AES 高阶掩码方案抗功耗攻击

何利文,国海轮,安　聪
(南京邮电大学,江苏 南京 210003)

摘　要:随着密码技术和信息技术的发展,目前的密码算法本身已足够强大,能够对抗传统的密码分析手段,但由于设备本身的工艺特性,其运行时会泄露如功耗、电磁、时间等信息,这些信息能够被攻击者利用从而破解密钥,该方法称为侧信道攻击。AES 加密算法容易受到侧信道攻击,为了解决该问题,通常通过添加一个或几个随机值即掩码。当设计 d 阶时每个值都用到掩码,执行 SCA 的复杂度呈指数增长,因此设计 d 阶为安全参数的掩码方案对密码实现的物理安全性有重要意义。根据 AES 常用的功耗攻击技术,提出了一种 d 阶掩码方案,此方案是基于 Ishai 等人在 Crypto 发表的面向硬件的掩码方案。与此方案相比,所设计的方案可以在处理器上有效的实现。实验结果表明,该方案降低了理论功耗和实践功耗之间的相关性,很好地保护了中间值不被泄露,提高了 AES 加密算法的抗功耗攻击能力。

关键词:AES 加密;侧信道攻击;CPA;高阶掩码;功耗攻击
中图分类号:TP31　　**文献标识码**:A　　**文章编号**:1673-629X(2022)05-0068-07
doi:10.3969/j.issn.1673-629X.2022.05.012

Anti Power Attack of AES High-order Mask Scheme

HE Li-wen,GUO Hai-lun,AN Cong,
(Nanjing University of Posts and Telecommunications,Nanjing 210003,China)

Abstract:Along with the development of the password technology and information technology,the current password algorithm itself is strong enough to against the traditional means of password analysis,but due to the technological characteristics of the device itself,its runtime leaked information,such as power consumption,electromagnetic,time,which can be used to break key attackers. This method is called the side channel attacks. AES algorithm is vulnerable to side channel attack. In order to solve this problem,one or more random values are added,namely masks. Masks are used for each value when d-order is designed,and the complexity of SCA execution increases exponentially. Therefore,it is of great significance to design a mask scheme with d-order as a security parameter for the physical security of cryptographic implementation. We propose a d-order mask scheme based on the hardware-oriented mask scheme published by Ishai et al. in Crypto,based on the commonly used power attack technology of AES. Compared with this scheme,the proposed scheme can be effectively implemented on the processor. The experimental results show that the proposed scheme reduces the correlation between theoretical power consumption and practical power consumption,protects the intermediate value from leakage,and improves the anti-power attack ability of AES encryption algorithm.

Key words:AES encryption;side channel attack;CPA;high-order mask;power consumption attack

0　引　言

随着信息化的高速发展,智能信息化在深刻影响着人们生活各方面的同时,也带来了信息安全问题,故受到了极大的重视。目前的密码算法已经可以对抗传统的数学分析,如代数分析、差分分析。但最近兴起的侧信道分析[1](side channel analysis,SCA),可以利用密码芯片在加密过程中泄露一些功耗、辐射等信息,这些信息可以帮助攻击者破解密钥。侧信道攻击中利用功耗泄露信息来破解密钥的方法称为功耗攻击(power analysis attack)。功耗分析包括简单功耗分析(simple power analysis, SPA)、差分功耗分析[2](different power analysis, DPA)、相关功耗分析(connection power analysis,CPA)等。

由于侧信道攻击的出现,也产生了许多对抗方法,例如加入掩码、使用数据与功耗不相关的逻辑单元、增加噪声[3]产生电路。该文设计了一种 $d+1$ 高阶掩码

收稿日期:2021-04-15　　**修回日期**:2021-08-18
基金项目:2018 年国家重点研发计划项目(2018YFB2100200)
作者简介:何利文(1968-),男,博士,教授,研究方向为网络、信息安全、云计算大数据分析与应用;通信作者:安　聪(1996-),男,硕士研究生,研究方向为信息安全。

方案,通过将敏感数据拆分成多份来消除明文与功耗之间的相关性。

1　AES 加密算法

AES[4]算法是一种对称分组密码算法。AES 将分组长度设为 128 比特、192 比特、256 比特三种密钥长度。后面以 128 位的密钥长度为例介绍。加密过程如图 1 所示。

