OTFDEC efficiency

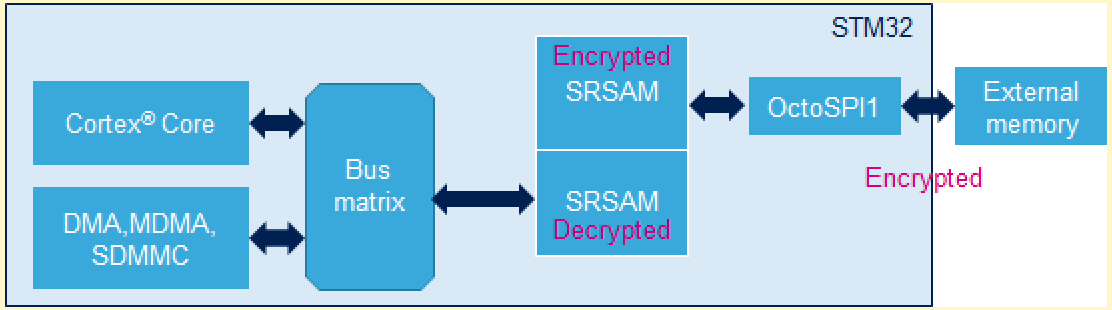
基于 STM32H735G-DK 板的验证

OTFDEC 简介

- OTFDEC = On-the-fly Decryption
- 应用场景：
  - 保护片外存储器上代码（包括指令/数据）的机密性，同时保证代码执行的效率

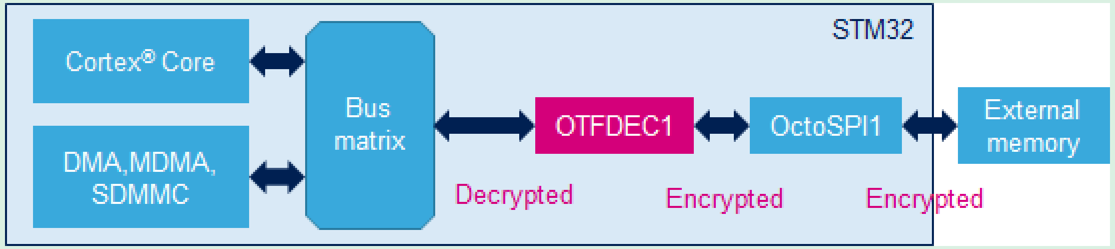
不带OTFDEC的情况

- 先把加密代码读到内部SRAM，
- 使用软件或者加解密硬件，解密到SRAM的其他区域
- 内核从SRAM执行解密后的代码



带OTFDEC的情况

- 配置OTFDEC：要作用的片外区域范围，加解密参数
- 配置OSPI工作在memory map模式
- 内核直接执行外部加密代码



OTFDEC 简介

- 工作原理：
  - 对外部存储区上加密代码的执行，OTFDEC解密后直接送到总线上，供内核执行
  - OTFDEC工作在AES-128-CTR模式
  - 需要配置OctoSPI工作在memory-map模式
  - 该场景下，OTFDEC工作在解密角色（也可以工作在加密角色，不在本场景讨论范围内）
  - OTFDEC有4个配置单元，可以独立控制外部存储区上的4个region

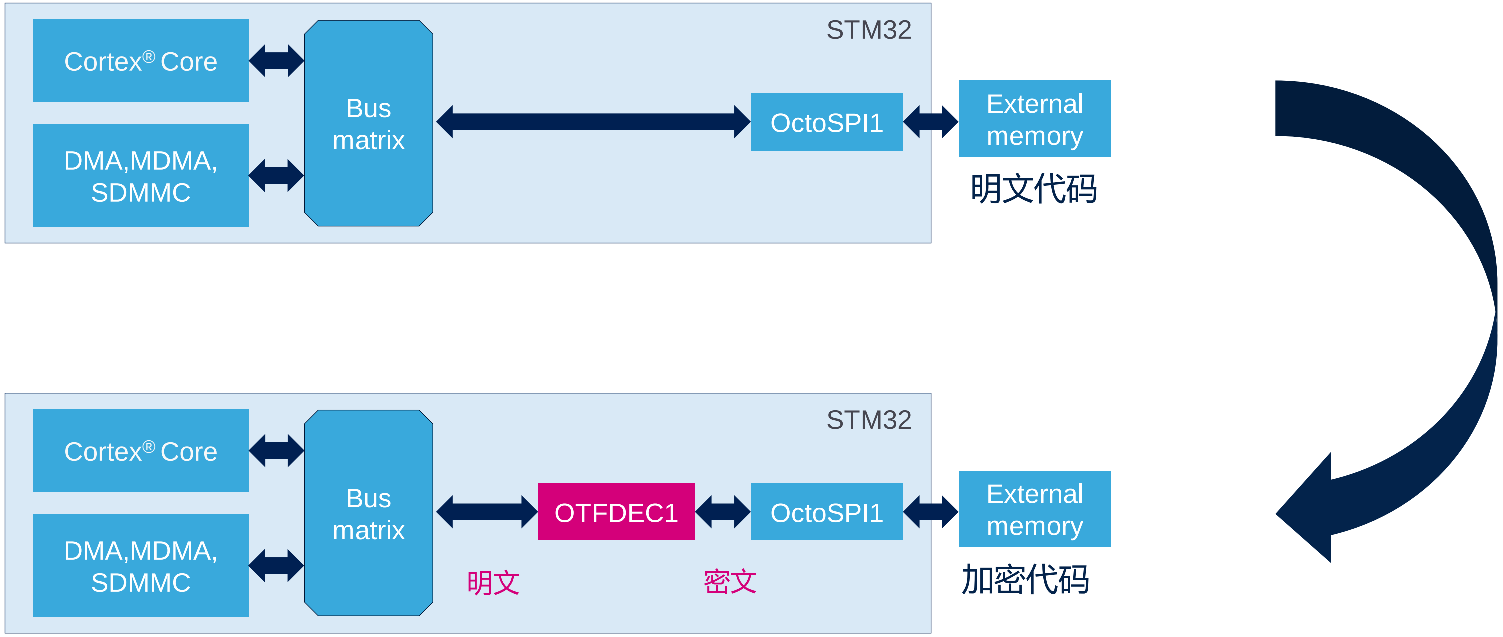
- 集成OTFDEC的STM32系列
  - STM32H73x：STM32H730，STM32H733，STM32H735
  - STM32L56x：STM32L562
  - STM32U58x：STM32U585

- 有关OTDEC更多详情，参见： **STM32L5 进阶课程， 11. OTFDEC，无缝扩大代码的安全存储空间**

life.augmented

OTFDEC 解密引入的延迟有多少?

- 相对于执行外部Flash上的明文代码，执行外部Flash的加密代码，OTFDEC解密操作引入的延迟有多少?



延迟：

片外代码读取延迟

解密带来的延迟

Demo 设计

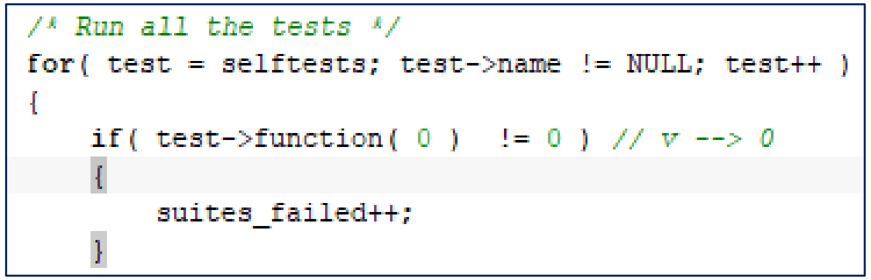
- 目标测试程序：Crypto_Selftest，开启最高优化等级后image size约为63K

