

针对增强型旋转 S 盒掩码方案的侧信道安全漏洞系统研究

刘泽艺^{1,2}, 王彤彤¹, 尹芷仪¹, 高能^{1,2}, 查达仁^{1,2}, 屠晨阳^{1,2}

¹中国科学院信息工程研究所, 北京 100093

²中国科学院大学网络空间安全学院, 北京 100049

摘要 增强型旋转 S 盒掩码方案(简称 RSM2.0)是一种全球知名的抗能量分析防御方案。该方案由 DPA Contest 国际侧信道大赛组委会首次提出并实现,旨在为高级加密标准 AES-128 提供高标准的安全防护。通过结合一阶掩码方案与乱序防御这两类经典的侧信道防御技术,组委会宣称 RSM2.0 具备非模板攻击免疫力并且能够抵抗多种已知的模板类攻击。为了验证 RSM2.0 方案的实际安全性,本文首先提出了一种通用的漏洞检测方法用以系统性的定位 RSM2.0 中存在的潜在安全漏洞,并且随后从模板类与非模板类分析两个角度展开研究。模板类研究方面,本文提出了一种泄露指纹利用技术从而能够以近乎 100% 的概率破解 RSM2.0 方案的随机掩码防护。为了进一步降低计算以及存储开销,本文又对泄露指纹技术进行优化并首次提出了“最邻近指纹距离均值”评价指标(MOND 指标)来客观地衡量不同泄露位置选取条件下泄露指纹攻击方案的性能优劣。在非模板类研究方面,我们设计了 4 种不同类型的非模板类二阶攻击方案,这些方案利用 RSM2.0 中乱序防护的设计缺陷,能够有效绕开乱序 S 盒的能量泄露,从而高效的破解全部 128 比特的算法主密钥。在实验验证阶段,我们向 DPA Contest 官方组委会提交了 2 套模板类攻击代码以及 4 套非模板类攻击代码。官方评估结果表明,我们提交的模板类最优方案只需使用 4 条能量曲线以及每条曲线 100ms 的时间开销即可达到 80% 的密钥破解全局成功率(GSR),而非模板类最优方案只需 257 条能量曲线以及每条曲线 50ms 的处理时间开销即可破解 RSM2.0 方案。为了进一步提升 RSM2.0 方案的实际安全性,本文还对 RSM2.0 的改进对策进行了一系列讨论,以便能够有效应对本文中提出的多种类型的安全威胁。

关键词 能量分析攻击; 国际侧信道竞赛; 模板攻击方案; 非模板攻击方案; 增强型旋转 S 盒掩码方案
中图分类号 TP309.7 DOI 号 10.19363/J.cnki.cn10-1380/tn.2019.07.03

Systematic Research on the Side Channel Vulnerabilities of Improved Rotating S-box Masking Scheme

LIU Zeyi^{1,2}, WANG Tongtong¹, YIN Zhiyi¹, GAO Neng^{1,2}, ZHA Daren^{1,2}, TU Chenyang^{1,2}

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

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract Improved Rotating S-box Masking Scheme (RSM2.0 for short) is a world-famous countermeasure against power analysis attacks. This scheme was first proposed and implemented by the committee of DPA Contest, aiming at providing advanced encryption standard AES-128 with high standard security protection. By combining both first order masking scheme and shuffling technique, the contest committee claims that RSM2.0 gains the resistance of non-profiled attacks and is capable to resist several kinds of existing profiled attacks. To study the practical security of RSM2.0, this paper first proposes a general detecting method to systematically locate the potential vulnerabilities in RSM2.0 and then conducts the research from both profiled and non-profiled perspectives. For research on profiled attacks, the paper proposes a “leakage fingerprint” exploitation technique to crack the random masks used in RSM2.0 with nearly 100% accuracy. To further reduce the computation and storage overhead, this technique is further optimized and we put forward for the first time an objective metric “Mean of Nearest Distance” (MOND) to evaluate the performance of fingerprint exploitation under the condition of selecting different leakage positions. For research on non-profiled attacks, we devise 4 kinds of second order schemes which can bypass the shuffling countermeasure to crack the whole 128-bit master key with high efficiency. In the phase of experimental validation, we upload to the official committee two profiled attacks and four non-profiled attacks. The official evaluation results show that the best profiled attack of ours needs only 4 traces and 100ms processing time per trace to reach 80% Global Success Rate (GSR), and the best non-profiled attack requires 257 traces and 50ms per trace to compromise RSM2.0. To further improve the practical security level of RSM2.0, this paper

通讯作者: 尹芷仪, 博士, 高级工程师, Email: llinshi@163.com。

本课题得到优秀青年科学基金(No. Y710061103)项目资助。

收稿日期: 2017-10-18; 修改日期: 2018-02-13; 定稿日期: 2019-06-06

also make a series of discussions on possible improvement strategies, thus setting obstacles for the threats proposed in this paper.