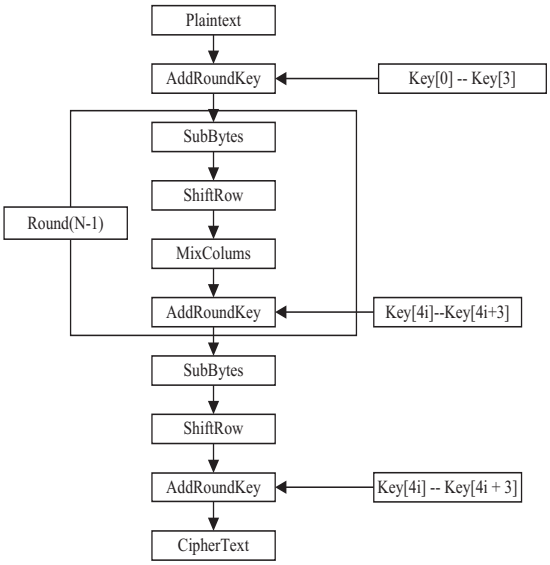

图 1　AES 加密过程

2　功耗攻击

目前应用最广泛的三种功耗攻击[5]手段为 SPA(简单功耗分析攻击)、CPA(相关功耗分析攻击)、DPA[6](差分功耗分析攻击)。SPA 比后面两种方法简单,对一些简单的加密算法是有效的,但对于一些复杂的算法比如 AES 算法、DES 算法,就不容易成功。CPA 和 DPA 的攻击效果比 SPA[7]好,目前功耗分析[7]方法的研究主要是后面两种。

2.1　CPA 攻击

相关能量分析(correlation power analysis,CPA)的攻击原理在于电子设备在进行加密操作的时候,它消耗的瞬时能量[8]和数据以及所进行的操作是相关的。攻击者需要从芯片设备上测量出大量的实时功耗数据,再通过建立功耗仿真模型计算出一组特定中间数据的预期功耗[9]。其理论基础是皮尔森相关系数[10]。

2.2　DPA 攻击

DPA 通过采集大量明文功耗与数学统计相结合,比较假设的中间值与实际功耗值之间的相关性来猜解出密钥[11]。

第一步:输入随机明文并记录功耗轨迹。将 n 组随机明文 $c=[c_0,c_1,\cdots,c_n]$(n 为一个较大的值,输入明文相互独立且随机逐一输入到加密芯片中进行加密运算。

第二步:计算区分函数 $D(m_i,k_j)$。对于一个 8 比特的密钥数据,将猜测密钥 K-guess 进行 0 到 255 之间的遍历猜测。

第三步:功耗波形子集划分。如第二步所示,对于猜测密钥 k_j 其对输入明文 m_i 进行区分函数的运算,且每个明文 m_i 对应着一组功耗波形数据。

第四步:计算差分功耗波形。步骤 3 中已经划分出两个功耗波形子集,对两个集合取平均和进行差分操作运算。

第五步:比较差分功耗波形峰值。在对 256 个猜测密钥猜测完毕后得出 256 组差分功耗波形数据。取每一组差分功耗波形的峰值进行比较,其中最大峰值对应的猜测密钥最有可能是真实的隐藏密钥。

3　高阶掩码方案

通过对敏感数据拆分为多份来消除明文与功耗的相关性。以汉明重量为随机种子生成随机掩码,用于高阶掩码刷新算法;重新设计了 S 盒,它由求逆运算和仿射变换组成,该方案对这两部分进行重新设计,新 S 盒采用按列的变换方式进行,由原来的查表操作变为平方乘运算。在加密过程中保证中间状态的敏感数据被随机拆分为 $d+1$ 份,其中任意一份与原来的状态值独立。

具体过程为:将明文信息拆分为 $d+1$ 份(d 为正整数),并根据随机掩码算法生成随机值掩码。首先将拆分的 $d+1$ 份明文和掩码的异或结果进行加密。前九轮依次执行字节替换、行移位、列混淆、轮密钥加操作,最后一轮进行密钥列混淆操作,然后将 $d+1$ 份结果异或得到最终的密文。

3.1　基本原理

加密算法高阶掩码实现时,在计算过程中每个敏感变量[12] x 被随机分割为 $d+1$ 份共享 $x_0,x_1,\cdots,x_d$。满足 $x_0\oplus x_1\oplus\cdots\oplus x_d=x$。每个掩码变量涉及到 d 个掩码时称为 d 阶掩码。方案满足下面 2 个性质:

(1)完整性:d 个异或的结果必须产生预期的密文。

(2) d 阶安全性:任何由 d 或者少于 d 的中间变量组成的元组必须独立于任何敏感变量。

假设掩码是均匀分布的,加入的掩码使得计算的每个中间变量统计上独立于任何敏感变量。所以利用中间变量相关的信息泄露的侧信道攻击不再那么顺利。d 阶掩码理论上容易受到 $d+1$ 阶 SCA 攻击。SCA 利用 $d+1$ 中间变量相关的信息泄露,信息的泄露取决于敏感变量。然而随着 d 的增加这种攻击变得不切实际,使得高阶掩码成为一种合理的解决方法。

3.2 安全乘法

安全乘法[13]是本方案的关键步骤,主要是在 S-box 阶段运用。在设计 S 盒时,字节替换的求逆和仿射变换必须重新设计,所以必须对之前的非线性操作做安全乘法。

安全乘法 :

Input:shares $a_i \oplus_i a_i = a$,shares $b_i \oplus_i b_i = b$

Output:shares $c_i \oplus_i c_i = ab$

1. for $i = 0$ to d do
2. for $j = i + 1$ to d do
3. $r_{i,j} \leftarrow \text{rand}(n)$
4. $r_{j,i} \leftarrow (r_{i,j} \oplus a_i b_j) \oplus a_j b_i$
5. for $i = 0$ to d do
6. $c_i \leftarrow a_i b_j$
7. for $j = 0$ to d do $c_i \leftarrow c_i \oplus r_{i,j}$

Ishai 等人[14]提出了一种高阶掩码 ISW 方案。用来保护任意 d 阶的硬件电路。假设要保护的电路都是由非和与门组成的,用掩码值将电路转换成一个新的保护电路。主要的问题是如何保护与门的安全,为此他们提出了下面的方案。

a 和 b 被分为 $d+1$ 份,即 $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$, $c = ab$, $\bigoplus_i c_i = c$

(1) $0 \leqslant i \leqslant j \leqslant d$,选一个随机数 $r_{i,j}$ 。

(2)$0 \leqslant i \leqslant j \leqslant d$,计算:

$$r_{j,i} = (r_{i,j} \oplus a_i b_j) \oplus a_j b_i$$

(3) $0 \leqslant i \leqslant j \leqslant d$,计算:

$$c_i = a_i b_j \oplus \bigoplus_{j \neq i} r_{i,j}$$

$$\bigoplus_i c_i = \bigoplus_i (a_i b_j \bigoplus_{j \neq i} r_{i,j}) =$$

$$\bigoplus_i (a_i b_j \bigoplus_{j>i} r_{i,j} \oplus \bigoplus_{j<i} (r_{j,i} \oplus a_i b_j \oplus a_j b_i)) =$$

$$\bigoplus_i (a_i b_j \bigoplus_{j<i} (a_i b_j \oplus a_j b_i) =$$

$$(\bigoplus_i a_i)(\bigoplus_i b_i)$$

虽然 ISW 方案理论上很安全,但它的实现开销很大。每个与门通过 $(d+1)2$ 和 $2d(d+1)$ 异或门进行编码。它要求在每个时钟周期产生 $d(d+1)2$ 个随机比特数据。对于 AES 算法来说,对 S 盒做屏蔽会增加电路的门数。随着 d 的增加门电路数急剧增加。

3.3 高阶 AES 掩码

AES 加密包括轮密钥加、字节替换、行移位、列混淆操作。其中只有 S-box[15]是非线性的,也是掩码方案的主要困难。

该 $d+1$ 阶掩码方案通过加入掩码[16]来增加抗功耗攻击操作。还是以异或方式为基础设计了 AES 算法中的安全乘法、安全 S 盒、安全行移位与安全列混淆。

Masking SubBytes:

方案主要包括对任意 d 阶幂函数做屏蔽。和 Blomer 等人[17]的方案相比,该文的求幂对 d 阶的掩码是安全的。

根据前面的知识,需要设计安全的求逆即 $(c_0, c_1, \cdots, c_d) \leftarrow \text{SecMult}((a_0, a_1, \cdots, a_d), (b_0, b_1, \cdots, b_d))$。