测试场景	描述	Demo Project	
		启动工程	目标测试工程
场景1	目标程序，明文，运行在片内Flash	不需要启动工程	Crypto_Selftest
场景2	目标程序，明文，运行在外外Flash	ExtMem_Boot： 从片上Flash首地址启动，配置OSPI 外设，跳转到外部Flash地址去执行	Crypto_Selftest_ext_plain：由Crypto_Selftest工程修改过来（调整链接文件，VTOR）
场景3	目标程序，密文，运行在片外Flash	ExtMem_Boot_OTFDEC： 由ExtMem_Boot工程修改过来，添加对OTFDEC外设的配置	Crypto_Selftest_ext_plain工程生成的明文代码bin，经过加密和其他处理后的加密代码bin； 处理过程使用PC上的脚本工具

life.augmented

测试场景1：Crypto_Selftest

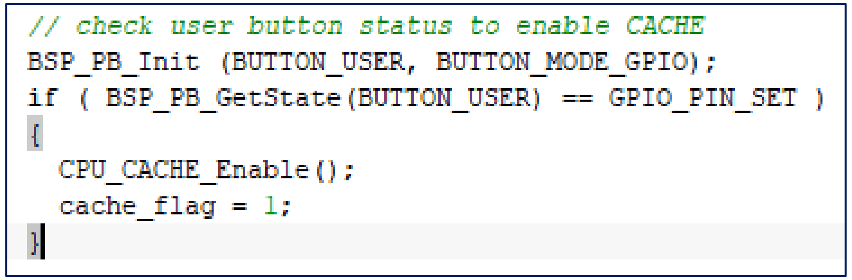
- 主测试体 selftests[]
  - 在<mbedtls_config.h>中选择selftests包含的测试案例



```
/* Run all the tests */
for( test = selftests; test->name != NULL; test++ )
{
    if( test->function( 0 )  != 0 ) // v --> 0
    {
        suites_failed++;
    }
```

TC	Selftest []
1	mbedtls_**md5**_self_test
2	mbedtls_**sha1**_self_test
3	mbedtls_**sha256**_self_test
4	mbedtls_**aes**_self_test
5	mbedtls_**ccm**_self_test
6	mbedtls_**entropy**_self_test_wrapper

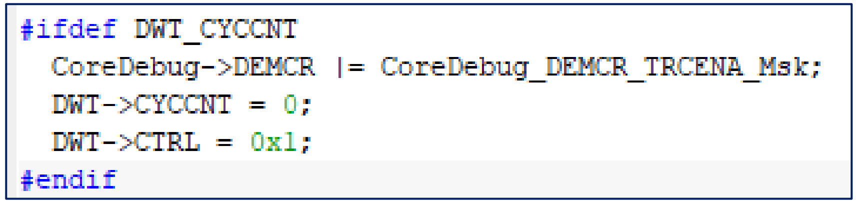
- 复位时检测到用户按键按下，才使能Cache



```
// check user button status to enable CACHE
BSP_PB_Init (BUTTON_USER, BUTTON_MODE_GPIO);
if ( BSP_PB_GetState(BUTTON_USER) == GPIO_PIN_SET )
{
  CPU_CACHE_Enable();
  cache_flag = 1;
}
```

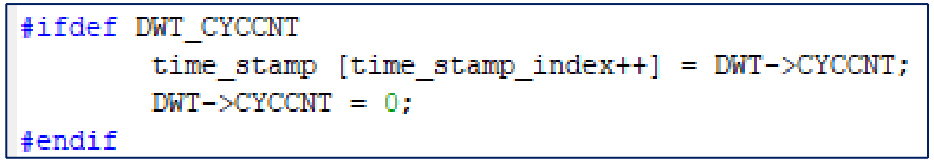
测试场景1：Crypto_Selftest

- 添加时间戳记录
  - 初始化内核计数器



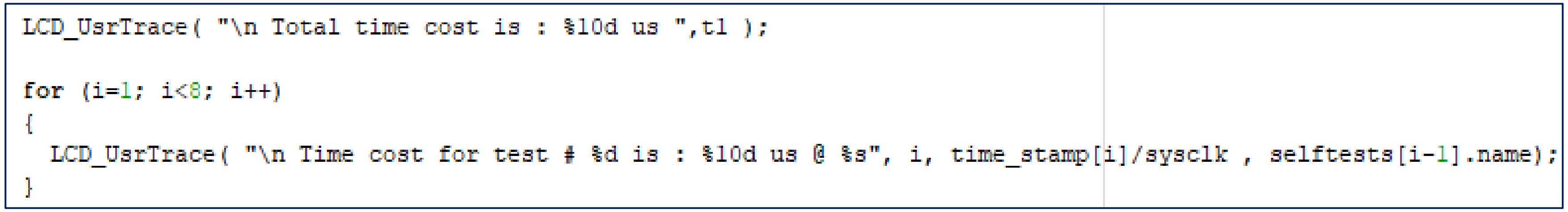
```
#ifdef DWT_CYCCNT
  CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
  DWT->CYCCNT = 0;
  DWT->CTRL = 0x1;
#endif
```

  - 保存该测试案例运行的计数值，并复位计数值



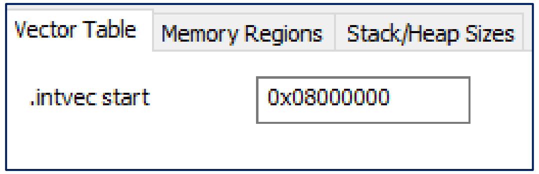
```
#ifdef DWT_CYCCNT
        time_stamp [time_stamp_index++] = DWT->CYCCNT;
        DWT->CYCCNT = 0;
#endif
```

  - 打印总时间花销和各个测试案例的时间花销

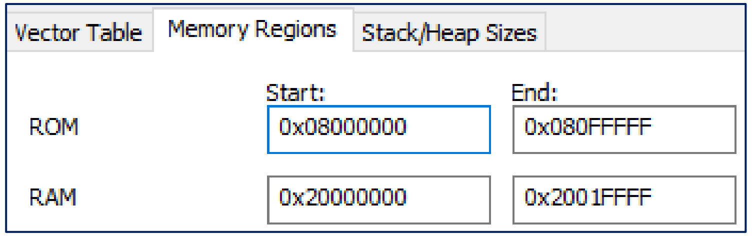


```
LCD_UsrTrace( "\n Total time cost is : %10d us ",t1 );

for (i=1; i<8; i++)
{
  LCD_UsrTrace( "\n Time cost for test # %d is : %10d us @ %s", i, time_stamp[i]/sysclk , selftests[i-1].name);
}
```

- 使用缺省链接文件
  - 从片上Flash首地址启动
  - 代码运行在片上Flash



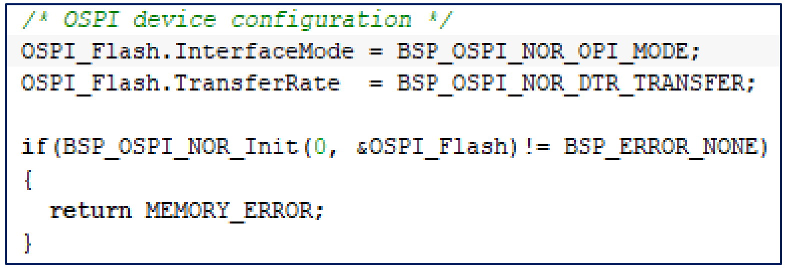
.intvec start: 0x08000000



	Start:	End:
ROM	0x08000000	0x080FFFFF
RAM	0x20000000	0x2001FFFF

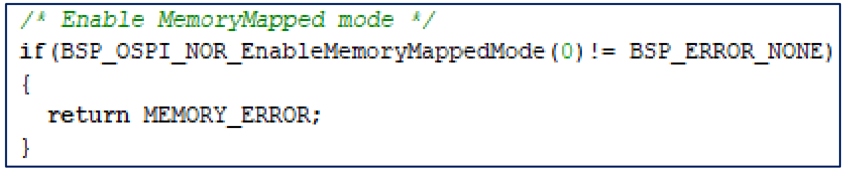
测试场景2

- 启动工程： ExtMem_Boot
  - 使用现成BSP驱动初始化OSPI接口



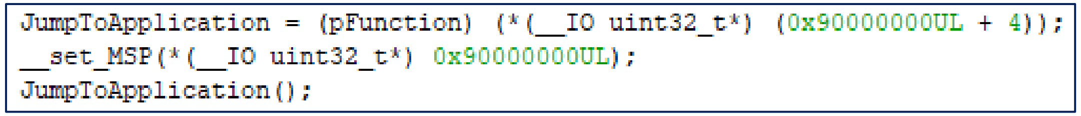
```
/* OSPI device configuration */
OSPI_Flash.InterfaceMode = BSP_OSPI_NOR_OPI_MODE;
OSPI_Flash.TransferRate  = BSP_OSPI_NOR_DTR_TRANSFER;

if(BSP_OSPI_NOR_Init(0, &OSPI_Flash)!= BSP_ERROR_NONE)
{
  return MEMORY_ERROR;
}
```

