] \ll 24) | (key[j + 1] \ll 16) | (key[j + 2] \ll 8) | (key[j + 3]);
  340
 341
                                                                                                            k[i] = k[i] ^ FK[i];
  342
                                                                                  }
 343
                                                                                  for (int i = 0; i < 32; i++)
 344
                                                                                   {
                                                                                                            tmp = k[1] ^ k[2] ^ k[3] ^ CK[i];
 345
                                                                                                              for (int j = 0; j < 4; j++)
  346
  347
                                                                                                              {
  348
                                                                                                                                      tmp\_8[j] = S\_box[tmp\_8[j]];
   349
                                                                                                            sm4_{key}[i] = k[0] ^ tmp ^ (tmp << 13) ^ (tmp >> 23);
    50
    51
                                                                                                            k[0] = k[1];
                                                                                                            k[1] = k[2];
   352
                                                                                                            k[2] = k[3];
   353
   354
                                                                                                            k[3] = sm4\_key[i];
  355
                                                                                  double start = clock();
  356
  357
                                                                                  for (int i = 0; i < 1000; i++)
  358
                                                                                   {
                                                                                                            SM4_Encrypt(cin, cin, sm4_key);
  359
 360
```

361

double end = clock();

```
cout << "运行1000次加密时间:" << end - start << "ms" << endl;
363
             start = clock();
364
             for (int i = 0; i < 1000; i++)
365
366
                 SM4\_Decrypt(\,cin\,\,,\,\,cin\,\,,\,\,sm4\_key\,)\,;
            }
             end = clock();
368
             cout << "运行1000次解密时间:" << end - start << "ms" << endl;
369
370
            free(sm4_key);
371
372
        return 0;
373 }
```