加密芯片那些事儿

作者：武者

目录

一、 为什么要用加密芯片

原因很简单：MCU 很容易被破解！ 图一为本人定期收到一家芯片解密公司的广告。不断有新的芯片被解密成功。破解一颗芯片，便宜的几百元，贵的几万元。记得 15 年前 Cypress 的一款 USB 芯片 Cy63001，号称不可被破解，当时我也问过市面上各家芯片破解公司，都说无法破解。后来我兴致勃勃地把它开发出来，并快活的收取着 licese 费用。没想到好景不长，只过了 2 年，这颗芯片就可以被破解，破解费 5 万元。这后又过 1 年，这颗芯片的破解费迅速降到 500 元！下降速度超过中国股市……

图 1：定期收到芯片解密广告

这些芯片破解公司可以非常容易的破解 MCU，提取芯片里面的二进制代码，还可以将代码进行反汇编，进行跟踪、调试。

如果在设计方案上加一颗加密芯片，并让主控 MCU 在工作的时候，跟这颗加密芯片有交互。这样即使主控 MCU 被破解，整套方案，没有了这颗加密芯片也是运转不起来的。

那么问题来了，这颗加密芯片是否安全可靠，是否能够不被破解？这就下面要讲的内容。

二、 加密芯片安全性考虑要素

加密芯片承载着整套方案的安全重任，其本身是否安全可靠，至关重要。衡量一颗加密芯片是否足够安全，主要考虑 2 方面：硬件、软件。

市面上的加密芯片五花八门，种类繁多，让我们看的眼花缭乱。一个加密芯片是否足够安全，加密芯片本身的硬件结构，至关重要。如果加密芯片自身能够像主控 MCU 一样被破解，那么整套方案就毫无安全可言。

在加密芯片硬件可靠的基础上，使用的是哪种软件方案也同样重要。有一些加密芯片硬件安全度很高，不可被破解，但使用的软件方案不好，这样也会被搞芯片破解的人，轻而易举的改动主控芯片的二进制码，跳过加密芯片运行，或者在功能上模拟出一样的加密芯片，从而破解整套方案。

一个好的安全加密芯片不但要有安全可靠，不可被破解的物理硬件，还要有可灵活设计的软件。二者缺一不可，否则再好的硬件，会因为软件设计的限制，被破解。再好的软件设计方案，也会因为，硬件安全程度不够，被侵入者全盘复制。

三、 加密芯片的硬件安全性

上世纪 70 年代初期，嵌入式系统是由分离部件如：CPU、ROM、RAM、I/O 缓存、串口和其他通信与控制接口组成的。我们可通过早期的单板机，清楚地看到，如图：