  - 配置OSPI工作在memory-map模式



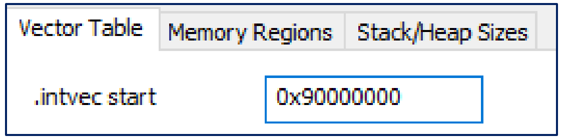
```
/* Enable MemoryMapped mode */
if(BSP_OSPI_NOR_EnableMemoryMappedMode(0)!= BSP_ERROR_NONE)
{
  return MEMORY_ERROR;
}
```

  - 跳转到外部Flash执行

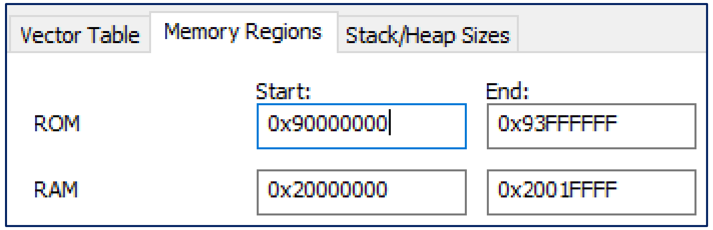


```
JumpToApplication = (pFunction) (*(__IO uint32_t*) (0x90000000UL + 4));
__set_MSP(*(__IO uint32_t*) 0x90000000UL);
JumpToApplication();
```

- 目标测试程序： Crypto_Selftest_ext_plain
  - 基于测试场景1的Crypto_Selftest工程修改
  - 代码运行在片外Flash

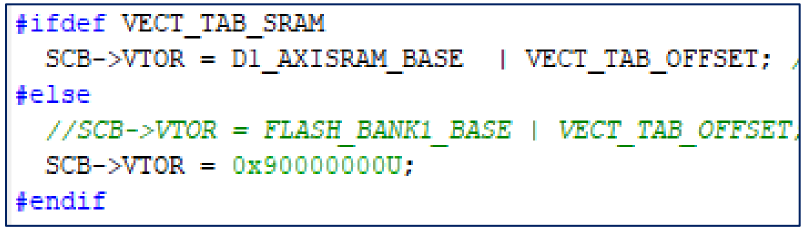


.intvec start 0x90000000



	Start:	End:
ROM	0x90000000	0x93FFFFFF
RAM	0x20000000	0x2001FFFF

  - VTOR做相应修改

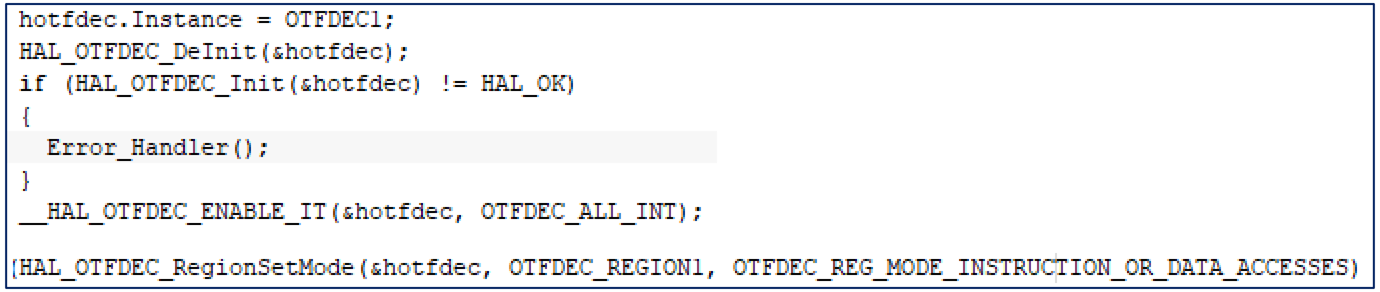


```
#ifdef VECT_TAB_SRAM
  SCB->VTOR = D1_AXISRAM_BASE  | VECT_TAB_OFFSET; /
#else
  //SCB->VTOR = FLASH_BANK1_BASE | VECT_TAB_OFFSET;
  SCB->VTOR = 0x90000000U;
#endif
```

life.augmented

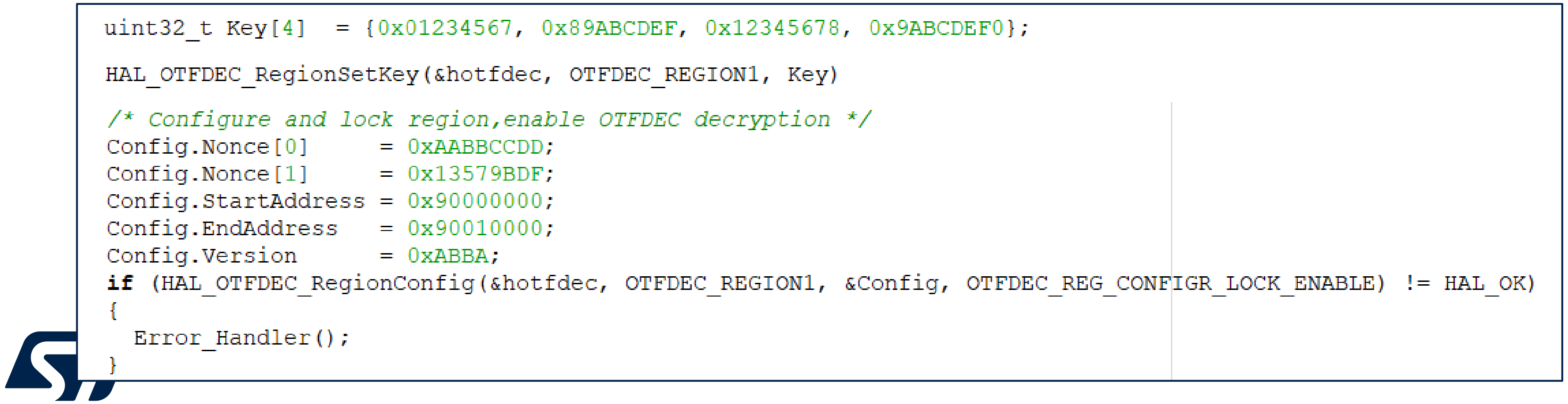
测试场景3

- 启动工程：ExtMem_Boot_OTFDEC
  - 在场景2的ExtMem_Boot工程基础上添加对OTFDEC的初始化



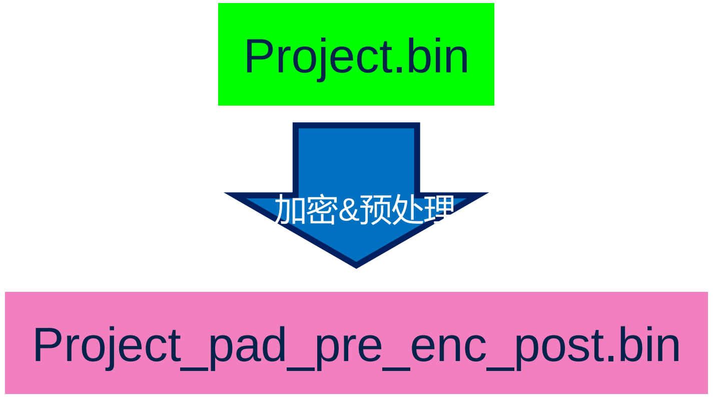
```
hotfdec.Instance = OTFDEC1;
HAL_OTFDEC_DeInit(&hotfdec);
if (HAL_OTFDEC_Init(&hotfdec) != HAL_OK)
{
  Error_Handler();
}
__HAL_OTFDEC_ENABLE_IT(&hotfdec, OTFDEC_ALL_INT);

(HAL_OTFDEC_RegionSetMode(&hotfdec, OTFDEC_REGION1, OTFDEC_REG_MODE_INSTRUCTION_OR_DATA_ACCESSES)
```

