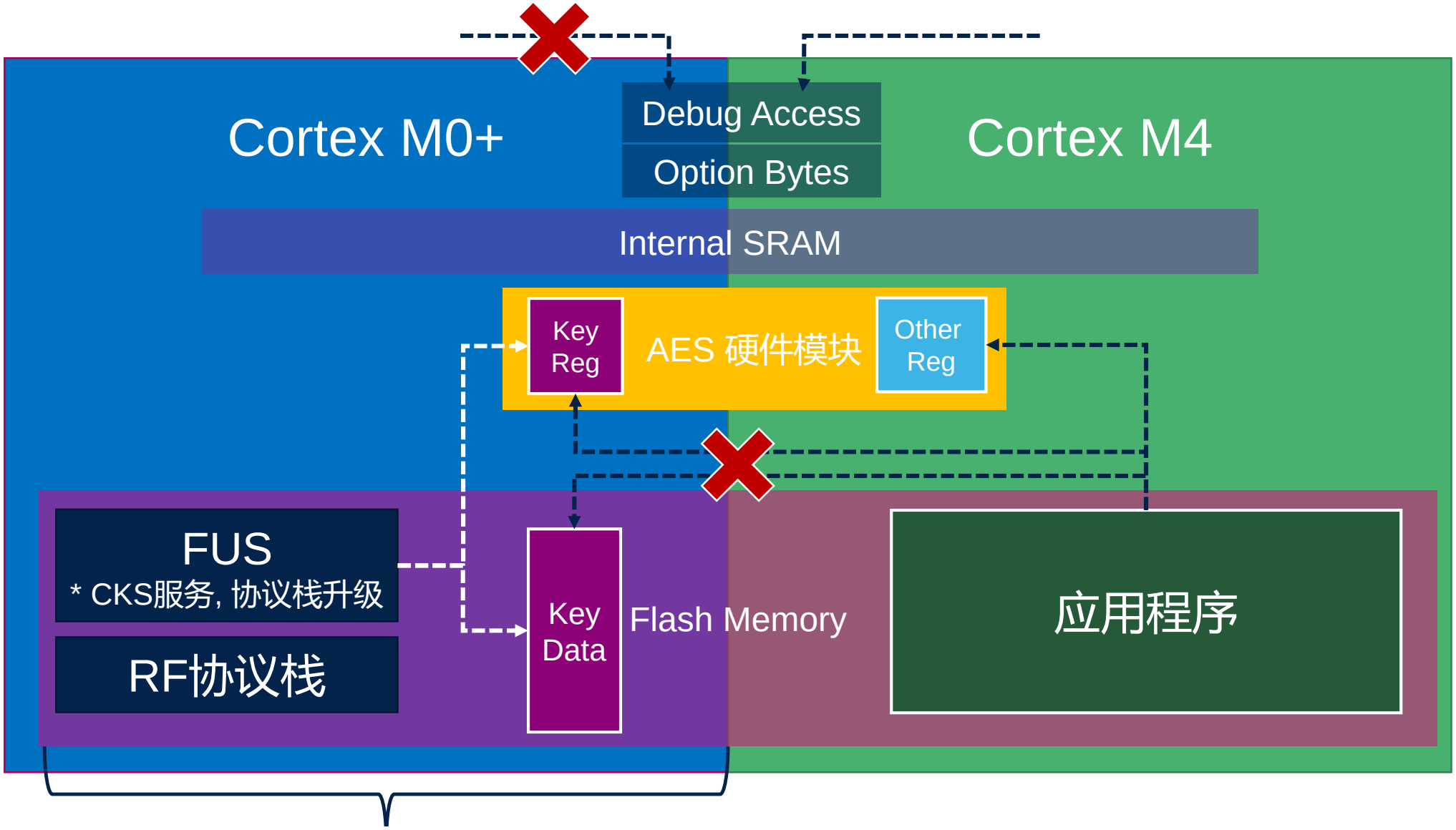
STM32WB的用户密钥存储(CKS)

