AES GCM 加密100000次代码

NuMicro® 32位系列微控制器范例代码介绍

文件信息

应用简述	此代码用 M2354 对 400 字节加密十万次，每加密一次 GPIO 反转一次。测 GPIO 反转持续时间，可得 400 字节加密十万次，耗时约 1.64 秒
BSP 版本	M2354 BSP CMSIS V3.00.003
开发平台	NuMaker-M2354 Ver1.1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

M2354 有超低功耗特点，并且有个加密外设，所以非常适合有数据加密通信的物联网应用。

用这个代码，可以测量出 M2354 对 400 字节采用 AES GCM 模式加密十万次，只需 1.64 秒。

1.1 原理

用 M2354 MCU 对 400 字节加密 100000 次，测量加密时间。

1.2 执行结果

每加密一次 PD3 引脚会反转一次。图 1-1 显示对 400 字节加密一次耗时仅 16.3us。

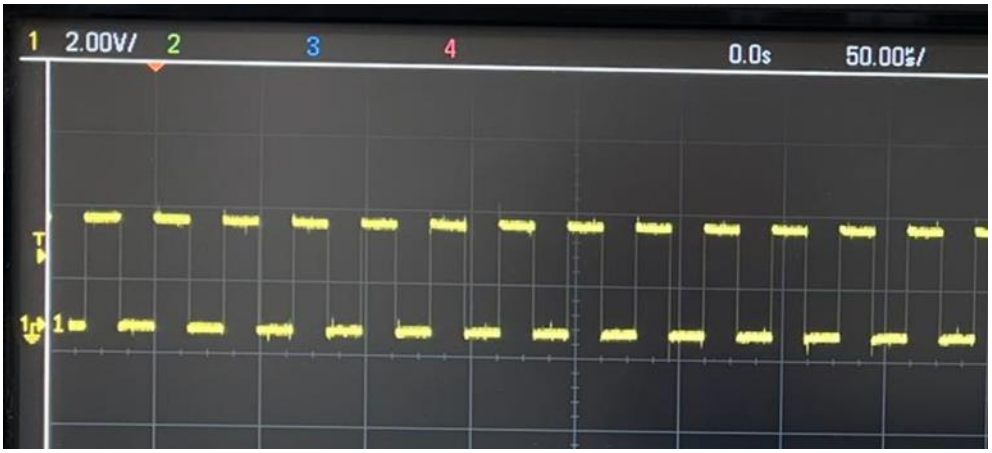

图 1- 1 每加密一次 PD3 引脚反转一次

图 1-2 显示对 400 字节，完成十万次加密，共耗时约 1.64 秒。

图 1-2 加密 100000 次的时间

图 1-3 左边是 UART 口打印的 IV, AU 数据和待加密的明文,右边是加密后的密文。

```
AES GCM Test

Iv(16):
0x0000  12345678 23456789 3456789a 00000001

Au(16):
0x0000  11223344 22334455 33445566 44556677

Plaintext(400):
0x0000  00000000 45643635 00000000 52347576
0x0010  00000001 65754453 00001000 86452343
0x0020  00000002 45643635 00002000 52347576
0x0030  00000003 65754453 00003000 86452343
0x0040  00000004 45643635 00004000 52347576
0x0050  00000005 65754453 00005000 86452343
0x0060  00000006 45643635 00006000 52347576
0x0070  00000007 65754453 00007000 86452343
0x0080  00000008 45643635 00008000 52347576
0x0090  00000009 65754453 00009000 86452343
0x00A0  0000000a 45643635 0000a000 52347576
0x00B0  0000000b 65754453 0000b000 86452343
0x00C0  0000000c 45643635 0000c000 52347576
0x00D0  0000000d 65754453 0000d000 86452343
0x00E0  0000000e 45643635 0000e000 52347576
0x00F0  0000000f 65754453 0000f000 86452343
0x0100  00000010 45643635 00010000 52347576
0x0110  00000011 65754453 00011000 86452343
0x0120  00000012 45643635 00012000 52347576
0x0130  00000013 65754453 00013000 86452343
0x0140  00000014 45643635 00014000 52347576
0x0150  00000015 65754453 00015000 86452343
0x0160  00000016 45643635 00016000 52347576
0x0170  00000017 65754453 00017000 86452343
0x0180  00000018 45643635 00018000 52347576

Cyphertext 400 bytes, Tag 16 bytes:
0x0000  f956c72b 493f1e5a 8bde6123 7d44409d
0x0010  dc931083 966ec7a8 169ef95f 842febbe
0x0020  8332ee60 9557b2b7 77b5afa2 a1a517df
0x0030  3f3c26ea c669edba 45929aaf e2cf1513
0x0040  58f08909 c83e09af af55d98f 05e9e649
0x0050  9583efbf bbacf07e 24e31b8c e2933900
0x0060  1199b232 d6898968 9c577443 2f6eb5b6
0x0070  b6d26d32 ded75426 6e4721cd 3168eeb9
0x0080  d0e2a29b b4e7c5cc be1faa20 3edf6366
0x0090  c80ba5ff 3cac625c 69b986a1 8facf0e3
0x00A0  e36b6fe7 88f9a707 0177bf19 9498bd15
0x00B0  101279cb d598f2c5 15656351 49f0f919
0x00C0  a03e0adc ce1e35fe 1b444462 e2b38260
0x00D0  6ece629b 72506320 25067308 ad04b13e
0x00E0  47aee9d8 b8bc1d5a 7a2afcb2 ffb3813b
0x00F0  e12fdb45 67eeecb1 9a12383e 9ca08ea0
0x0100  76b8de97 5c08cf5c abdf3267 51f8ee78
0x0110  596c3b44 1aafe225 f7618a8b 01bdd623
0x0120  79eb9da5 8ca54a05 5ccb4e28 1e70d7b7
0x0130  cf756671 fc868142 1a5d4ed0 1a05cffa
0x0140  2c880d1f e9f2aa03 a0e4ae47 76036b9b
0x0150  d3d17c66 2a73963d 272f4174 8b872cac
0x0160  dbc4e12e edbf1d9e 77ecec6d a1ec616a
0x0170  bb475033 3a569e89 b2e07696 ca9c9d28
0x0180  6f989c60 cb0e4dbe eeecd530 b31fe8ad
0x0190  54ef9b1d 75b86407 629d6b2c df5e4fee

Tag = d7190cde a46c9ea2 70c94a7b 27e3e368
```