  - 对OTFDEC解密参数的设置



```
 uint32_t Key[4]  = {0x01234567, 0x89ABCDEF, 0x12345678, 0x9ABCDEF0};

 HAL_OTFDEC_RegionSetKey(&hotfdec, OTFDEC_REGION1, Key)

 /* Configure and lock region,enable OTFDEC decryption */
 Config.Nonce[0]      = 0xAABBCCDD;
 Config.Nonce[1]      = 0x13579BDF;
 Config.StartAddress = 0x90000000;
 Config.EndAddress   = 0x90010000;
 Config.Version      = 0xABBA;
 if (HAL_OTFDEC_RegionConfig(&hotfdec, OTFDEC_REGION1, &Config, OTFDEC_REG_CONFIGR_LOCK_ENABLE) != HAL_OK)
 {
   Error_Handler();
 }
```

- 使用PC端工具，对测试场景2中跑在外部Flash的代码做加密和其他处理
  - 加密相关参数要和配置OTFDEC的解密参数一致（AES是对称加密算法）
  - 使用Utilities目录下的工具做处理



Project.bin

加密&预处理

Project_pad_pre_enc_post.bin

测试场景3，OTFDEC解密参数的设置

- AES-128-CTR的密钥
  - 用户任意指定
  - 本例程中 uint32_t Key[4] =

  {0x01234567, 0x89ABCDEF, 0x12345678, 0x9ABCDEF0}

OTFDEC 密钥								
地址	**字节 0**	**字节 1**	**字节 2**	**字节 3**	**字节 4**	**字节 5**	**字节 6**	**字节 7**
0x0	0x67	0x45	0x23	0x01	0xEF	0xCD	0xAB	0x89
0x8	0x78	0x56	0x34	0x12	0xF0	0xDE	0xBC	0x9A

- AES-128-CTR的IV，采用如下结构
  - 由三部分组成

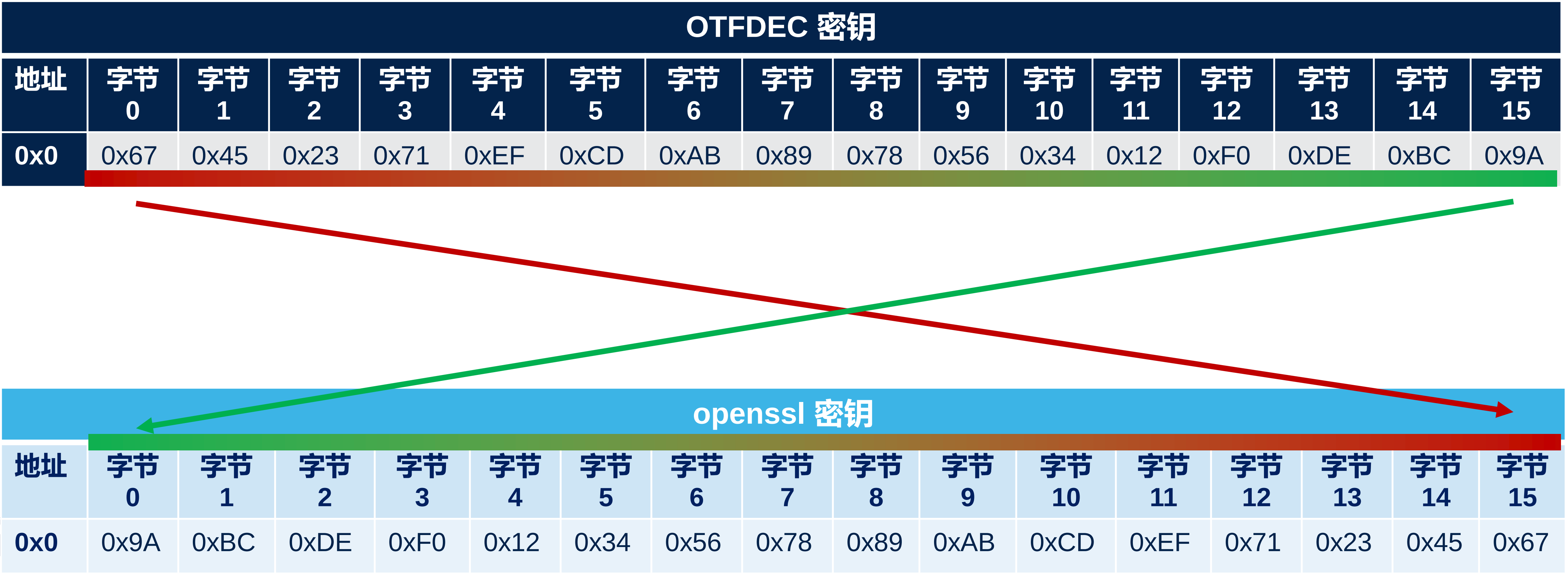
结构体类型 OTFDEC_RegionConfigTypeDe	变量 Config	
Nounce [0]	0xAABBCCDD	用户任意设定
Nounce [1]	0x13579BDF	
StartAddress	0x90000000	所作用的外部存储区范围
EndAddress	0x90010000	
Version	0xABBA	用户任意指定，存储在以上范围内代码的版本标识

测试场景3，PC端加解密工具参数的设置

使用openssl

- openssl命令的密钥

```
openssl.exe enc -aes-128-ctr …… -K 9ABCDEF01234567889ABCDEF01234567 ……
```



OTFDEC 密钥

地址	字节 0	字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	字节 7	字节 8	字节 9	字节 10	字节 11	字节 12	字节 13	字节 14	字节 15
0x0	0x67	0x45	0x23	0x71	0xEF	0xCD	0xAB	0x89	0x78	0x56	0x34	0x12	0xF0	0xDE	0xBC	0x9A

openssl 密钥

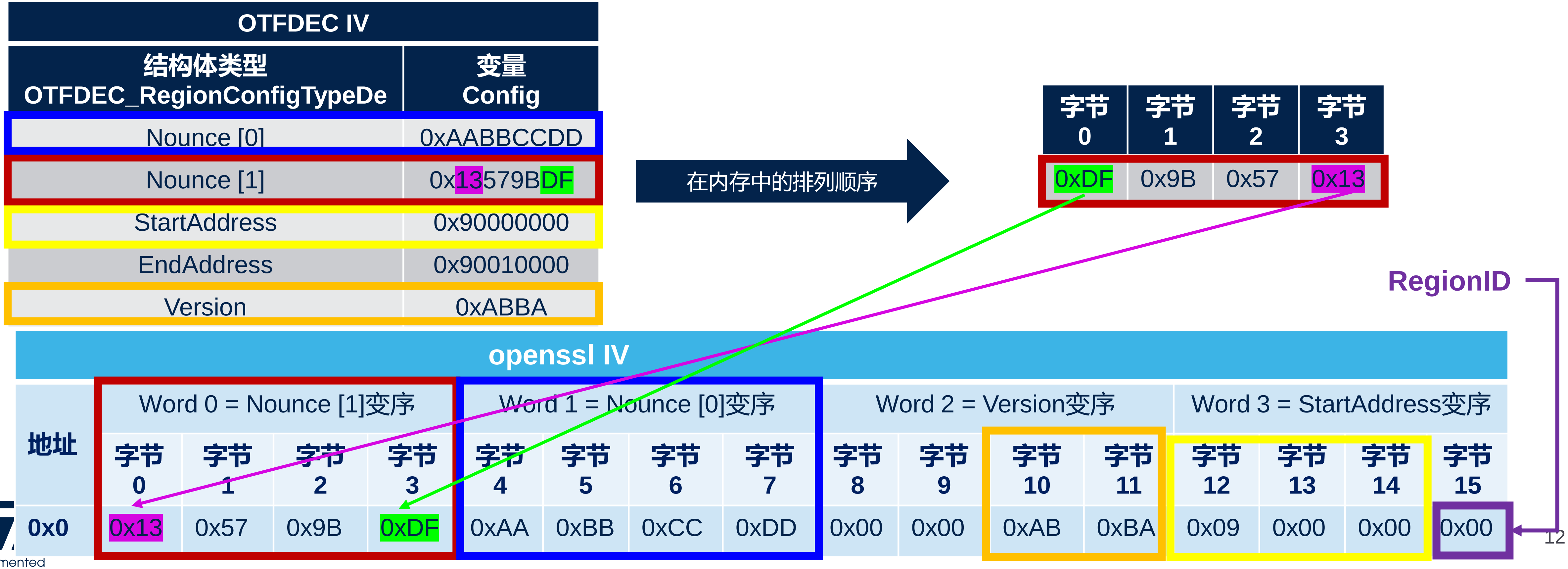
地址	字节 0	字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	字节 7	字节 8	字节 9	字节 10	字节 11	字节 12	字节 13	字节 14	字节 15
0x0	0x9A	0xBC	0xDE	0xF0	0x12	0x34	0x56	0x78	0x89	0xAB	0xCD	0xEF	0x71	0x23	0x45	0x67