Blomer 等人[18]已经描述过 $d = 1$ 阶的,核心思想是对一阶掩码输入做求幂比如平方乘算法,同时保证进行掩码校正。Rivain 和 Prouf[19]提出了一种对 d 阶掩码输入进行运算的求幂算法,并保证 d 阶 SCA 安全。虽然这种方法的乘法只达到了 4 次,但是需要 7 次额外的平方操作。在求逆的过程中将其减少到 2,并且不需要平方。Rivain and Prouff 改进了之前的求逆操作[13],将乘法次数从 13 减少到 4 次。

$$z \leftarrow x^2$$

$$y \leftarrow z * x$$

$$w \leftarrow y^4$$

$$y \leftarrow yw$$

$$y \leftarrow y^{16}$$

$$y \leftarrow yw$$

$$y \leftarrow yz$$

在掩码时, x 、y 、z 、w 被分成 $d+1$ 份最终得到 $(\bigoplus_i^d y_i) \leftarrow (\bigoplus_i^d x_i)$ 。

安全求逆算法 :

Input:shares $x_i \oplus_i x_i = x$

Output:shares $y_i \oplus_i y_i = x^{-1}$

1. for $i = 0$ to d do $z_i \leftarrow x_i^2$
2. RefreshMasks$(z_0, \cdots, z_d)$
3. $(y_0, \cdots, y_d) \leftarrow \text{SecMult}((z_0, \cdots, z_d), (x_0, \cdots, x_d))$
4. for $i = 0$ to d do $w_i \leftarrow y_i^4$
5. RefreshMasks$(w_0, \cdots, w_d)$
6. $(y_0, \cdots, y_d) \leftarrow \text{SecMult}((y_0, \cdots, y_d), (w_0, \cdots, w_d))$
7. for $i = 0$ to d do $y_i \leftarrow y_i^{16}$
8. $(y_0, \cdots, y_d) \leftarrow \text{SecMult}((y_0, \cdots, y_d), (w_0, \cdots, w_d))$
9. $(y_0, \cdots, y_d) \leftarrow \text{SecMult}((y_0, \cdots, y_d), (z_0, \cdots, z_d))$

随机掩码算法 rand( ) :

首先确定随机掩码个数为 16 个,需要生成 16 个[0,8]的汉明重量 $\text{HW}_i = \{1,3,6,0,2,4,5,7,8,1,4,5,2,8,6,3\}$ 的不同取值确定相应的随机掩码字节,比特 1 的位置随机放置。

RefreshMasks 掩码刷新算法 :

Input:shares $x_i \oplus_i x_i = x$

Output:shares $x_i \oplus_i x_i = x$

1. for $i = 0$ to d do
2. tmp $\leftarrow$ rand( )
3. $x_0 \leftarrow x_0 \oplus$ tmp
4. $x_i \leftarrow x_i \oplus$ tmp

对仿射变换[20]做掩码操作很容易,其中需要注意如果 d 是偶数要异或 0x63。

安全 S 盒:

Input:shares $x_i \oplus_i x_i = x$

Output:shares $y_i \oplus_i y_i = S(x)$

1. $(y_0,\cdots,y_d) \leftarrow \text{SecExp254}((x_0,\cdots,x_d))$
2. for $i = 0$ to d do $y_i \leftarrow \text{Afine}(y_i)$
3. if($d \bmod 2 == 1$) then $y_0 \leftarrow y_0 \oplus 0x63$

AES 密钥扩展产生一个 $4*4(N_r+1)$ 字节数组 w。称为密钥表,其中 N_r 是轮数,$w_{*,j}$ 表示 w 的第 j 列。每一组 4 列 $(w_{*,4r-3},w_{*,4r-2},w_{*,4r-1},w_{*,4r})$ 组成一圆键 K_r,并在第 r 个轮密钥加阶段异或到状态矩阵。Subword 接受一个四字节输入并将 AES S-box 应用于每个字节,让 Rotword 以一个四字节作为输入,从上往下循环移动一个字节。轮常量异或 Rcon。密钥列表 w 的第 j 列被定义为 $w_{*,j} = w_{*,j-Nk} \oplus t$。

$T =$

$\text{RotWord}(\text{SubWord}(w_{*,j-1})) \oplus \text{Rcon}_{j/N_k} \text{if}(j \bmod N_k = 0)$