STMCU 中国团队

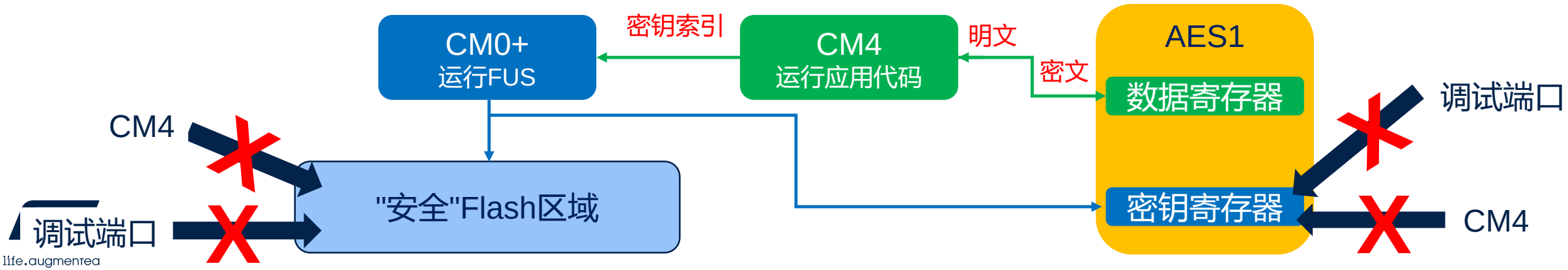
STM32WB双核架构和双核间的隔离机制



这一段Flash只能被CM0+访问：”安全”Flash

CKS (Customer Key Storage)

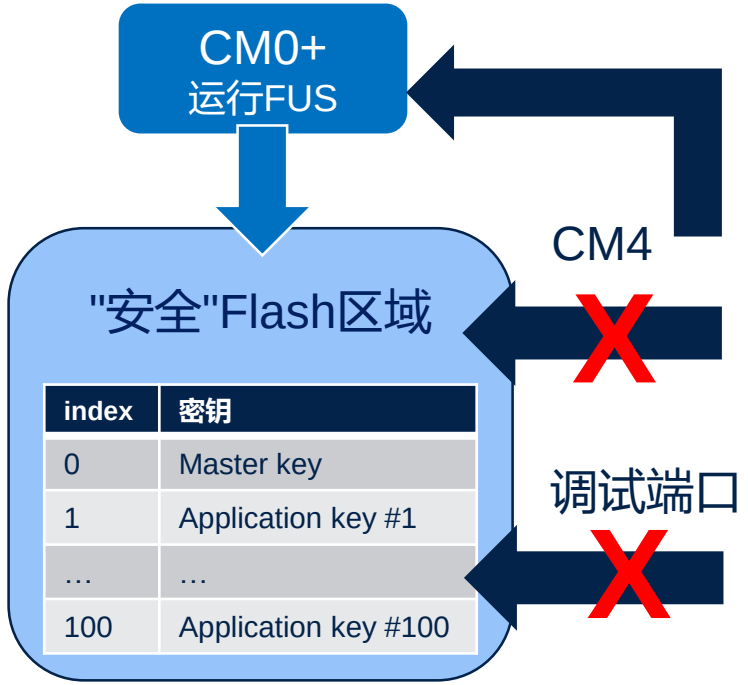
- AES算法是应用程序中经常用到的一种保障数据机密性和完整性的方法，例如用于隐私数据的加密存储、加密通信等。其中，密钥作为最敏感的信息也需要受到保护
- STM32WB的CKS功能提供一种在MCU中保护AES密钥的存储及使用的方法
  - 存储在片上Flash的密钥无法通过调试端口获取（即使在RDP0条件下）
  - 运行在CM4内核的应用程序代码也无法获得内部Flash中存储的密钥，避免软件漏洞带来的风险
  - 应用程序代码依旧能够通过CKS和AES1硬件模块使用存储的密钥进行加解密操作
  - CKS能够存储多组密钥，应用程序代码可以通过密钥索引来指定AES运算所使用的密钥