测试场景3，PC端加解密工具参数的设置

使用openssl

- openssl命令的IV

```
openssl.exe enc -aes-128-ctr …… -iv 13579BDFAABBCCDD0000ABBA09000000 ……
```



OTFDEC IV	
结构体类型 OTFDEC_RegionConfigTypeDe	变量 Config
Nounce [0]	0xAABBCCDD
Nounce [1]	0x13579BDF
StartAddress	0x90000000
EndAddress	0x90010000
Version	0xABBA

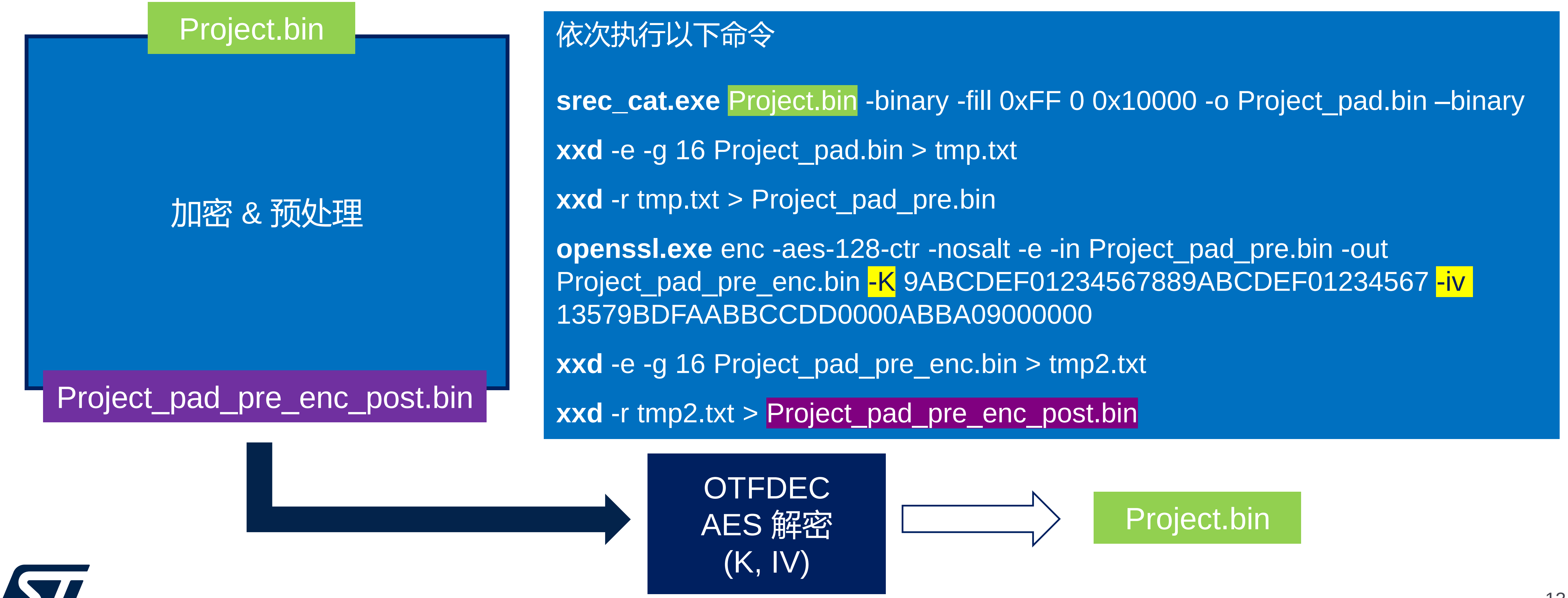
字节 0	字节 1	字节 2	字节 3
0xDF	0x9B	0x57	0x13

openssl IV																
地址	Word 0 = Nounce [1]变序				Word 1 = Nounce [0]变序				Word 2 = Version变序				Word 3 = StartAddress变序			
	字节 0	字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	字节 7	字节 8	字节 9	字节 10	字节 11	字节 12	字节 13	字节 14	字节 15
0x0	0x13	0x57	0x9B	0xDF	0xAA	0xBB	0xCC	0xDD	0x00	0x00	0xAB	0xBA	0x09	0x00	0x00	0x00

测试场景3，PC端加解密工具的消息输入

使用openssl

Crypto_Selftest_ext_plain工程生成的明文代码：Project.bin

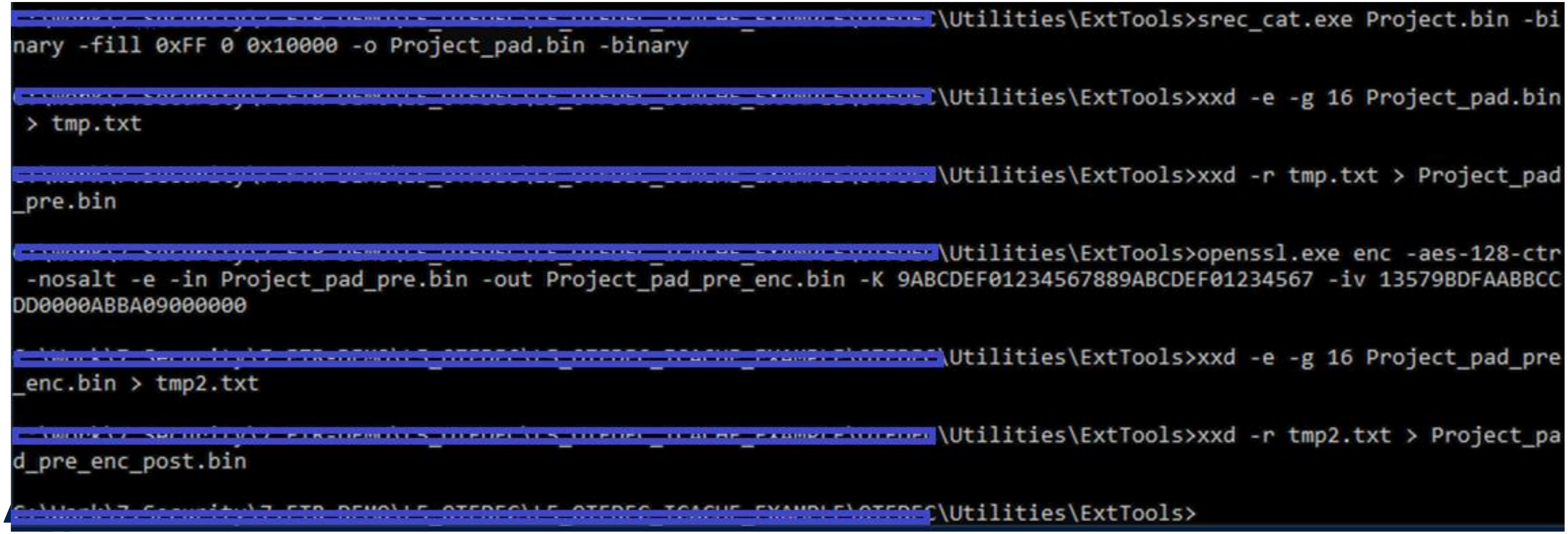


依次执行以下命令

```
srec_cat.exe Project.bin -binary -fill 0xFF 0 0x10000 -o Project_pad.bin –binary

xxd -e -g 16 Project_pad.bin > tmp.txt

xxd -r tmp.txt > Project_pad_pre.bin

openssl.exe enc -aes-128-ctr -nosalt -e -in Project_pad_pre.bin -out
Project_pad_pre_enc.bin -K 9ABCDEF01234567889ABCDEF01234567 -iv
13579BDFAABBCCDD0000ABBA09000000

xxd -e -g 16 Project_pad_pre_enc.bin > tmp2.txt

xxd -r tmp2.txt > Project_pad_pre_enc_post.bin
```