$\text{SubWord}(w_{*,j-1}) \text{if}(N_k = 8) \text{and}(j \bmod N_k = 4)$

$w_{*,j-1} \text{otherwise}$

为了安全地进行 d 阶密钥扩展[21],将 w 分为 $d+1$ 个 $w_0,w_1,\cdots,w_d$。每个共享的第一列都填满了刚开始时的密钥共享。每次新的时间表列 $w_{*,j}$ 重新计算,它的 $d+1$ 份计算如下:

$$(w_i)_{*,j} = (w_i)_{*,j-N_k} \oplus t_i$$

t_i 表示 w 的四个字节的共享安全计算得到 t 的 4 个字节的共享 $w_{*,j-1}$,从上面的描述可以很容易推导出这样的安全计算。Subword 通过之前描述的安全 s-box 计算应用于字节共享 $(w_0)_{l,j}\cdots(w_d)_{l,j}$ 为每个 row-coordinatel $\in [1,4]$。因为 Rotword 相对于异或是线性的,所以它会分别应用于每一份分享。最后 Rcon_{j/N_k} 必须添加到 t,它被添加到它的每一份如 t_0。

密钥扩展算法:

Input:shares $k_i \oplus_i k_i = k$

Output:shares $w_i \oplus_i w_i = w$

1. for $j = 1$ to N_k do
2. for $i = 0$ to d do $(w_i)_{*,j} \leftarrow (k_i)_{*,j}$
3. for $j = N_k + 1$ to $4*(N_k+1)$ do
4. for $i = 0$ to d do $t_i \leftarrow (w_i)_{*,j-1}$
5. if($j \bmod N_k = 0$) or ($N_k = 8$) and ($j \bmod N_k = 4$) then
6. for $l = 1$ to 4 do $((t_0)_l\cdots(t_d)_l) \leftarrow \text{SecSbox}((t_0)_l\cdots(t_d)_l)$
7. if($j \bmod N_k = 0$) then
8. for $i = 0$ to d do $t_i \leftarrow \text{RotWord}(t_i)$
9. $t_0 \leftarrow t_0 \oplus \text{Rcon}_{j/N_k}$
10. for $i = 0$ to d do $(w_i)_{*,j-1} \leftarrow (w_i)_{*,j-N_k} \oplus t_i$

3.4 高阶掩码 AES 算法流程

AES 操作一个 $4*4$ 的状态矩阵,由明文初始化,加密结束时得到密文。AES 加密 1 到 9 轮,每轮由 4 个阶段组成:轮密钥加、字节替换、行移位、列混淆。最后一轮没有列混淆。AES 加密有 10、12 和 14 轮,轮数取决于密钥长度。

加密开始前密钥已经被掩码屏蔽,并且把其分为 $d+1$ 份作为加密输入。开始的时候状态矩阵(16 个明文)被分为 $d+1$ 个状态 $s_0,s_1,\cdots,s_d$。满足 $s = s_0 \oplus s_1 \oplus \cdots \oplus s_d$。在加密结束时 $d+1$ 份异或得到密文。

Masking AddRoundkey:每轮的轮密钥加操作是每个状态和对应轮的密钥异或得出的,如下:$s \oplus k_r = (s_0 \oplus k_0) \oplus \cdots \oplus (s_d \oplus k_d)$

Masking ShiftRows and MixColumns:

行移位和列混淆都属于线性变换[22],对于行移位,状态的最后三行中每行分别循环移动 1 个字节、2 个字节、3 个字节;对于列混淆,将状态的每列看作有限域上的一个多项式,通过系数合并得到 $c(x) = 3x^2 + 2x^2 + x + 2$,然后在 $\text{modulo}(x^4+1)$ 下将它与一个固定多项式进行乘法操作。

因为这两个算法都是线性的,所以对它们加入掩码相对来说比较容易,只要将它们分别对应于每个状态共享[23]即可。

$\text{ShiftRows}(s) \oplus \text{ShiftRows}(s_i)$

$\text{MixColums}(s) \oplus \text{MixColums}(s_i)$

完整 AES 加密算法过程:

Input:p, key shares $k_i \oplus_i k_i = k$

Output:ciphertext c

1. $s_0 \leftarrow p$
2. for $i = 0$ *to* d do
3. $s_i \leftarrow \text{rand}(16*8)$
4. $s_0 \leftarrow s_0 \oplus s_i$

1 到 9 轮

5. for $r = 0$ to $N_r - 1$ do
6. for $i = 0$ to d do $s_i \oplus k_i^r$
7. for $l = 1$ to 4, $j = 1$ to 4 do
8. $((s_0)_{l,j}\cdots(s_d)_{l,j}) \leftarrow \text{SecSbox}((s_0)_{l,j}\cdots(s_d)_{l,j})$
9. for $i = 0$ to d do $s_i \leftarrow \text{MixColumns}(\text{ShiftRows}(s_i))$