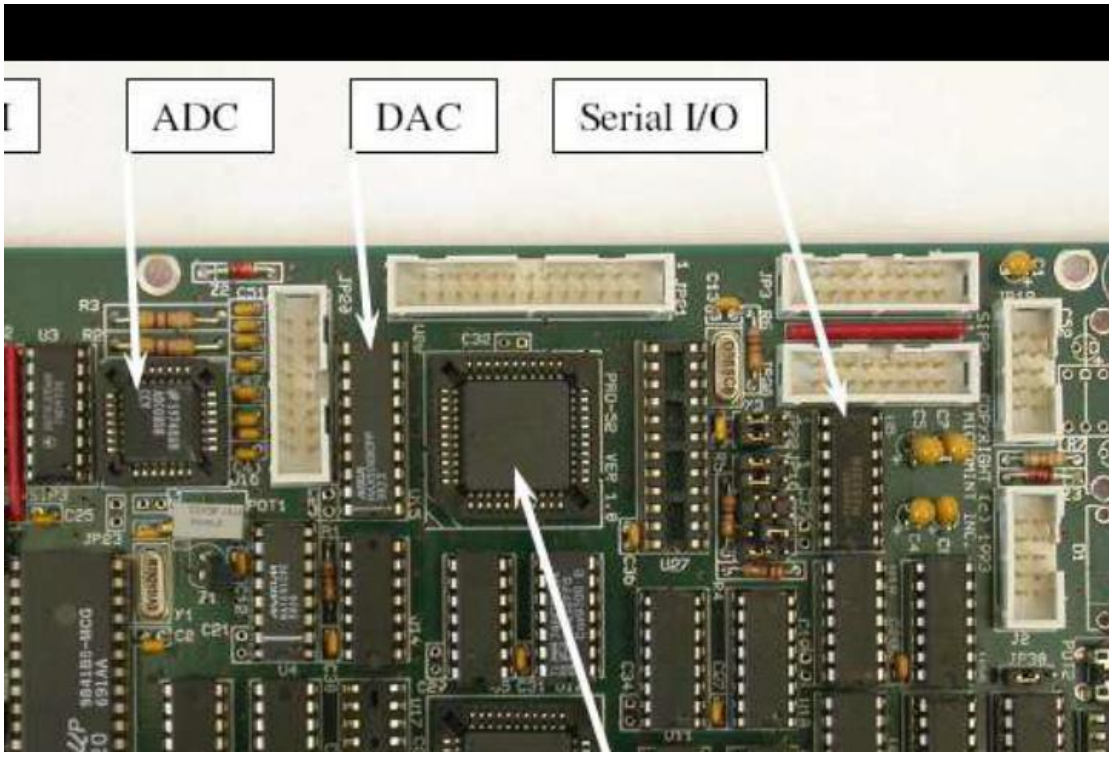

图 2：早期单板机

有一些加密芯片，使用比较偏门的 MCU 实现(甚至一些 MCU 订制厂商，同时也在

卖加密芯片)。这些加密芯片，再偏门，也能够被芯片破解者通过侵入式或非侵入式攻击，轻松得到加密芯片的内部代码，然后破解者从加密芯片厂商购买同样加密芯片，自行烧录，从实现整套方案破解。

目前智能卡芯片内核的加密芯片，最为安全可靠。其他的内核芯片，都有可能被破解。智能卡内核之所以最安全，是因为其使用了存储器总线加密技术，顶层金属网络设计，混合逻辑设计等，并且智能卡芯片提供了很多的防止攻击保护，如：防止电源噪声攻击，时钟噪声攻击等等。有兴趣的可以搜索一篇《MCU 芯片加密历程》的文章。