加密 & 预处理

- 切换到如下路径，在命令行窗口依次输入以下命令
  - H7_OTFDEC_Efficiency\Utilities\ExtTools
  - 完成从Project.bin 到 Project_pad_pre_enc_post.bin的 “加密 & 预处理”

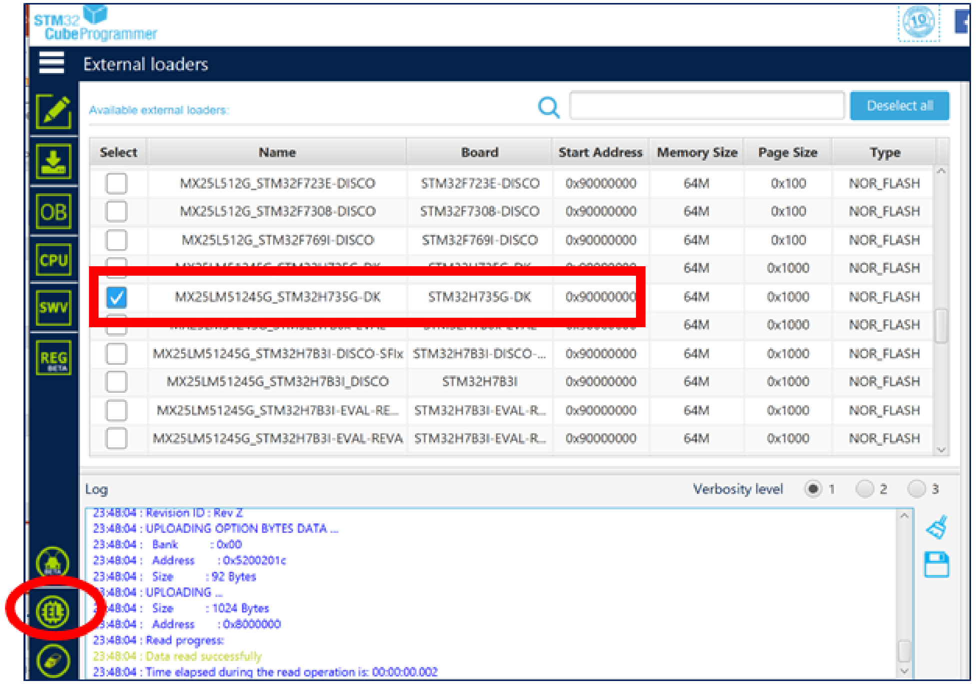


```
[redacted]\Utilities\ExtTools>srec_cat.exe Project.bin -bi
nary -fill 0xFF 0 0x10000 -o Project_pad.bin -binary

[redacted]\Utilities\ExtTools>xxd -e -g 16 Project_pad.bin
 > tmp.txt

[redacted]\Utilities\ExtTools>xxd -r tmp.txt > Project_pad
_pre.bin

[redacted]\Utilities\ExtTools>openssl.exe enc -aes-128-ctr
 -nosalt -e -in Project_pad_pre.bin -out Project_pad_pre_enc.bin -K 9ABCDEF01234567889ABCDEF01234567 -iv 13579BDFAABBCC
DD0000ABBA09000000

[redacted]\Utilities\ExtTools>xxd -e -g 16 Project_pad_pre
_enc.bin > tmp2.txt

[redacted]\Utilities\ExtTools>xxd -r tmp2.txt > Project_pa
d_pre_enc_post.bin

[redacted]\Utilities\ExtTools>
```

验证

【1】把场景3的启动工程ExtMem_Boot_OTFDEC下载到STM32H735G-DK板

【2】使用STM32CubeProgrammer以Under reset的方式把Project_pad_pre_enc_post.bin也下载



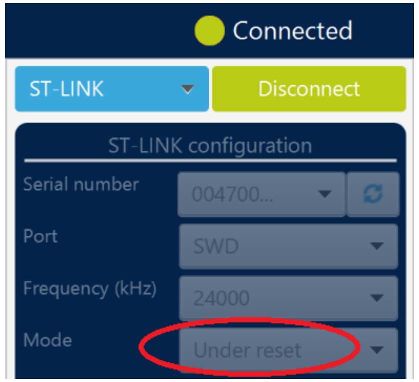
Available external loaders:

Deselect all

Select	Name	Board	Start Address	Memory Size	Page Size	Type
□	MX25L512G_STM32F723E-DISCO	STM32F723E-DISCO	0x90000000	64M	0x100	NOR_FLASH
□	MX25L512G_STM32F7308-DISCO	STM32F7308-DISCO	0x90000000	64M	0x100	NOR_FLASH
□	MX25L512G_STM32F769I-DISCO	STM32F769I-DISCO	0x90000000	64M	0x100	NOR_FLASH
□	[illegible]	[illegible]	[illegible]	64M	0x1000	NOR_FLASH
☑	MX25LM51245G_STM32H735G-DK	STM32H735G-DK	0x90000000	64M	0x1000	NOR_FLASH
□	[illegible]	[illegible]	[illegible]	64M	0x1000	NOR_FLASH
□	MX25LM51245G_STM32H7B3I-DISCO-SFIx	STM32H7B3I-DISCO-...	0x90000000	64M	0x1000	NOR_FLASH
□	MX25LM51245G_STM32H7B3I_DISCO	STM32H7B3I	0x90000000	64M	0x1000	NOR_FLASH
□	MX25LM51245G_STM32H7B3I-EVAL-RE...	STM32H7B3I-EVAL-R...	0x90000000	64M	0x1000	NOR_FLASH
□	MX25LM51245G_STM32H7B3I-EVAL-REVA	STM32H7B3I-EVAL-R...	0x90000000	64M	0x1000	NOR_FLASH

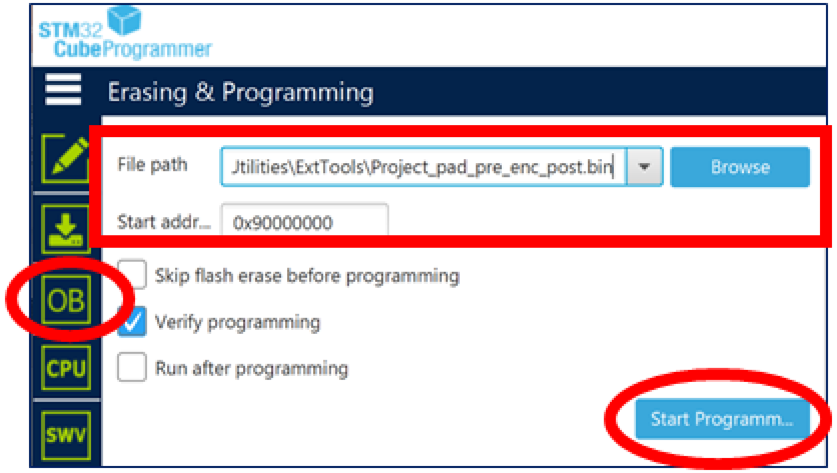
Log   Verbosity level ● 1 ○ 2 ○ 3

```
23:48:04 : Revision ID : Rev Z
23:48:04 : UPLOADING OPTION BYTES DATA ...
23:48:04 :   Bank        : 0x00
23:48:04 :   Address     : 0x5200201c
23:48:04 :   Size        : 92 Bytes
23:48:04 : UPLOADING ...
23:48:04 :   Size        : 1024 Bytes
23:48:04 :   Address     : 0x8000000
23:48:04 : Read progress:
23:48:04 : Data read successfully
23:48:04 : Time elapsed during the read operation is: 00:00:00.002
```



