基于无保护 AES 芯片的 CPA 攻击

王立敏[1],丁洁[2]

[1]中国科学院信息工程研究所 第五实验室 北京 中国 100093

[2] 中国科学院信息工程研究所 第五实验室 北京 中国 100093

摘要 高级加密标准（Advanced Encryption Standard，AES）是最常用的加密算法之一。 为了提升实际应用中加解密操作的速度，或者在小型芯片上完成加密工作，AES 通常被集成在加密芯片中。这使得其容易遭受侧信道攻击，尤其是能量分析攻击。本文中将采用相关能量分析 （Correlation Power Analysis，CPA）对 AES 的能量迹和字节替换环节之间的关系进行统计分析来猜测其对应的密钥。其结果表明对于普通的无保护 AES 芯片，CPA 攻击十分有效。

关键词 **侧信道攻击，AES，密码芯片，相关能量分析，能量迹**

The CPA Attack For Unprotected AES Chips

Wang Limin[1], Ding jie[2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Abstract Advanced Encryption Standard is one of the most commonly used encryption algorithms. In order to improve the speed of encrypted and decrypted operations or encrypt data on chips , the independent AES chip is used for encryption, which makes the chip vulnerable to Side-Channel Attack, especially to Power Analysis. This paper will guess the key of the AES by analyzing the correlation between power trace and SubBytes operation. This experiment shows that CPA attack can help crack secret keys more efficiently.

Key words Side-Channel Attack, AES, Cipher Chip, Correlation Power Analysis, Power Trace

1 引言

侧信道分析技术在硬件安全领域中应用十分广泛。其中较为常见的一种是能量分析，它主要包括三种分析方法，简单能量分析（Simple Power Analysis, SPA），差分能量分析(Differential Power Analysis, DPA)以及相关能量分析(Correlation Power Analysis, CPA)。 SPA 和 DPA 最初由 Paul Kocher, Joshua Jae, 和 Benjamin Jun 在 1999 年提出[1]。 SPA 利用的是密码芯片在进行不同的指令操作时所消耗的能量也不同这一特性，例如 AES 芯片在执行 10 轮操作时，它的 10 个能量消耗峰值将会十分明显。而 DPA 则更加复杂，它将测出来的能量迹分为几类，并计算它们均值的差，若假设的密钥是正确的，则会出现一个能量波峰，否则差值会在 0 附近波动。DPA 是十分有效的能量攻击方法，但是它依然存在诸如幽灵峰值的问题，而本试验中给出的能量迹又足够多，因此为了提高破译密钥的正确性，本文将采用 CPA 来对 AES 进行攻击。CPA 由 E.Brier 提出，是从 DPA 改进而来，它采用的是汉明重量模型[2]。

2 AES 介绍

AES 在安全性不低于三重数据加密算法（Triple Data Encryption Algorithm，TDEA，也叫 3DES）的同时，运算速度还比它快，因此被采用来替代原先的数据加密标准（Data Encryption Standard，DES），它具有很好的抗差分密码分析和线性密码分析的能

力。

AES 根据密钥的长度的不同有三种不同的版本分别为 AES-128，AES-192 以及 AES-256。本文只讨论利用 CPA 攻击无保护的 AES-128 算法，如算法 1 所示。

算法 1：无保护的 AES-128 算法

输入：明文 X[0-15],轮密钥 RoundKey[0-10]

输出：密文 X[0-15]

```
1.  for r = 0;r ≤ 8;r++ do
2.        X = X ⊕ RoundKey[r]; /*加轮密钥*/
3.        for i = 0; i≤15; i++ do
4.            Xi = SubBytes(Xi);     /*字节替换*/
5.        end
6.        X = ShiftRows(X);          /*行变换*/
7.        X = MixColumns(X);         /*列混淆*/
8.  end
9.  X = X⊕RoundKey[9];               /*最后一轮*/
10. for I = 0; i≤15; i++ do
11.       Xi = SubBytes(Xi);
12. end
13. X = ShiftRows（X）
14. X = X⊕RoundKey[10]               /*获取密文*/
```

