STM32 Cryptolib 使用技巧——AES GCM 解密认证失败问题的研究

关键字：Cryptolib，AES，解密

1. 引言

X-CUBE-CRYPTOLIB 是基于 STM32 的 Crypto 算法库，支持对称密钥、非对称密钥、哈希等多种算法。正确地使用 Cyrptolib 算法库，可以在应用程序中实现数据加密、设备身份认证、加密通信等多种应用层所需的安全功能。相反，若不能正确地使用算法库往往会带来加解密数据错误等系列问题。

关于 STM32 Crypto 算法库应用中的常见的问题之一就是应用程序没有使能 MCU 的 CRC 模块，尽管输出的数据和期望值不同，但加解密函数的调用并未返回异常。本文在此描述另外一种没有正确使用算法库的情况。

2. 问题描述

客户应用项目中需要在固件更新过程中对固件进行加密并验证，根据推荐采用了 AES-GCM 算法完成该任务。下载的固件通过 AES-GCM 进行加密，并带 TAG 可以用于验证固件来源的合法性。在项目的代码中使用 X-Cube-Cryptolib 进行 AES-GCM 运算。上位机使用 cryptopp820 加密库对固件目标 bin 文件进行加密，然后在 MCU 上通过 X-Cube-Cryptolib 加密库进行 AES-GCM128 解密，解密数据没有问题，但是 TAG 数据总是无法校验通过。

3. 问题分析与定位

正常情况下，当调用 AES_GCM_Decrypt_Finish(P_pAESGCMctx, NULL, P_pOutputSize);执行后，AESctx.mFlags 结果会提示是否通过校验。如果校验成功，该值应该等于 0x22，而运行结果中看到的却是 0x12。

确认库函数使用方法

将调用 AES-GCM 功能的代码放在 X-Cube-Cryptolib 中一个简单的测试程序的环境进行测试，查看是否有该问题，结果发现测试程序中 AES-GCM 校验是可以成功的，但是集成到客户应用中时就无法成功。那我们接下来重点研究应用程序环境。

应用程序环境

应用程序使用了 FreeRTOS，基于 IAR 编译环境。

查看库文件的使用

确认使用了正确的库文件。

确认是否存在多线程访问

AES-GCM 的函数会在几个线程中调用，而且确认不会出现同时调用的情况，不存在 raise condition 的问题。

查看内存使用情况

最初怀疑是否因为任务栈溢出造成，于是查看内存使用情况。

- IAR stack size: 0x4800
- IAR heap size: 0x4000
- FreeRTOS heap size: 85KB
- 执行 AES 运算的线程 stack size: 2560B

通过 FreeRTOS 的 uxTaskGetStackHighWaterMark() 函数查看该线程还有 500 字节左右剩余空间。

AESGCMctx_stt 结构的大小有 2360 字节，AES-GCM 加解密函数需要的 stack 大小大概在 450 字节左右，但是应用代码中将该变量定义为全局变量，以便可以在几个不同的线程中使用，这样可以确认线程栈大小没有问题，不存在 stack overflow 的问题。

查看生成代码的.map 文件

通过比较，所有 cryptolib 中的 symbol 在 map 中的大小都正常，唯一有问题的是 **AESGCMctx_stt** 结构变量的大小。

- 应用代码 .map: AESctx 0x2002f1f4 0x8f8 Data Gb AES_GCM_Decrypt.o [1]
- 正常测试代码 .map: AESctx 0x20002e64 0x938 Data Gb AES_GCM.o [1]

验证不过的问题应该和这个结构的数据有直接关系，接下来研究是什么造成了这个不同。

查看项目使用的 crypto 库头文件

经检查，INCLUDE_AES192 和 INCLUDE_AES256 两个宏定义在 config.h 的定义中被注释掉，这将导致 **aes_gcm.h** 中 **AESGCMctx_stt** 数据结构的成员变量

uint32_t amExpKey[CRL_AES_MAX_EXPKEY_SIZE];的大小发生变化，因为 CRL_AES_MAX_EXPKEY_SIZE 的定义根据 INCLUDE_AES128/192/256 是否定义会有所不同。

```
//#define INCLUDE_DES ((uint16_t)0x0001) /*!< DES functions are
included in the library. */
//#define INCLUDE_TDES ((uint16_t)0x0002) /*!< TripleDES (TDES)
functions are included in the library. */
#define INCLUDE_AES128 ((uint16_t)0x0004) /*!< AES functions with key
size of 128 bit are included in the library. */
//#define INCLUDE_AES192 ((uint16_t)0x0008) /*!< AES functions with
key size of 192 bit are included in the library. */
```

```
//#define INCLUDE_AES256 ((uint16_t)0x0010) /*!< AES functions with
key size of 256 bit are included in the library. */
//#define INCLUDE_ARC4 ((uint16_t)0x0020) /*!< ARC4 functions are
included in the library. */
//#define INCLUDE_CHACHA ((uint16_t)0x0040) /*!< ChaCha functions are
included in the library. */
//#define INCLUDE_CHACHA20POLY1305 ((uint16_t)0x0080) /*!< oly1305-
AES functions are included in the library */
```

4. 问题解决

将 config.h 里面被注释掉的两行定义打开，重新编译，此时 TAG 验证能够正常通过。

```
#define INCLUDE_AES128 ((uint16_t)0x0004) /*!< AES functions with
key size of 128 bit are included in the library. */
#define INCLUDE_AES192 ((uint16_t)0x0008) /*!< AES functions with
key size of 192 bit are included in the library. */
#define INCLUDE_AES256 ((uint16_t)0x0010) /*!< AES functions with
key size of 256 bit are included in the library. */
```

5. 小结

简言之，如果使用 X-Cube-Cryptolib 库的话，作为用户就不要改动 config.h 的内容，不可想当然地自行调整配置。其实，库是预先编译好了的，所有的功能都已经包含，只是链接的时候根据用户使用到的函数去链接最终的目标文件。这个客户就是按照以为关闭某些配置可以节省代码空间的想法，贸然注释掉了他以为自己不需要的功能，造成数据结构大小发生变化等，影响加密库的正常使用。

版本历史

日期	版本	变更
2021 年 08 月 02 日	1.0	

重要通知 – 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其 各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图等）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。