ST-LINK configuration

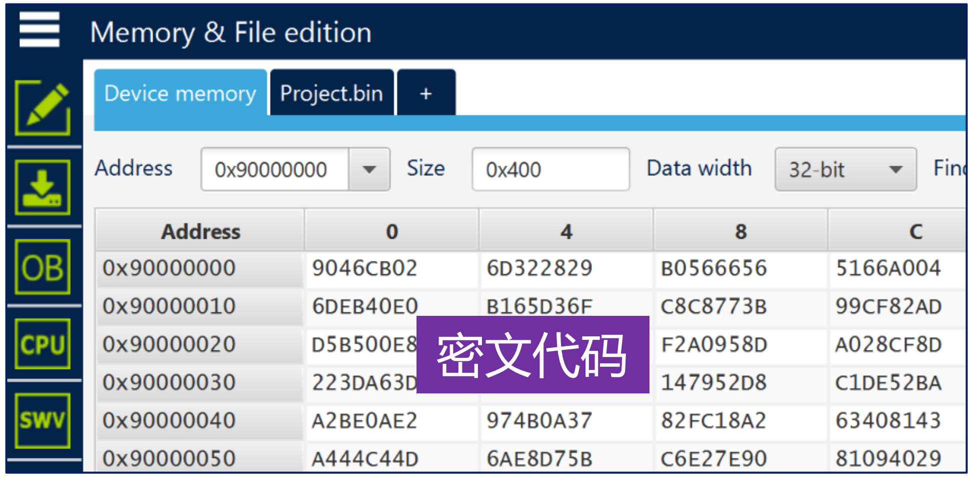
Serial number: 004700...
Port: SWD
Frequency (kHz): 24000
Mode: Under reset



File path: Jtilities\ExtTools\Project_pad_pre_enc_post.bin
Start addr...: 0x90000000

□ Skip flash erase before programming
☑ Verify programming
□ Run after programming

【3】查看0x9000 0000处的加密代码



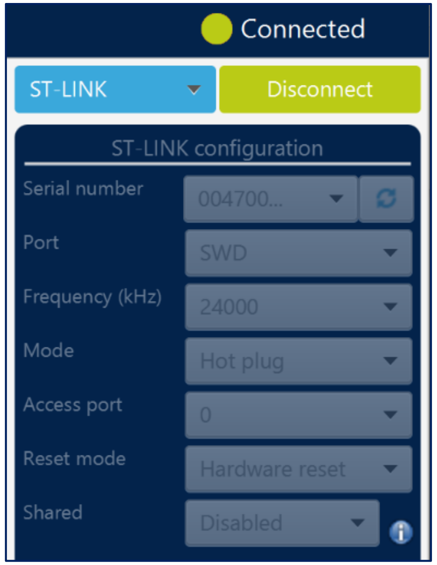
Address: 0x90000000   Size: 0x400   Data width: 32-bit

Address	0	4	8	C
0x90000000	9046CB02	6D322829	B0566656	5166A004
0x90000010	6DEB40E0	B165D36F	C8C8773B	99CF82AD
0x90000020	D5B500E8	[illegible]	F2A0958D	A028CF8D
0x90000030	223DA63D	[illegible]	147952D8	C1DE52BA
0x90000040	A2BE0AE2	974B0A37	82FC18A2	63408143
0x90000050	A444C44D	6AE8D75B	C6E27E90	81094029

life.augmented

验证

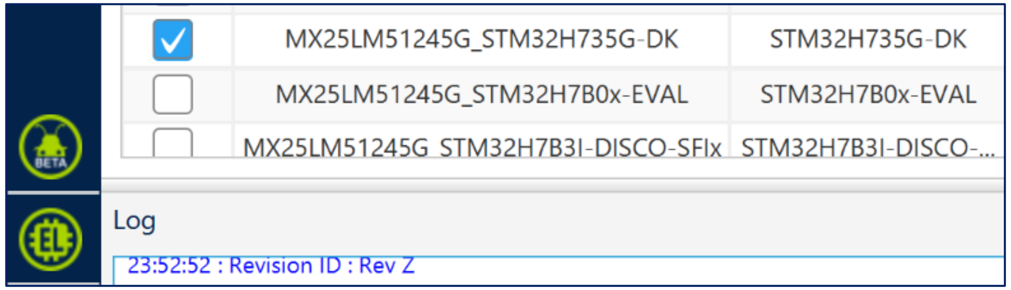
【4】使用STM32CubeProgrammer，以hotplug方式连接板子



ST-LINK configuration

Serial number	004700...
Port	SWD
Frequency (kHz)	24000
Mode	Hot plug
Access port	0
Reset mode	Hardware reset
Shared	Disabled

【5】使能板载OSPI Flash的Loader

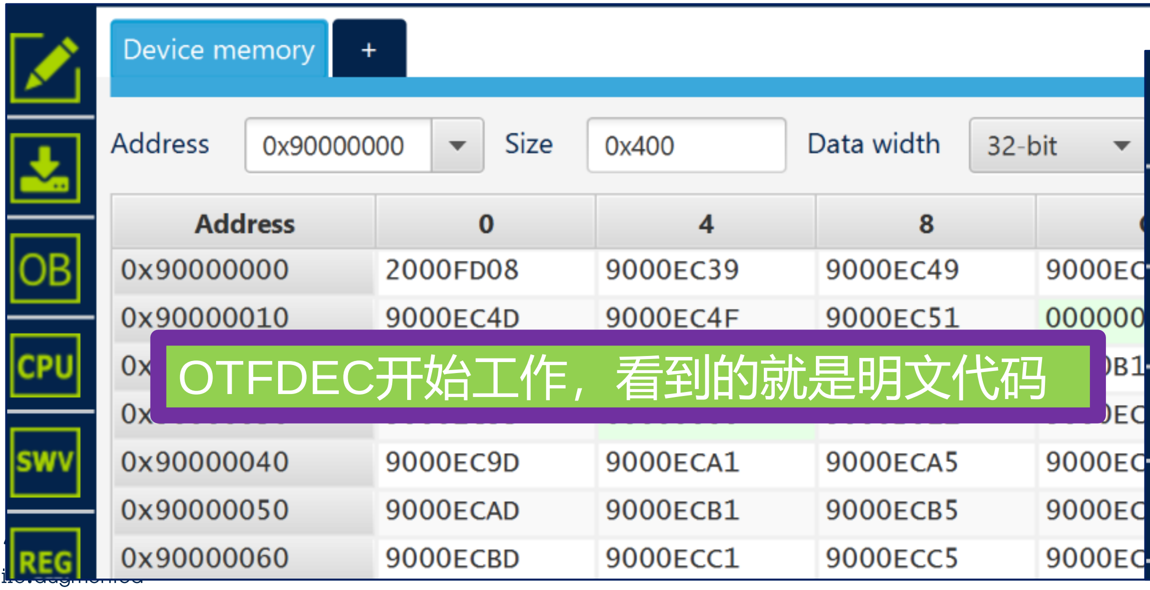


☑	MX25LM51245G_STM32H735G-DK	STM32H735G-DK
□	MX25LM51245G_STM32H7B0x-EVAL	STM32H7B0x-EVAL
□	MX25LM51245G_STM32H7B3I-DISCO-SFIx	STM32H7B3I-DISCO-...

Log

```
23:52:52 : Revision ID : Rev Z
```

【6】查看0x9000 0000的内容

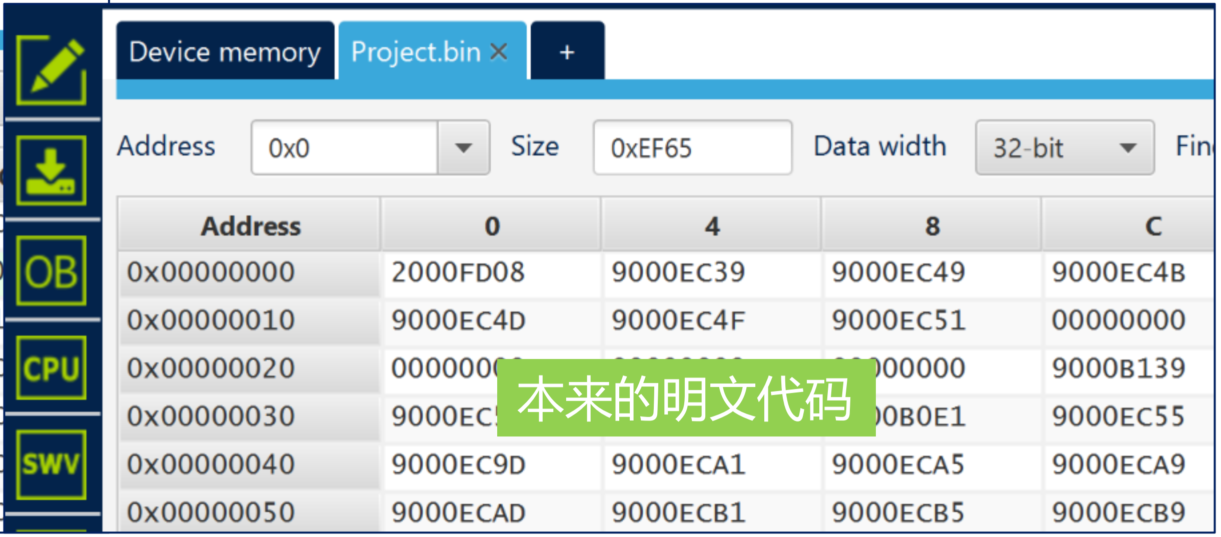


Address 0x90000000 Size 0x400 Data width 32-bit

Address	0	4	8	C
0x90000000	2000FD08	9000EC39	9000EC49	9000EC
0x90000010	9000EC4D	9000EC4F	9000EC51	000000
0x				B1
0x				EC
0x90000040	9000EC9D	9000ECA1	9000ECA5	9000EC
0x90000050	9000ECAD	9000ECB1	9000ECB5	9000EC
0x90000060	9000ECBD	9000ECC1	9000ECC5	9000EC