Key words power analysis attack; DPA contest; profiled attacks; non-profiled attacks; improved rotating s-box masking scheme

1 研究背景

1.1 低熵掩码结合乱序防御

自从能量分析技术由 Kocher 首次提出以来, 大量研究成果^[1-4]表明绝大多数密码算法的标准软硬件实现均无法抵抗能量分析攻击的威胁。因此在密码算法的具体实现中引入能量分析攻击抵抗能力是当前密码算法在实际部署中的一个必不可少的环节。

能量分析攻击主要利用密码芯片在运行过程中产生的与计算中间数据以及操作指令相关的能量泄露信息来进行密钥恢复。因此, 为了抵抗这种攻击, 切断上述能量泄露中的依赖性与关联性就成为能量分析防御技术的核心理念。

掩码技术和乱序技术是两类研究的最为深入的经典防御技术。通过引入随机数的方式, 掩码方案^[5-6]将原始密码算法中的敏感中间值划分为多个独立的子部分, 同时保证每一部分中间值的随机性。这种方案有效切断了敏感中间值与真实能量泄露之间的内在关系, 因此能够抵抗侧信道领域常见的能量分析方法, 如文献[7-8]。另一方面, 乱序方案^[9, 10]则从时间维度为密码实现提供安全性保护。在随机索引表的帮助下, 乱序防御方案通过在算法执行过程的不同时刻插入随机延时, 或者打乱密码算法各指令操作的执行顺序等方式来随机化敏感中间值所对应的能量泄露出现时刻, 进而为绝大多数依赖固定时刻泄露点的能量分析方法设置攻击障碍。

然而, 无论是掩码技术还是乱序技术其在实际部署中依然存在诸多问题。从安全性的角度来看, 理论上 n 阶掩码方案无法抵抗 $n+1$ 阶攻击, 即攻击者通过结合使用 n 阶掩码方案划分出的 $n+1$ 个独立拆分值的能量泄露即可获得与未拆分敏感中间值相关联的联合能量泄露信息, 进而可以使用该联合泄露值进行密钥恢复。独立使用的乱序方案同样存在相关的攻击案例。由于受到密码算法计算逻辑的制约, 乱序方案往往只能在受限制的密码算法操作范围内执行(如 AES 算法 16 个 S 盒变换之间)。这种情况下, 通过累加求和上述有限范围内所有能量泄露的方式^[11], 攻击者仍然能够获得与无保护敏感中间值相关的累计能量泄漏信息。另一方面, 从资源开销的角度来讲,

掩码方案所带来的算法实现安全性提升是通过更多的占用电路资源换取的。由于在随机掩码的生成, 存储, 计算, 消除等各个环节中均需要额外的操作进行处理, 传统掩码方案不可避免的会引入额外的电路资源开销。

为了抵抗针对单一防御策略的现有攻击, 在获得更高安全等级的同时又能尽可能的限制额外资源开销, 在设计能量分析防御方案的过程中结合使用掩码与乱序两种防御技术^[12-15]已经成为当前防御方案设计方向的主流趋势。

在上述防御方案设计中, 改进型旋转 S 盒掩码方案(下文称为 RSM2.0)作为近几年提出的一种低熵掩码防御方案将低开销与高安全性并重的设计理念发挥到了极致。通过将随机掩码的取值范围限定在 16 个固定元素中并且巧妙设计 S 盒输入输出掩码之间的转换关系, 该方案大大降低了整体方案的额外资源开销。另一方面, 该低熵掩码中同时引入了乱序设计对算法重点位置的能量泄露进行了有效保护, 进而与低熵掩码方案一道为整体密码算法提供较强的安全保证。具体来说, 与原始版本的旋转 S 盒掩码方案(简称 RSM)不同的是, RSM2.0 方案通过引入乱序索引数组的方式分别对 AES 算法首轮与末轮中 16 个非线性 S 盒变换的执行顺序进行随机化保护, 进而能够在单条能量曲线上有效隐藏 S 盒能量泄露的产生位置。而从掩码方案上来看, RSM2.0 方案使用新引入的随机偏移数组取代原始版本中的单一索引变量, 从而保证 AES 状态矩阵中的每个字节位置均能够使用独立的随机掩码进行保护。通过使用这两种全新的防护方案设计, RSM2.0 复合防御方案的实际安全强度得到了极大的提高。