最后一轮

10. for $i = 0$ to d do $s_i \leftarrow s_i \oplus k_i^{N_r}$
11. for $l = 1$ to 4, $j = 1$ to 4 do
12. $((s_0)_{l,j}\cdots(s_d)_{l,j}) \leftarrow \text{SecSbox}((s_0)_{l,j}\cdots(s_d)_{l,j})$
13. for $i = 0$ *to* d do $s_i \leftarrow \text{ShiftRows}(s_i)$
14. for $i = 0$ *to* d do $s_i \leftarrow s_i \oplus k_i^{N_r+1}$
15. $c \leftarrow s_0$
16. for $i = 1$ to d do $c \leftarrow c \oplus s_i$

4 实验结果

验证设计方案的正确性,中间 16 个字节被分为四份,四份的异或结果为密文。由于篇幅限制该文只给出最后一轮结果,见表 1。由最后的密文可以看出,该方案设计是正确的。

表 1 异或结果

P	32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34	异或结果
	C86F D3 48 29 A5 A1 04 C2 CF AA 45 DO 8D D1 23 DC DD 43 2E 40 8F 77 3C 11 36 F3 96 EO 54 BD 20 6ABD DC 07 7A 3E B6 2D 15 6C 48 07 7C BD 37 CO 47 OD 90 78 36 C8 71 7F 42 9C 94 DF 51 9F CC F1	39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A OB 32(密文)

ChipWhisperer[19] 是一个完全开放的嵌入式硬件安全研究平台,可用于基于硬件加密系统或软件加密系统的侧信道攻击实验平台。它里面包含了一系列可用于侧信道分析的软硬件工具集,如目标加密设备、低功耗采集设备、功耗分析软件等。用到的设备为 ChipWhisperer-Lite Board 和 XMEGA[24] Target Board (CW303)。上述方案用 C 语言实现,然后对 XMEGA 设备进行编程,将写好的代码传到目标设备。实验对没有添加掩码和添加掩码的 AES 算法进行功耗攻击,攻击点选取在 S 盒之后。表 2 是攻击结果,对 16 字节密钥的攻击。

表 2 无掩码 AES 算法 CPA 攻击结果

	0	1	2	3	4	5	6	7
P	0	0	0	0	0	0	0	0
0	2B 0.836 2	7E 0.894 3	15 0.706 2	16 0.763 8	28 0.545 9	AE 0.914 5	D2 0.754 9	A6 0.885 7
1	F4 0.414 9	8B 0.420 8	98 0.419 8	2D 0.459 6	E7 0.448 7	AF 0.445 7	25 0.538 5	CB 0.484 5
2	07 0.403 4	07 0.415 3	29 0.415 4	88 0.395 2	2B 0.419 9	44 0.432 6	06 0.487 0	B7 0.447 4
3	7B 0.393 9	80 0.395 4	28 0.412 8	3F 0.392 3	7C 0.398 7	F9 0.426 0	4F 0.465 8	C4 0.441 8
	8	9	10	11	12	13	14	15
P	0	0	0	0	0	0	0	0
0	A8 0.809 1	F7 0.803 9	15 0.661 4	88 0.788 4	09 0.485 0	CF 0.845 2	4F 0.673 0	3C 0.856 1
1	AA 0.557 9	97 0.460 8	OE 0.444 5	24 0.444 5	AA 0.459 4	FB 0.448 5	D8 0.467 6	BA 0.448 9
2	7D 0.454 2	B0 0.436 7	43 0.436 9	7C 0.442 0	F3 0.430 8	68 0.443 0	22 0.456 2	DE 0.416 6
3	A8 0.423 0	1E 0.436 4	46 0.433 0	99 0.437 7	DD 0.419 5	E9 0.421 2	DE 0.431 7	CF 0.414 9

表 2 中的结果是由 ChipWhisperer 平台的 Analyzer 工具分析实现的,图中第一排即为破解出来的密钥,在分析时,每一个字节所有猜测密钥都进行计算,采用的是相关系数公式计算假设功耗和真实功耗的相关程度,相关度越高密钥排名越靠前。由于没有加入掩码进行防护中间值很容易泄露,密钥轻松地就被破解。

由表 3 可以看出,最后的密钥都没有被成功破解,原因在于 S 盒替换不同于之前的查表操作,而是采用按列的变换进行的,加密过程中所有敏感数据被随机拆分为 $d+1$ 份进行计算,每一份都与原来的状态独立,而且小于 d 份异或结果和原来的状态也是独立的。从表 3 可以看出,每个密钥都没有正确计算出来。该方案降低了中间状态值和实际功耗的相关性,最终密钥没有被破解,而且加密出来的密文和未添加掩码的结果是一样的,表明了设计方案的正确性。