AES-128 算法需要经过 10 轮的操作，除了最后一轮之外，前 9 轮都包括加轮密钥，字节替换，行变换，列混淆这四个步骤，而第十轮则少一个列混淆。这里的 11 个 128 位的轮密钥是原始的 128 位密钥经过密钥扩展得到的。

2.1 密钥扩展

密钥扩展是将最初始的 128 位密钥扩展成 11 个轮密钥，方便每一轮中的操作。假设有这样一个 128 位的密钥，如图 1 所示。然后按每 4 字节为一列排列成图 2 所示的阵列。

0x7770268a51bfa9b22f6f4069c395db5b

图 1 给定的 AES 密钥

w0	w1	w2	w3
0x77	0x51	0x2f	0xc3
0x70	0xbf	0x6f	0x95
0x26	0xa9	0x40	0xdb
0x8a	0xb2	0x69	0x5b

图 2 密钥的排布

图 2 中每一列分配一个标记 wi，第一列为 w0，第二列为 w1，以此类推。每一个轮密钥的第一列生成方式都较为复杂，这里以生成第 5 列（w4）为例。

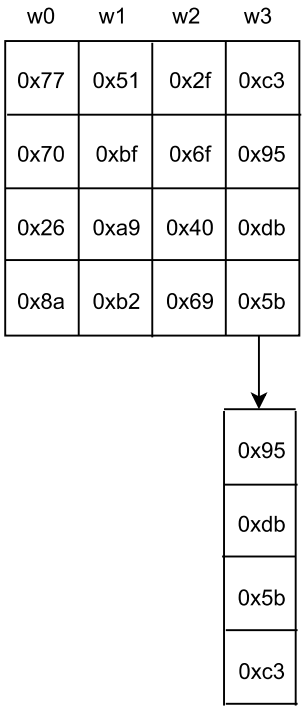

图 3 w3 左移一位

如图 3 所示，当将最初的密钥排列完毕以后，需要将最后一列进行循环左移。并且左移后的每个元素都要经过 S-BOX 的字节替换处理，图四表示了这一过程。关于 S-BOX 字节替换的详细描述可参看 2.3 部分。

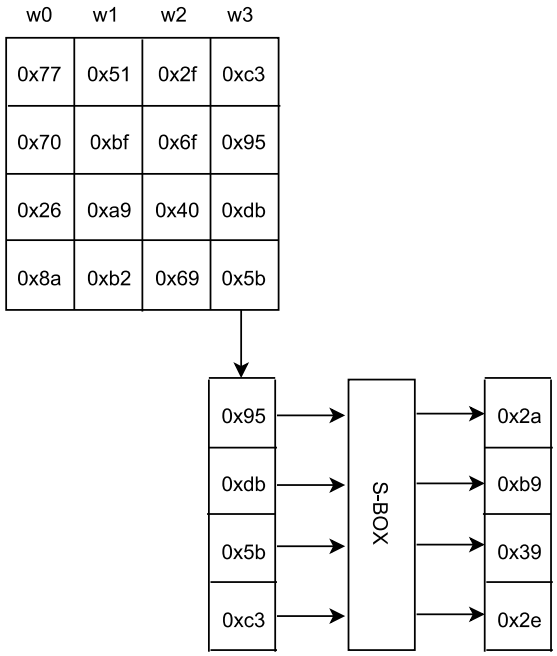

图 4 左移后经过 S-BOX 进行字节替换

假设正在生成第 i 列密钥，这里是第 5 列，即为 w4。由于此处是第二个轮密钥的第一列，则经过字节替换后的结果，需要与 wi-4 即 w0 以及 rcon[i/4-1]即 rcon[0]进行异或才能最终得出 w4。其中 rcon 是如图 6 所示的 10 列数字。