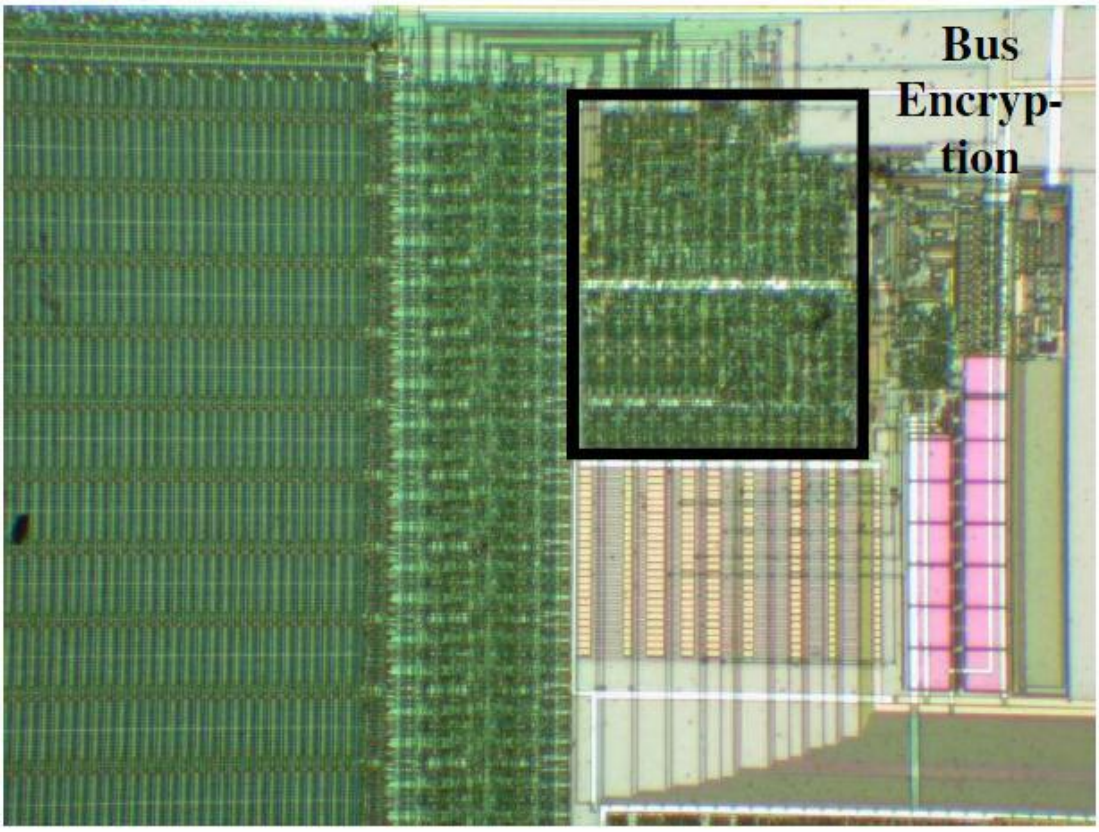

图 3：智能卡存储器总线加密技术，存储器中的数据都是密文存储，
即使被入侵者得到，也无法使用。

目前银行卡、电信 SIM 卡，社保卡等涉及到钱的安全领域，基本上都是使用智能卡芯片，并且各自都有其行业规范。就是因为智能卡芯片的安全程度最高，物理上被攻击的可能性极小。

如何判断加密芯片是否为智能卡内核的加密芯片？

由于智能卡的高安全性，不排除一些加密芯片供应商，会说自己的芯片使用的是智能卡内核。判断其真伪，很简单，只要问他们加密芯片是否能提供智能卡通信接口（即：ISO7816 接口），如果提供，则买一个智能卡读写器，操作一下即可。如果不提供 ISO7816 接口，则肯定不是智能卡内核的。另外还有个判断依据，问其加密芯片内部，是否提供硬件的 DES 算法寄存器，DES 算法硬件支持，是银行卡，社保卡等智能卡的标配。我所了解的智能卡内核的加密芯片有：中巨的 SMEC98SC、凌科芯安的 LKT4100F、ATMEL 的 AT88SC、Infineon 的 SLE 77CF1200S 等等。

四、 加密芯片的软件安全性

在加密芯片硬件不可破解的基础上，我们再谈谈加密芯片的软件方案。加密芯片物理硬件不可破解后，并不是就万事大吉了，软件设计的好坏，同样会影响到整套方案的安全。加密芯片的软件工作原理，我分为三个类型：1. 真值点判断类型； 2. 数据加解密类型; 3. 功能运算类型。下面列举一些简单的例子，解释其软件工作原理。

1. 真值点判断工作模式

主控 MCU 在工作时，判断一下外部的加密芯片是否合法，然后决定自身是否要正常工作，我把这一类加密芯片工作方式，统称为真值点判断类型。

这类判断加密芯片是否合法的方式有：PIN 码验证，对称算法运算(AES，DES 等)，非对称算法运算(RSA，ECC 等)，散列算法(HMAC-MD5， HMAC-SHA，HMAC-SM3 等)，挑战码方式(如 ATMEL 的 AT88SC 系列芯片)……

真值点判断方式，操作简单，对主控芯片原有代码改动较小。加密芯片提供者甚至可以提供简单配置一下密钥，就能工作起来。但这类方式有个致命弱点，如果侵入者将主控 MCU 的代码，反汇编，并作改动就可以绕过加密芯片，实现破解。如下面的反汇编代码中，将 C:0x0420 地址代码，改成直接挑砖指令：SJMP 0x042C，将完美绕过加密芯片。

所有这类型工作方式，都存在被破解的可能性，破解难度取决于找到对应真值点的位置，一旦找到，整套方案就被破解了。

```
void main()
{
  if(SafeCheck() == 0)        //开关判断
  {
    while(1);                 //不满足运行条件
  }
  else
  {
    WorkMode();
  }
}

unsigned char SafeCheck(void)//判断是否满足安全条件
{
  return 1;                   //可改为判断外部加密芯片是否正确
}

void WorkMode(void)           //进入MCU正常工作状态
{
  printf("Work OK!");
  while(1);                   //可改为正常工作代码
}
```

图 4：加密芯片工作在真值点判断模式下的原代码

```
                main:
C:0x0420    120440   LCALL    SafeCheck(C:0440)
C:0x0423    EF       MOV      A,R7
C:0x0424    7002     JNZ      C:0428
C:0x0426    80FE     SJMP     C:0426
C:0x0428    12042C   LCALL    WorkMode(C:042C)
C:0x042B    22       RET
              WorkMode:
C:0x042C    7BFF     MOV      R3,#0xFF
C:0x042E    7A04     MOV      R2,#0x04
C:0x0430    7937     MOV      R1,#0x37
C:0x0432    120065   LCALL    PRINTF(C:0065)
C:0x0435    80FE     SJMP     C:0435
C:0x0437    57       ANL      A,@R1
C:0x0438    6F       XRL      A,R7
C:0x0439    726B     ORL      C,0x2D.3
C:0x043B    204F4B   JB       0x29.7,C:0489
C:0x043E    2100     AJMP     C:0100
              SafeCheck:
C:0x0440    7F01     MOV      R7,#0x01
C:0x0442    22       RET
```

将此处代码直接改成：
SJMP C:042C
将直接跳过加密芯片，正常工作!