表 3　文中方案结果

	0	1	2	3	4	5	6	7
P	177	78	242	128	163	154	98	36
0	35 0.669 3	48 0.650 9	24 0.687 7	16 0.763 8	A7 0.646 6	AE 0.914 5	8C 0.618 8	C3 0.650 9
1	E2 0.625 2	B6 0.625 0	AF 0.631 6	4D 0.639 7	2E 0.644 6	51 0.614 6	28 0.632 9	8A 0.640 1
2	D9 0.621 7	FF 0.621 6	A5 0.630 3	04 0.631 9	D2 0.636 2	59 0.604 5	E6 0.617 3	23 0.635 8
3	CB 0.616 4	FE 0.603 7	EF 0.624 0	58 0.622 6	3D 0.610 6	B9 0.603 9	16 0.613 7	E5 0.627 3
	8	9	10	11	12	13	14	15
P	88	91	141	134	133	253	214	36
0	E4 0.646 5	AE 0.653 1	1D 0.626 1	A4 0.725 1	CD 0.731 6	7A 0.750 0	00 0.727 9	3A 0.717 7
1	83 0.612 0	8C 0.615 2	B0 0.621 7	BE 0.711 0	92 0.726 7	BF 0.714 3	8B 0.716 2	E5 0.710 3
2	5D 0.611 3	BD 0.615 0	2B 0.615 1	85 0.702 0	6F 0.725 8	A1 0.705 7	9C 0.707 7	C6 0.707 1
3	4D 0.592 7	35 0.613 7	11 0.613 1	70 0.686 3	C4 0.714 1	3C 0.697 4	95 0.702 6	FC 0.687 0

从表 4 可以看出,随着明文条数的增加,添加了掩码的 AES 算法因为消除了功耗之间的相关性,因此能抵抗高阶功耗攻击。

表 4　随机明文功耗相关性对比

明文条数	无掩码	文中方案
1 000	2.443 276	0.125 68
6 000	2.470 210	0.091 25
25 000	2.752 432	0.069 82

5　结束语

提出了一种 $d+1$ 的高阶掩码方案抵抗高阶功耗攻击。该方案以汉明重量为种子刷新掩码,构造了有限域上的安全乘法,采用平方和实现有限域上的求逆变换,再结合仿射变换实现 S 盒的替换,提高了新 S 盒的替换速度,并且保证安全。实验结果表明该方案相比已有的方法在一定程度上有很大优势。尽管该方案是运用在 AES 加密算法上来实现的,但可以运用到类似的密码算法上来抵御功耗攻击。加密算法作为芯片的灵魂,对芯片安全起着重要作用。随着 AI 技术的快速发展,侧信道攻击结合 AI 技术使得加密算法变得更容易破解。如何在保证加密效率不变的情况下,加密算法可以抵御高阶差分功耗攻击将会是未来的研究重点。

参考文献:

[1] KOCHER P, JAFFE J, JUN B. Differential power analysis [C]//Advances in cryptology. Santa Barbara, California, USA: Springer-Verlag, 1999: 388-397.

[2] NIKOVA S, RIJMEN V, SCHLAFFER M. Secure hardware implementation of non-linear functions in the presence of glitches [C]//Information security and cryptology. Seoul, Korea: Springer, 2008: 218-234.

[3] OSWALD E, MANGARD S, PRAMSTALLER N. Secure and efficient masking of AES - a mission impossible cryptology print archive [R]. New York: Elisabeth Oswald, 2004.

[4] RIVAIN M. On the physical security of cryptographic implementations [D]. Luxembourg: University of Luxembourg, 2009.

[5] PIETRZAK K. A leakage-resilient mode of operation [C]// Advances in cryptology - EUROCRYPT 2009. [s. l.]: [s. n.], 2009: 462-482.

[6] OSWALD E, MANGARD S, HERBST C, et al. Practical second-order DPA attacks for masked smart card implementations of block ciphers [J]. Topics in cryptology - CT-RSA. San Jose, CA, USA: Springer, 2006: 192-207.