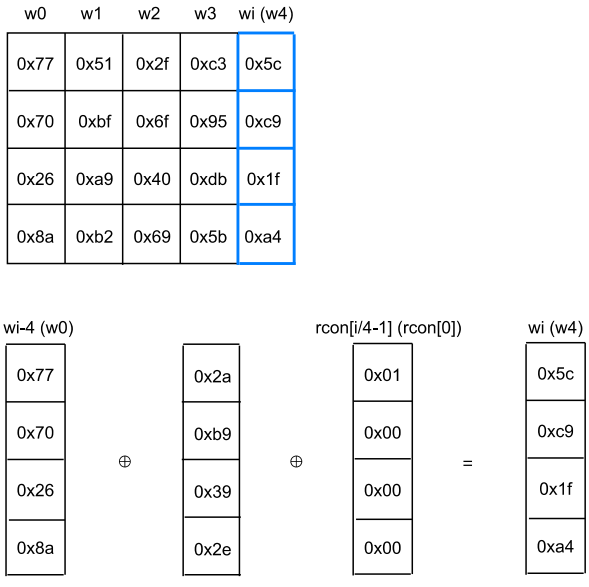

图 5 轮密钥第一列生成

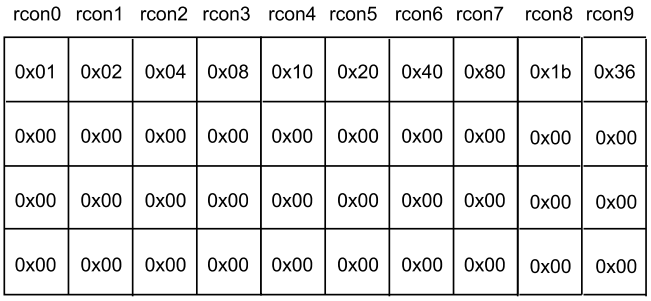

图 6 RCON 数组

只有轮密钥的第一列需要按照上述方法生成，接下来的三列则通过图 7 所示的方法，通过将 wi-1 以及 wi-4 异或来求得，例如这里轮密钥第二列 w5，则是通过 w4 和 w1 异或得来的。

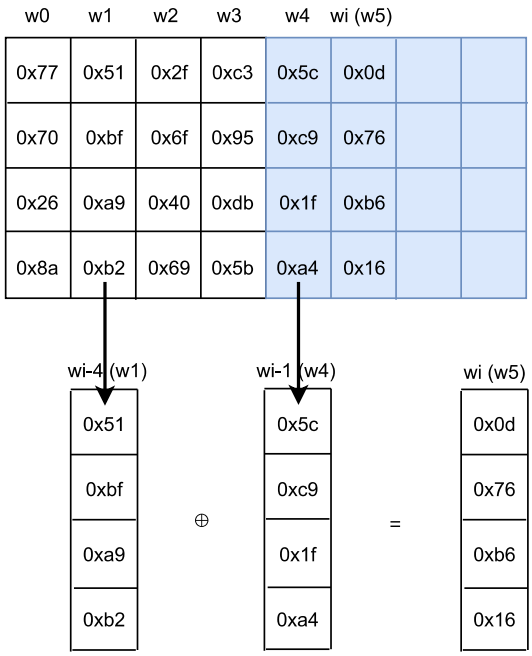

图 7 轮密钥第二列生成

之后的每一个轮密钥都将按照上述方法进行计算得出，轮密钥第一列 wi=SubByte（LeftShift（wi-1））⊕wi-4⊕rcon[i/4-1],而第二，三，四列的计算规则为 wi= wi-1⊕wi-4。图 8 即为新生成的轮密钥，可以很明显看出轮密钥 0 就是最开始输入的密钥，通常对 AES 的能量分析，都只分析第一轮，这一轮的轮密钥加操作使用的是轮密钥 0，即原始输入的密钥。因此在进行 CPA 攻击时不需要经过密钥扩展这一步。