用户密钥存储

- 用户密钥存储在"安全"Flash区域，用户代码和调试端口不可访问
  - CKS最多可以存储100个用户应用密钥（用于AES运算）
  - 允许存储128位或者256位的AES密钥
- 对密钥的操作只能通过用户代码调用FUS接口完成
  - 把密钥写到“安全”Flash区域，需要安全的操作环境

CKS功能	对密钥的操作	参数
Store/Write key provisioning	用户把密钥写到"安全"Flash区域	In：密钥类型、密钥数据本身 Out：分配的该密钥索引
Load	用户把指定密钥装载到AES1的密钥寄存器中	In：密钥索引
Lock	在下次复位之前指定的密钥不能被装载到AES1密钥寄存器中	In：密钥索引



life.augmented

用户密钥的写入 (Key Provisioning)

- 可以通过用户代码完成，也可以通过CubeProgrammer的GUI或者命令行完成
  - 写入应用密钥的明文//simple key，需要安全的操作环境
  - 写入应用密钥的密文//encrypted key，无需安全的操作环境
    - 基于AES-128 GCM
  - 写入用于解密encrypted key的密钥的明文//master key，需要安全的操作环境
- 该操作在手册中用“write”或“load”表示

用户密钥的写入 (Key Provisioning)
通过应用代码实现

- 参考例程：
  - STM32Cube_FW_WB_Vxxx\Projects\P-NUCLEO-WB55.Nucleo\Applications\CKS\
  - 给CKS_param赋值

```
typedef PACKED_STRUCT{
 uint8_t KeyType;
 uint8_t KeySize;
 uint8_t KeyData[32 + 12];
 } SHCI_C2_FUS_StoreUsrKey_Cmd_Param_t;

SHCI_C2_FUS_StoreUsrKey_Cmd_Param_t CKS_param;
CKS_param.KeyType = …;
CKS_param.KeySize = …;
memcpy(CKS_param.KeyData, pKeySimple_128, 16)
```

128位/16字节 密钥本身

```
static const uint8_t pKeySimple_128[16] = {
  0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2, 0xA6,
  0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C
};
```

  - 调用FUS服务：SHCI_C2_FUS_StoreUsrKey
    - 输入：CKS_param结构体
    - 输出：芯片为该密钥分配的索引

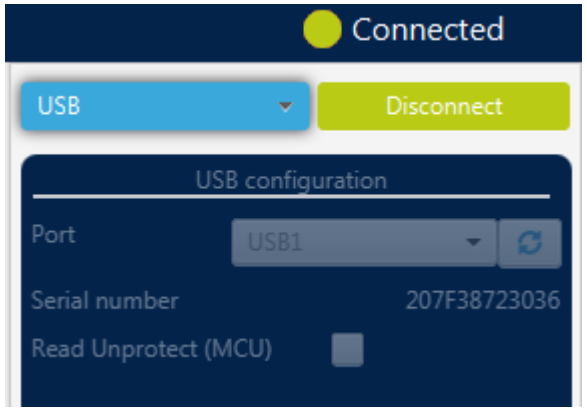
```
SHCI_C2_FUS_StoreUsrKey (&CKS_param, &key_simple_128_idx );
```

life.augmented

用户密钥的写入 (Key Provisioning)
使用STM32CubeProgrammer GUI实现

- STM32CubeProgrammer从2.4版本开始支持
  - 芯片切换到系统Bootloader启动运行DFU，
  - STM32CubeProgrammer以USB方式连接芯片