图 5：加密芯片工作在真值点判断模式下的反汇编

将 C:0x0420 地址代码，改成直接挑砖指令：SJMP 0x042C，将完美绕过加密芯片

2. 数据加解密工作模式

将一部分数据密文存放在加密芯片中，当主控 MCU 工作时，从加密芯片密文读出，然后在主控芯片中再解成明文使用，这一类统称为数据加解密类型。常见的加解密算法有对称算法(AES，DES 等)，非对称算法运算(RSA，ECC)等等。

这类工作模式，同样有漏洞，要求入侵者能够调试反汇编的代码。如图 7，在 C:0x01C0 处打断点，将变量 bDecryptData 中的明文数据得到，也将成功破解整套方案。

```
//数据解密函数，输入'A'，输出'B'。
void Decrypt(char *pCryptData, int iLen, char *pDecryptData)
{
  int i;
  for(i = 0; i < iLen; i++)
  {
    pCryptData[i] = pDecryptData[i] + 1;
  }
}


void main()
{
  char bCryptData[8] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};
  char bDecryptData[8] = {0};
  Decrypt(bCryptData, sizeof(bCryptData), bDecryptData);
  while(1);
}
```

图 6：加密芯片工作在数据加解密模式下的原代码

```
                Decrypt:
C:0x0148    8B18      MOV       0x18,R3
C:0x014A    8A19      MOV       0x19,R2
C:0x014C    891A      MOV       0x1A,R1
C:0x014E    8C1B      MOV       0x1B,R4
C:0x0150    8D1C      MOV       0x1C,R5
C:0x0152    E4        CLR       A
C:0x0153    FF        MOV       R7,A
C:0x0154    FE        MOV       R6,A
C:0x0155    C3        CLR       C
C:0x0156    EF        MOV       A,R7
C:0x0157    951C      SUBB      A,0x1C
C:0x0159    E51B      MOV       A,0x1B
C:0x015B    6480      XRL       A,#P0(0x80)
C:0x015D    F8        MOV       R0,A
C:0x015E    EE        MOV       A,R6
C:0x015F    6480      XRL       A,#P0(0x80)
C:0x0161    98        SUBB      A,R0
C:0x0162    5022      JNC       C:0186
C:0x0164    AB1D      MOV       R3,0x1D
C:0x0166    AA1E      MOV       R2,0x1E
C:0x0168    A91F      MOV       R1,0x1F
C:0x016A    8F82      MOV       DPL(0x82),R7
C:0x016C    8E83      MOV       DPH(0x83),R6
C:0x016E    1200F9    LCALL     C?CLDOPTR(C:00F9)
C:0x0171    04        INC       A
C:0x0172    AB18      MOV       R3,0x18
C:0x0174    AA19      MOV       R2,0x19
C:0x0176    A91A      MOV       R1,0x1A
C:0x0178    8F82      MOV       DPL(0x82),R7
C:0x017A    8E83      MOV       DPH(0x83),R6
C:0x017C    120126    LCALL     C?CSTOPTR(C:0126)
C:0x017F    0F        INC       R7
C:0x0180    BF0001    CJNE      R7,#0x00,C:0184
C:0x0183    0E        INC       R6
C:0x0184    80CF      SJMP      C:0155
C:0x0186    22        RET
```

```
                main:
C:0x0187    7808      MOV       R0,#0x08
C:0x0189    7C00      MOV       R4,#0x00
C:0x018B    7D00      MOV       R5,#0x00
C:0x018D    7BFF      MOV       R3,#0xFF
C:0x018F    7A01      MOV       R2,#0x01
C:0x0191    79C5      MOV       R1,#0xC5
C:0x0193    7E00      MOV       R6,#0x00
C:0x0195    7F08      MOV       R7,#0x08
C:0x0197    1200D3    LCALL     C?COPY(C:00D3)
C:0x019A    7810      MOV       R0,#0x10
C:0x019C    7C00      MOV       R4,#0x00
C:0x019E    7D00      MOV       R5,#0x00
C:0x01A0    7BFF      MOV       R3,#0xFF
C:0x01A2    7A01      MOV       R2,#0x01
C:0x01A4    79CD      MOV       R1,#0xCD
C:0x01A6    7E00      MOV       R6,#0x00
C:0x01A8    7F08      MOV       R7,#0x08
C:0x01AA    1200D3    LCALL     C?COPY(C:00D3)
C:0x01AD    751D00    MOV       0x1D,#0x00
C:0x01B0    751E00    MOV       0x1E,#0x00
C:0x01B3    751F10    MOV       0x1F,#0x10
C:0x01B6    7B00      MOV       R3,#0x00
C:0x01B8    7A00      MOV       R2,#0x00
C:0x01BA    7908      MOV       R1,#0x08
C:0x01BC    7D08      MOV       R5,#0x08
C:0x01BE    7C00      MOV       R4,#0x00
C:0x01C0    120148    LCALL     Decrypt(C:0148)
C:0x01C3    80FE      SJMP      C:01C3
                ix1000:
C:0x01C5    4142      AJMP      C:0242
C:0x01C7    434445    ORL       0x44,#0x45
C:0x01CA    46        ORL       A,@R0
C:0x01CB    47        ORL       A,@R1
C:0x01CC    48        ORL       A,R0
```