w0	w1	w2	w3	w4	w5	w6	w7	.....			
0x77	0x51	0x2f	0xc3	0x5c	0x0d	0x22	0xe1				
0x70	0xbf	0x6f	0x95	0xc9	0x76	0x19	0x8c				
0x26	0xa9	0x40	0xdb	0x1f	0xb6	0xf6	0x20				
0x8a	0xb2	0x69	0x5b	0xa4	0x16	0x7f	0x24				

轮密钥0　　轮密钥1　　.....

图 8 新生成的轮密钥

2.2 加轮密钥

AES 经过密钥扩展得到轮密钥之后，将进行 10 轮操作，在每一轮操作中都需要加轮密钥。 首先需要将明文和轮密钥按 4 字节为一列排成 4 列，分别命名为 P0-P15 和 C0-C15,然后将明文和密文，按字节进行异或，即 P0⊕C0-P15⊕C15。得到的最终结果可用于后续的字节替换。

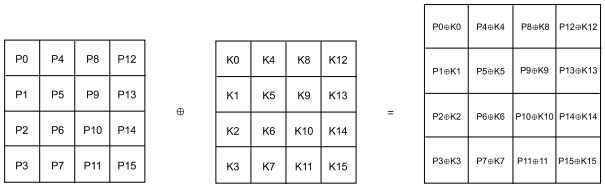

图 9 轮密钥加

2.3 字节替换

字节替换一般使用的是 Rijndael S-box[3],它将需要替换的字节的高四位作为横坐标，低四位作为纵坐标从 S-BOX 中选中对应的字节作为替代值。

如图 10，待替代字节为 0x01,则横坐标为高四位 0x0,纵坐标为低四位 0x1，从 S-BOX 中找到对应坐标的值 0x7c，然后将其替换，图 10 展示 S-BOX 的替换过程，其中的 S-BOX 只是完整表格的一部分。

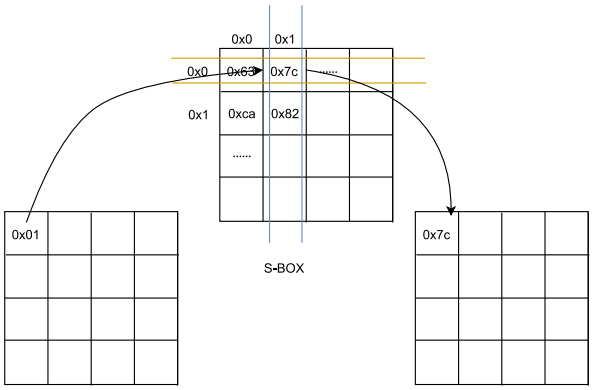

图 10 S-BOX 替换

3 CPA 攻击

CPA 攻击主要采用汉明重量模型。计算汉明重量与对应能量迹之间的相关系数，若相关系数越大，则说明他们之间的相关性越强，若数据中某一猜测密钥对应的相关系数相比于其他相关系数要大的多，就可以说明这一猜测密钥是正确的。

3.1 汉明重量

汉明重量可以表示一个二进制字符串中 1 的个数。已经有论文通过实验证明了输出结果的汉明重量与能量消耗之间有明确的关系，能量消耗随着汉明重量的增大而增大，而且有较为明确的界限，经过计算，它们的相关系数甚至能够达到 0.9919[4]。这也说明 CPA 使用汉明重量模型是合理的。