图 1-3 加密前的明文和加密后的密文

图 1-4 显示解密的结果和开始的明文一样. Tag 数据也一致。

```
Decryptoin output(400) and Tag(16):
0x0000  00000000 45643635 00000000 52347576
0x0010  00000001 65754453 00001000 86452343
0x0020  00000002 45643635 00002000 52347576
0x0030  00000003 65754453 00003000 86452343
0x0040  00000004 45643635 00004000 52347576
0x0050  00000005 65754453 00005000 86452343
0x0060  00000006 45643635 00006000 52347576
0x0070  00000007 65754453 00007000 86452343
0x0080  00000008 45643635 00008000 52347576
0x0090  00000009 65754453 00009000 86452343
0x00A0  0000000a 45643635 0000a000 52347576
0x00B0  0000000b 65754453 0000b000 86452343
0x00C0  0000000c 45643635 0000c000 52347576
0x00D0  0000000d 65754453 0000d000 86452343
0x00E0  0000000e 45643635 0000e000 52347576
0x00F0  0000000f 65754453 0000f000 86452343
0x0100  00000010 45643635 00010000 52347576
0x0110  00000011 65754453 00011000 86452343
0x0120  00000012 45643635 00012000 52347576
0x0130  00000013 65754453 00013000 86452343
0x0140  00000014 45643635 00014000 52347576
0x0150  00000015 65754453 00015000 86452343
0x0160  00000016 45643635 00016000 52347576
0x0170  00000017 65754453 00017000 86452343
0x0180  00000018 45643635 00018000 52347576
0x0190  d7190cde a46c9ea2 70c94a7b 27e3e368

Test PASS!
```

图 1-4 解密后与明文对比结果

2. 代码介绍

代码是定义了三个数组：g_IvAuPlain[128] 存放 IV 向量，AU 数据和明文，Buff_1[128] 和 Buff_2[128] 做为加密缓存。

这三个数组的开头 32 个字节，必须是一样的，都是 24 字节的向量 IV 跟着 一个字长的数据 1，然后是 4 个字长的认证数据 AU。

```
__ALIGNED(4) uint32_t g_IvAuPlain[128] =
{
    0x12345678, 0x23456789, 0x3456789a, 0x1,             // 3words of IV following 1
    0x11223344, 0x22334455, 0x33445566, 0x44556677,      // 16Bytes authentication data

                // The plaintext starting from &g_IvAuPlain[8] MUST be 16*n bytes
0x00000000,0x45643635,0x00000000,0x52347576,0x00000001,0x65754453,0x00001000,0x86452343,
0x00000002,0x45643635,0x00002000,0x52347576,0x00000003,0x65754453,0x00003000,0x86452343,
0x00000004,0x45643635,0x00004000,0x52347576,0x00000005,0x65754453,0x00005000,0x86452343,
0x00000006,0x45643635,0x00006000,0x52347576,0x00000007,0x65754453,0x00007000,0x86452343,
0x00000008,0x45643635,0x00008000,0x52347576,0x00000009,0x65754453,0x00009000,0x86452343,
0x0000000a,0x45643635,0x0000a000,0x52347576,0x0000000b,0x65754453,0x0000b000,0x86452343,
0x0000000c,0x45643635,0x0000c000,0x52347576,0x0000000d,0x65754453,0x0000d000,0x86452343,
0x0000000e,0x45643635,0x0000e000,0x52347576,0x0000000f,0x65754453,0x0000f000,0x86452343,
0x00000010,0x45643635,0x00010000,0x52347576,0x00000011,0x65754453,0x00011000,0x86452343,
0x00000012,0x45643635,0x00012000,0x52347576,0x00000013,0x65754453,0x00013000,0x86452343,
0x00000014,0x45643635,0x00014000,0x52347576,0x00000015,0x65754453,0x00015000,0x86452343,
0x00000016,0x45643635,0x00016000,0x52347576,0x00000017,0x65754453,0x00017000,0x86452343,
0x00000018,0x45643635,0x00018000,0x52347576,
};


__ALIGNED(4) uint32_t Buff_1[128]={
    0x12345678, 0x23456789, 0x3456789a, 0x1,          // 3words of IV following 1
    0x11223344, 0x22334455, 0x33445566, 0x44556677,   // 16Bytes authentication data
              // Plaintext is after 32 bytes
 } ;

__ALIGNED(4) uint32_t Buff_2[128] = {
    0x12345678, 0x23456789, 0x3456789a, 0x1,          // 3words of IV following 1
    0x11223344, 0x22334455, 0x33445566, 0x44556677,   // 16Bytes authentication data
              // Plaintext is after 32 bytes
 }
```