[7] MARTINASEK Z, ZEMAN V. Innovative method of the power analysis [J]. Radioengineering, 2013, 22 (2): 586-

594.
[8] 刘 鸣. 密码芯片的功耗分析及抗功耗分析研究[D]. 北京:清华大学,2005.
[9] GIERLICHS B, LEMKERUST K, PAAR C. Templates vs. stochastic methods[C]//Proceedings of the workshop on cryptographic hardware and embedded systems (CHESO6). Yokohama:[s. n.],2006:70.
[10] 徐 佩. 智能卡 AES 加密模块抗侧信道攻击掩码技术研究与实现[D]. 重庆:重庆大学,2015.
[11] MAGHREBI H, PORTIGLIATTI T, PROUFF E. Breaking cryptographic implementations using deep learning techniques[C]//International conference on security, privacy, and applied cryptography engineering. Istanbul, Turkey: Springer,2016:3-26.
[12] 童元满,王志英,戴 葵,等. 一种抗 DPA 及 HO-DPA 攻击的 AES 算法实现技术[J]. 计算机研究与发展,2009,46(3):377-383.
[13] CHOUDARY O, KUHN M G. Efficient template attacks[C]//International conference on smart card research & advanced applications. France:[s. n.],2013:75.
[14] RIVAIN M, PROUFF E. Provably secure higher-order masking of AES[C]//Crypto graphic hardware and embedded systems. Santa Barbara, CA, USA:[s. n.],2010:413-427.
[15] ISHAI Y, SAHAI A, WAGNER D. Private circuits: securing hardware against probing attacks[C]//Annual int. cryptology conf. (CRYPTO 2003). California, USA:[s. n.],2003:463-481.
[16] OSWALD E, MANGARD S, PRAMSTALLER N, et al. A side-channel analysis resistant description of the AES S-box[J]. Integration, the VLSI Journal, 2005, 3557(43):413-423.
[17] BLÖMER J, GUAJARDO J, KRUMMEL V. Provably secure masking of AES[C]//Selected areas in cryptography, Waterloo, Canada: Springer,2004:69-83.
[18] PETIT C, STANDAERT F X, PEREIRA O, et al. A block cipher based pseudo random number generator secure against side-channel key recovery[C]//Symposium on information, computer and communications security. Tokyo, Japan: ACM,2008:56-65.
[19] 蔡泽民,王 奕,李仁发. 基于代数表达式功耗模型的差分功耗分析攻击[J]. 计算机应用,2014,34(2):448-451.
[20] LERMAN L, POUSSIER R, BONTEMPI G, et al. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis)[M]//Constructive side-channel analysis and secure design. Berlin, Germany: Springer,2015:20-33.
[21] LIU P C, CHANG H C, LEE C Y. A true random-based differential power analysis countermeasure circuit for an AES engine[J]. IEEE Transactions on Circuits and Systems II: Express Briefs,2012,59(2):103-107.
[22] RAVANELLI M, BENGIO Y. Speaker recognition from raw waveform with SincNet[C]//Proceedings of 2018 IEEE spoken language technology workshop (SLT). Athens (GR): IEEE,2018:1021-1028.
[23] ZHOU W H, KONG F T. Electromagnetic side channel attack against embedded encryption chips[C]//2019 19th IEEE international conference on communication technology. Xi'an: IEEE,2019:30.
[24] BENADJILA R, PROUFF E, STRULLU R, et al. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database[J]. Journal of Cryptographic Engineering,2020,10:163-188.

(上接第 67 页)

search and Development,2019,56(1):58-68.
[11] LEHTO O P. Containers, Meet HPC[EB/OL]. (2015-12-30)[2019-08-12]. https://medium. com/@ ople /containers-meet-hpc-2aab7aa2d54a.
[12] 尹 飞,龙玲莉,孔 峥,等. 面向动态负载的集群容器部署方法[J]. 计算机应用,2021,41(6):1581-1588.
[13] 袁冬冬,姚 伟,卞中昊. 容器技术应用在跨网部署中的自动化研究[J]. 网络安全技术与应用,2020(11):19-20.
[14] 张 晓,张思蒙,石 佳,等. Ceph 分布式存储系统性能优化技术研究综述[J]. 计算机科学,2021,48(2):1-12.
[15] 姚朋成. Ceph 异构存储优化机制研究[D]. 重庆:重庆邮电大学,2019.
[16] LOU C. An analysis of the application scenarios of docker container[EB/OL]. (2018-09-11)[2019-08-06]. https://sq. 163yun. com/blog/article/197408114326077440.
[17] 郑富文. Kubernetes 容器云资源分配与编排策略[D]. 重庆:重庆邮电大学,2020.