3.2 CPA 攻击方法

由于字节替换占用了总能量消耗的大部分[5]，因此在 CPA 攻击时可以只考虑 S-BOX 的输出与能量迹之间的关系。

本文将密钥分成 16 个子密钥分别破解，每一子密钥为一个字节。首先考虑第一个子密钥 GuessKey 的破解，这里需要将其从 0 遍历到 255，然后每遍历一个值，就参照 AES 的一轮加密过程，将其与明文 pt 进行异或，异或之后再经过 S-BOX 的字节替换即可得到用于求解汉明重量的输入字符串 input，如算法 2 第 6 行所示。值得注意的是 AES 加密需要对原始密钥进行密钥扩展，将一个原始密钥扩展成 11 个轮密钥，但是如算法 2 第 4 行所示，这里轮秘钥并不是密钥扩展得到的，而是直接对原始密钥进行了异或。这是因为，从本文 2.1 部分密钥扩展的过程可以看出，第一个轮密钥就是原始密钥本身，而且此处我们也只需要考虑 AES 的第一轮能量消耗和汉明重量之间的关系即可，因此在这里进行 AES 第一轮操作时可以不用进行密钥扩展而直接使用原始密钥。而后将输入字符串 input 的汉明重量求出，存在数组中，并求出他们与能量迹的 Pearson 相关系数。于是便可以得到横坐标为 256 个 GuessKey，纵坐标为相关系数的统计图（请参看附件文件夹中的图），若遍历到的密钥不正确，则相关系数的波动幅度并不大，而当猜测的密钥正确时，对应的相关系数将会是一个明显的波峰。

算法 2：CPA 攻击算法

输入：能量迹 traces，明文 pt

输出：密钥 keys

```
1.  for i = 0; i<16 ;i++ do
2.     for GuessKey=0;GuessKey<256;GuessKey++ do
3.          for j = 0;j<TraceCnt;j++ do
4.              input=pt[j][i]⊕GuessKey;
5.              input=SubBytes(input);
6.              hws[j]=HammingWeight(input);
7.          end
8.
9.          for j = 0;j<PointsCnt;j++ do
10.           pccs[j]=PCC(Trans(traces)[j],hws);
11.             /*PCC 为求 Pearson 相关系数*/
12.             /*Trans 为矩阵转置*/
13.         end
14.
15.         CPA[GuessKey]=Max(pccs)
16.    end
17.
18.    keys[i]=Argmax(CPA)
19. end
```

4 结论

本次实验所得到的能量迹数据为第 36 组，攻击源代码可以参看 source 目录，先通过 ReadFile.py 读取实验数据,并将其转为攻击代码需要的格式。再执行 CPA.py 进行攻击，执行过程中通过 matplolt 画图，可以很清晰地看出相关系数峰值，这些图可以在附件文件夹中看到。最终测算出来的密钥为 0x77 70 26 8a 51 bf a9 b2 2f 6f 40 69 c3 95 db 5b。使用 source 目录下的 AES 代码[6]加密明文，并将得到的密文与给定的密文进行对比，可确定其为正确的密钥。因此可以认为本攻击手段采用 CPA 攻击是合理的。

参考文献

[1] KOCHER P, JAFFE J, JUN B. Differential power analysis[C]//Annual International Cryptology Conference. Springer, 1999: 388–397.

[2] BRIER E, CLAVIER C, OLIVIER F. Correlation Power Analysis with a Leakage Model[G]//JOYE M, QUISQUATER J-J. Cryptographic Hardware and Embedded Systems - CHES 2004. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 3156: 16–29.

[3] DAEMEN J. The Rijndael Block Cipher[J]. : 45.

[4] LO O, BUCHANAN W J, CARSON D. Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)[J]. Journal of Cyber Security Technology, 2017, 1(2): 88–107.

[5] MORIOKA S, SATOH A. An Optimized S-Box Circuit Architecture for Low Power AES Design[G]//KALISKI B S, KOÇ çetin K, PAAR C. Cryptographic Hardware and Embedded Systems - CHES 2002. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, 2523: 172–186.

[6] Skycker/AES: Rijndael cipher algorithm[EB/OL]. [2018-07-31]. https://github.com/Skycker/AES.