加密过程:

1. 对缓存 Buff_1[0] ~ Buff_1[107] 中的 IV, AU 和 400 字节加密，加密后的密文存放到缓存 Buff_2[8] ~ Buff_2[107], 标签数据 Tag 存入 Buff_2[108] ~ Buff_2[111].

2. 对缓存 Buff_2[0] ~ Buff_2[107] 中的 IV, AU 和 400 字节，做加密运算，加密结果（密文）存入 Buff_1[8] ~ Buff_1[107], 标签数据 Tag 存入 Buff_1[108] ~ Buff_1[111].

上述流程完成了两次加密，以上过程重复 50000 次后，就完成了十万次加密。

```
    do
    {
        CRPT->AES_GCM_IVCNT[0] = Len_Iv ;                   // Set byte count of IV
        CRPT->AES_GCM_ACNT[0] = Len_Au_Aligned;             // Set bytes count of A
        CRPT->AES_GCM_PCNT[0] = Len_Ptext_Aligned;          // Set bytes count of P

        CRPT->AES_SADDR = (uint32_t)Buff_1 ;                // Configure DMA of AES
        CRPT->AES_DADDR = (uint32_t)(&Buff_2[Len_IvAu_Aligned / 4]);
        CRPT->AES_CNT   = Len_IvAu_Aligned + Len_Ptext_Aligned;

        AES_Run(CRPT_AES_CTL_ENCRPT_Msk | KEY_SIZE_u32 | GCM_MODE | DMAEN | DMALAST) ;
        LED_LIGHT()  ;                          // Indicate that encrypt has finished once
        ++Buff_2[1] ;
        ++Buff_1[1] ;                           // IV is changed for next encrypting

        if (Tagbuf[0] == 0)                     // Copy the Tag of first encrypting
        {
            for (i = 0; i < 4; ++i)             // Tag is following cyphertext
                Tagbuf[i] = Buff_2[(Len_IvAu_Aligned + Len_Ptext_Aligned) / 4 + i] ;
        }

        //      printf("Cypher text %dbytes, Tag %dbytes:\n", Len_Ptext_Aligned,16);
        //      PrintHex_4B(&Buff_2[Len_IvAu_Aligned/4], Len_Ptext_Aligned/4 +4);

        CRPT->AES_GCM_IVCNT[0] = Len_Iv ;                   // Set byte count of IV
        CRPT->AES_GCM_ACNT[0] = Len_Au_Aligned;             // Set bytes count of A
        CRPT->AES_GCM_PCNT[0] = Len_Ptext_Aligned;          // Set bytes count of P

        CRPT->AES_SADDR = (uint32_t)Buff_2 ;                // Configure DMA of AES
        CRPT->AES_DADDR = (uint32_t)(&Buff_1[Len_IvAu_Aligned / 4]);
        CRPT->AES_CNT   = Len_IvAu_Aligned + Len_Ptext_Aligned;

        AES_Run(CRPT_AES_CTL_ENCRPT_Msk | KEY_SIZE_u32 | GCM_MODE | DMAEN | DMALAST) ;
        LED_DARK() ;                            // Indicate that encrypt has finished once
        ++Buff_2[1] ;
        ++Buff_1[1] ;                           // IV is changed for next encrypting
    } while (++Cnt < 50000);                    // 2*50000 times encrypting
```

十万次加密结束后，再打印出密文和第一次标签数据 Tag.

```
    printf("Cyphertext %d bytes, Tag %d bytes:\n", Len_Ptext_Aligned, 16);
    PrintHex_4B(&Buff_1[Len_IvAu_Aligned / 4], Len_Ptext_Aligned / 4 + 4); // Printing
Cypertext

    printf("Tag = ");

    for (i = 0; i < 4; ++i) printf("%08x ", Tagbuf[i]) ;         // Printing Tag

    printf("\n\n") ;
```

解密和加密过程相似，解密不是本文讨论的议题，此不再累述。

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M2354_Series_BSP_CMSIS_V3.00.003
- IDE 版本
  - Keil uVersion 5.26

3.2 硬件需求

- 电路组件
  - NuMaker-M2354 Ver1.1

4. 目录信息

EC_M2354_AES_GCM_100K_Times_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *EC_ M2354_AES_GCM_100K_Times.uvprojx*。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2022.07.07	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.