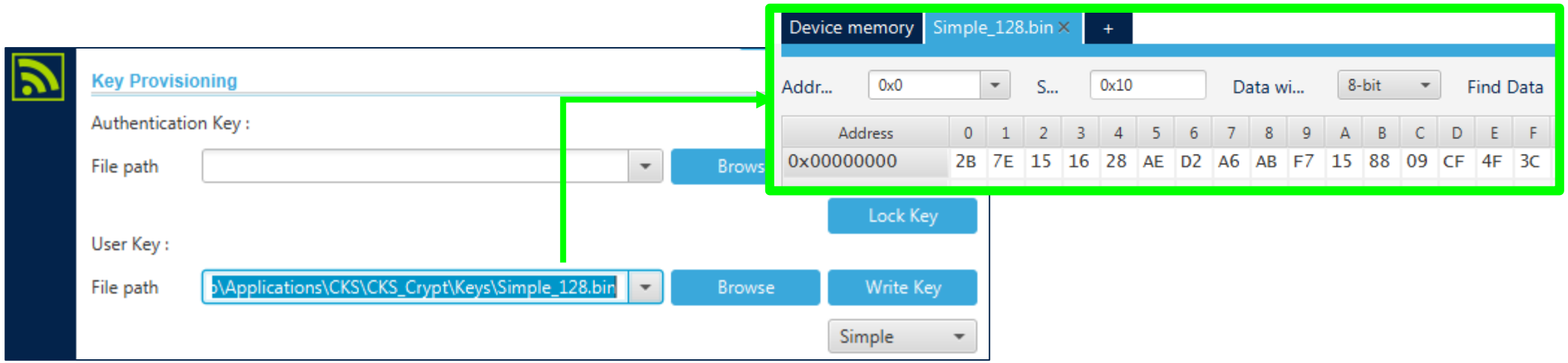
  } 如左图



  - 选择KEY文件，
  - 指定KEY类型，
  - 写入KEY

  } 如下图

- 注意：GUI界面没有返回为该密钥分配的索引，需要用户自己记录



应用密钥的装载和使用

- Load：AES1的密钥装载
  - SHCI_C2_FUS_LoadUsrKey (key_simple_128_idx)
  - 该操作会把AES1密钥寄存器配置成Secure
    - SAES1@SYSCFG_SIPCR
    - 只能由运行在CM0+上的FUS来配置
    - 运行在CM4上的用户代码可以读取其状态
    - 安全状态和AES1时钟是否使能没有关系

CKS功能	参数
Store/Write key provisioning	In：密钥类型、密钥数据本身 Out：分配的该密钥索引
Load	In：密钥索引
Lock	In：密钥索引

- 使用步骤
  - Step1：AES1模块初始化
    - 初始化结构体的pKey，设置为空指针即可
    - 使能AES1时钟
  - Step2：密钥装载
    - 要在AES1模块disable1的时候 (EN=0)
  - Step3：使用AES1做加解密

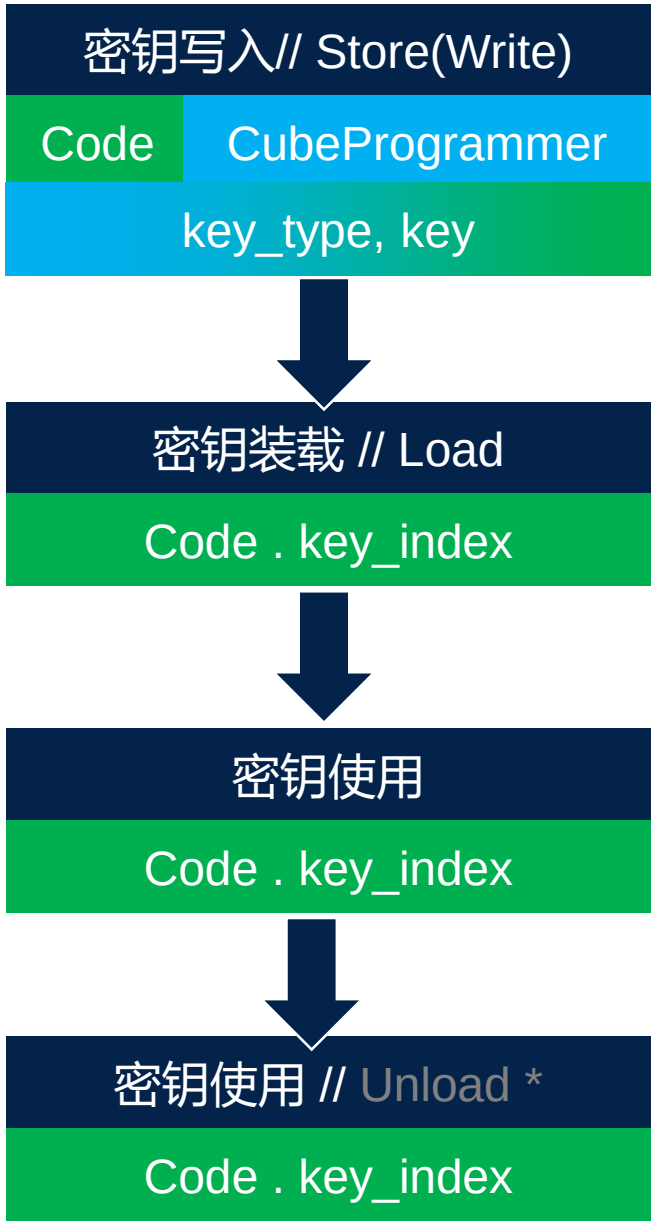
```
hcryp1.Init.DataType      = CRYP_DATATYPE_8B;
hcryp1.Init.KeySize       = CRYP_KEYSIZE_128B;
hcryp1.Init.Algorithm     = CRYP_AES_CBC;
/* Key will be provided by CKS service */
hcryp1.Init.pKey          = NULL;
hcryp1.Init.pInitVect     = AESIV;

HAL_CRYP_Init (&hcryp1);

SHCI_C2_FUS_LoadUsrKey (key_simple_128_idx);

HAL_CRYP_Encrypt(&hcryp1, Plaintext, size, EncryptedBuf, timeout)
```