在此处打断点，可获取解密后数据!

图 7：加密芯片工作在数据加解密模式下的反汇编

将 C:0x01C0 地址打断点，并把对应地址变量内容读出，
然后替换汇编代码，也将成功破解

3. 功能运算工作模式

将主控芯片的一部关键代码放在加密芯片中，当主控 MCU 工作时，传入参数，请求加密芯片执行运算，并获取计算结果。这一类工作方式，统称为功能运算类型。

这类工作模式，是将一部分代码放入加密芯片中运行，例如图 8 中，计算圆周长的代码在加密芯片中，主控芯片只需要传给加密芯片圆的半径，就能够得到圆的周长。即使入侵者把主控 MCU 的反汇编代码，每一行都理解透，在不知道“圆周长”计算公式下，他们就算有通天的本事，也无法解密整套方案。

```c
//圆周长计算C = 2 * p * r, 放入加密芯片中
unsigned int CalPerimeter(unsigned int r)
{
  return (2 * 3.14 * r);
}


void main()
{
  int C;
  int r = 123;
  C = CalPerimeter(123);
  while(1);
}
```

图 8：加密芯片工作在功能运算模式下的原代码

```
                CalPerimeter:
C:0x01D6    AD07      MOV       R5,0x07
C:0x01D8    AC06      MOV       R4,0x06
C:0x01DA    E4        CLR       A
C:0x01DB    12010E    LCALL     C?FCASTI(C:010E)
C:0x01DE    7BC3      MOV       R3,#0xC3
C:0x01E0    7AF5      MOV       R2,#0xF5
C:0x01E2    79C8      MOV       R1,#0xC8
C:0x01E4    7840      MOV       R0,#0x40
C:0x01E6    120003    LCALL     C?FPMUL(C:0003)
C:0x01E9    120147    LCALL     C?CASTF(C:0147)
C:0x01EC    22        RET
                main:
C:0x01ED    750A00    MOV       0x0A,#0x00
C:0x01F0    750B7B    MOV       0x0B,#0x7B
C:0x01F3    7F7B      MOV       R7,#0x7B
C:0x01F5    7E00      MOV       R6,#0x00
C:0x01F7    1201D6    LCALL     CalPerimeter(C:01D6)
C:0x01FA    8E08      MOV       0x08,R6
C:0x01FC    8F09      MOV       0x09,R7
C:0x01FE    80FE      SJMP      C:01FE
```

图 9：加密芯片工作在功能运算模式下的反汇编

五、 结束语

一个好的加密方案，即少不了安全可靠的物理硬件的支持，也少不了灵活可变的软件支持。一些加密芯片厂商，一味的强调自己的硬件多么强大，支持算法种类多么的多，算法密钥长度多么的长，然而没有提供很好的软件开发支持，只能够配置密钥数据等，再强大的硬件，也可被轻松软件破解；另外一些加密芯片厂商，一味的强调自己的软件开发，多么灵活多变，速度多么的快，然而其硬件只是普通较偏门的 MCU，不是智能卡内核芯片，这样再完善的软件算法，也无法挡住入侵者直接破解其硬件，将整个加密芯片的二进制码全盘复制，并采购其一样的加密芯片，实现整个方案的破解。