OTFDEC开始工作，看到的就是明文代码

【7】打开Project.bin，查看内容



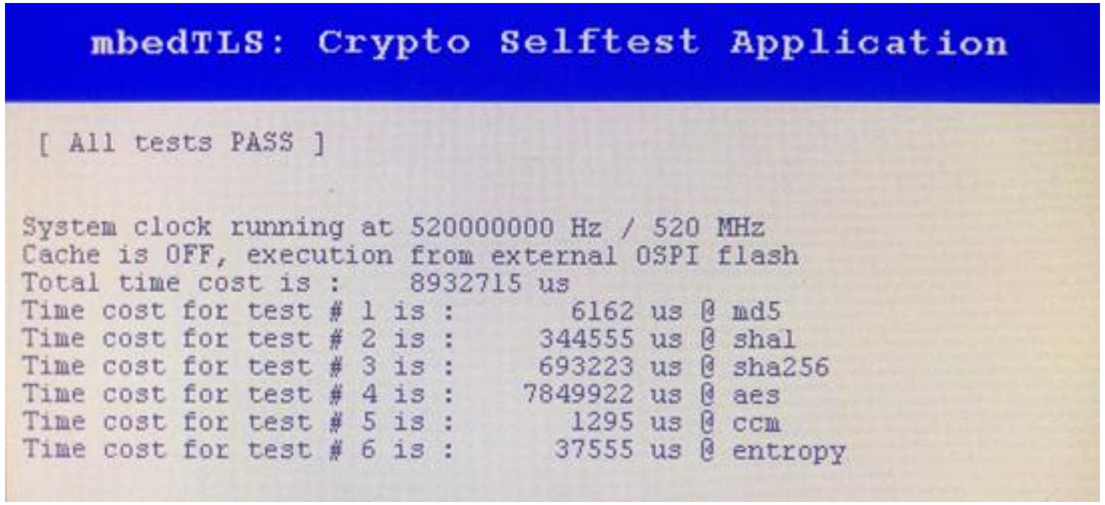
Address 0x0 Size 0xEF65 Data width 32-bit Fin

Address	0	4	8	C
0x00000000	2000FD08	9000EC39	9000EC49	9000EC4B
0x00000010	9000EC4D	9000EC4F	9000EC51	00000000
0x00000020	000000		000000	9000B139
0x00000030	9000EC		0B0E1	9000EC55
0x00000040	9000EC9D	9000ECA1	9000ECA5	9000ECA9
0x00000050	9000ECAD	9000ECB1	9000ECB5	9000ECB9

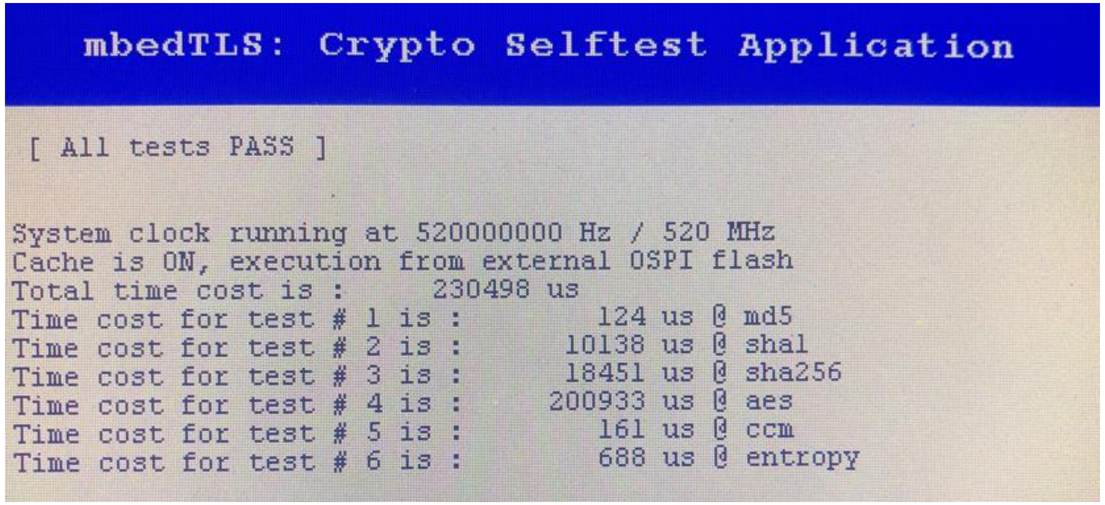
本来的明文代码

板子脱机运行

测试场景	描述	Demo Project	
		启动工程	目标测试工程
场景3	目标程序，密文，运行在片外Flash	ExtMem_Boot_OTFDEC：由ExtMem_Boot工程修改过来，添加对OTFDEC外设的配置	Crypto_Selftest_ext_plain工程生成的明文代码bin，经过加密和其他处理后的加密代码bin； 处理过程使用PC上的脚本工具



```
mbedTLS: Crypto Selftest Application

[ All tests PASS ]

System clock running at 520000000 Hz / 520 MHz
Cache is OFF, execution from external OSPI flash
Total time cost is :    8932715 us
Time cost for test # 1 is :       6162 us @ md5
Time cost for test # 2 is :     344555 us @ sha1
Time cost for test # 3 is :     693223 us @ sha256
Time cost for test # 4 is :    7849922 us @ aes
Time cost for test # 5 is :       1295 us @ ccm
Time cost for test # 6 is :      37555 us @ entropy
```



```
mbedTLS: Crypto Selftest Application

[ All tests PASS ]

System clock running at 520000000 Hz / 520 MHz
Cache is ON, execution from external OSPI flash
Total time cost is :      230498 us
Time cost for test # 1 is :         124 us @ md5
Time cost for test # 2 is :       10138 us @ sha1
Time cost for test # 3 is :       18451 us @ sha256
Time cost for test # 4 is :      200933 us @ aes
Time cost for test # 5 is :         161 us @ ccm
Time cost for test # 6 is :         688 us @ entropy
```

复位时不按住蓝色用户按键，未开启Cache

复位时按住蓝色用户按键，开启Cache

三种场景，运行目标测试程序Crypto_Selftest

TC	function	Iteration	us @ 520MHz Cache OFF/ON		
			Case 1 Int. flash	Case 2 Ext. flash/plain	Case 3 Ext. flash/cypher
1	mbedtls_**md5**_self_test	7	108 73	6,162 125	6,162 126
2	mbedtls_**sha1**_self_test	3	19,263 9,826	344,642 10,137	344,621 10,138
3	mbedtls_**sha256**_self_test	6	38,587 17,732	693,545 18,435	693,511 18,437
4	mbedtls_**aes**_self_test		471,244 197,498	7,748,346 199,403	7,748,102 199,300
5	mbedtls_**ccm**_self_test	3	103 36	1,286 157	1,288 156
6	mbedtls_**entropy**_self_test_wrapper		844 577	37,557 685	37,552 684
all			530,151 225,745	8,831,541 228,943	8,831,239 228,844

结论：代码运行在外部Flash的时候，运行明文和使用OTFDEC运行密文，效率相差无几；要提高代码运行在外部Flash的效率，主要加速措施是使能内核自动的Cache

Thank you

life.augmented

For further support in creating a PowerPoint presentation, including graphic assets, formatting tools and additional information on the ST brand

you can visit the ST Brand Portal
https://brandportal.st.com

life.augmented