应用密钥的使用

- Lock：
  - SHCI_C2_FUS_LockUsrKey (key_simple_128_idx)
  - 对某个key lock之后，将无法再对该key进行Load操作，再次Load该Key时FUS API返回错误代码0xFF；但是不影响已经在AES1密钥寄存器中的key
  - 对该key load的禁止操作，会一直生效直到下次系统复位

- 注意事项：关于后续更新
  - 为了避免有效密钥一直存在于AES1密钥寄存器中，建议使用完毕后Load一个dummy key或者disable AES1
    - 后续FUS版本(从STM32CUbeWB1.11开始)会增加API： Unload
  - 早期出厂芯片预装的FUS版本较老，在FUS升级的时候，通过CSK存储在芯片里的用户key会被擦除
    - 从FUS1.1.1.1或者FUS1.2.0开始，再做FUS升级，不会影响到已经存在的用户key

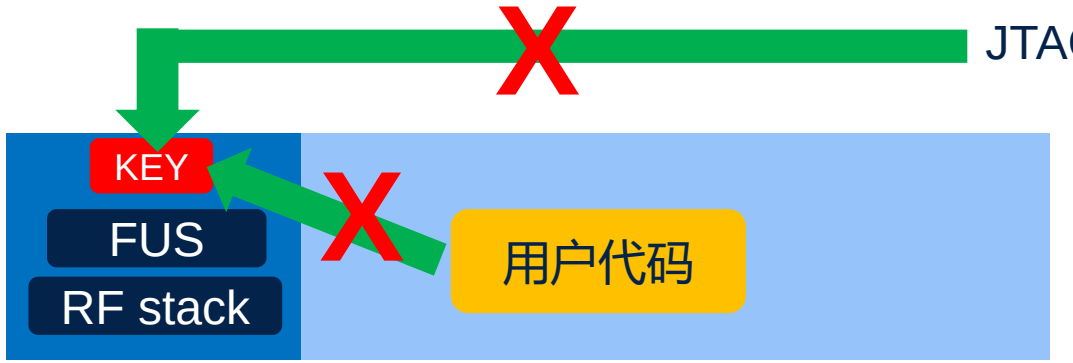
小结



* 在STM32CubeWB1.11中的FUS将会支持

- 使用CKS可以大幅度提高用户密钥存储和使用的安全性

常规方式，密钥直接存储在user flash上
存在密钥通过调试端口或者被恶意代码获取的风险



使用CKS，密钥存储在“安全”Flash区域
密钥无法通过调试端口或应用程序直接访问

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、更正、增强、修改和改进的权利，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。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。

Thank you

life.augmented