1.2 DPA contest 国际侧信道大赛

DPA contest 侧信道大赛始于 2008 年, 其由法国科学院及巴黎高科电信研究院联合举办, 旨在展示和验证国际最前沿的侧信道攻防技术, 并为全球的密码分析人员提供客观公正的能量分析竞赛平台。

在当前第四阶段第二子阶段的竞赛中, 低熵掩码与乱序相结合的 RSM2.0 软件方案由 DPA Contest 大赛组委会首次实现并被选为本阶段竞赛的攻击目标, 在全球范围内征集攻击方案。随后, 越来越多的研究人员对这一新型防御技术产生了浓厚的研究兴

趣, 在组委会搭建的竞赛平台中以及赛场之外大量针对该新型防御方案的研究工作也随之展开。

本文剩余部分的组织结构如下: 第 2 章阐述了 RSM2.0 防御方案的命名由来、核心加密模块以及该方案的详细算法流程; 在第 3 章中, 当前已有的针对 RSM2.0 的相关研究工作以及现有研究存在的局限性首先被加以分析, 为了弥补现有工作的不足, 本章进一步提出针对 RSM2.0 的系统性研究框架; 依照该研究框架, 本文在第 4、5 章中详细阐述了针对 RSM2.0 防御方案的模板类攻击方案以及非模板类攻击方案, 并在第 6 章为本文所指出的安全漏洞设计针对性的防御对策, 以期能够完善 RSM2.0 的防御方案设计, 从而全面提升 RSM2.0 方案的安全等级。

2 RSM2.0 算法描述

本章首先对 RSM2.0 方案的实现过程进行回顾, 包括引入的新特性, 新修改的固定掩码数组以及该方案与原始的 RSM 方案的一些异同点。值得注意的是, 为了表述的方便, 下文中除非特殊声明, 否则所有的加法符号均是在模 16 的意义下进行定义的。

2.1 RSM2.0 核心功能模块

RSM2.0 方案是一个基于 AES-128 核心密码算法的布尔掩码方案^①。相比于原始的 RSM 方案, 该增强型方案的安全性完全依赖于新引入的三个数组, 他们分别是固定掩码数组 $M[]$, 随机偏移数组 $O()$ 以及乱序数组 $Sf[]$ 。

对于掩码数组而言, 在算法设计中该数组的元素是预先固定并且公开可知的。数组中包含 16 个两两不同的 8 比特固定掩码值且这些掩码值是依照参考文献[16]中的最小化互信息泄露的准则进行严格选取的。在 V4.2 阶段的竞赛过程中, 考虑到侧信道不可区分性以及安全性等问题, 该固定掩码数组被重新更新^[17]。最新的固定掩码数组使用取自汉明重量集合 $\{2, 4, 6\}$ 的掩码值来取代原本取自 $\{0, 4, 8\}$ 汉明重量集合的原始掩码元素。掩码数组中的所有元素信息 $M[i], i \in [0, 15]$ 如下所示。

$$M = \{0x03, 0x0c, 0x35, 0x3a, 0x50, 0x5f, 0x66, 0x69, 0x96, 0x99, 0xa0, 0xaf, 0xc5, 0xca, 0xf3, 0xfc\}$$

对于随机偏移数组而言, 在每次算法加密的过程中该数组均需要重新更新, 且数组中的所有元素对于攻击者来说均是未知的。随机偏移数组包含 16 个元素, 每个元素的取值范围从 0 到 15。该数组的

作用是与固定掩码数组配合使用, 从而获得 RSM2.0 方案中掩码使用的实际随机性。更为确切的说, 每个随机偏移值 $O[i], i \in [0, 15]$, 唯一的决定了一个选定的掩码值, 即 $M[O(i)]$ 。该选定掩码值随后与加密算法中的敏感中间值相异或, 从而将所有敏感中间值进行随机化处理。由于每个随机索引值 $O(i)$ 都是独立的且服从均匀分布, 每个 $M[O(i)]$ 元素服从取值范围为 16 个固定掩码值的均匀分布, 因此称该掩码方案为 4 比特熵值的低熵掩码方案。

乱序数组与随机偏移数组一样在每次加密过程中进行刷新, 并且该数组对于算法分析人员来说也是未知的。与随机偏移数组不同的是, 本数组是整数 0 到 15 的一个随机置换用以对 16 个 S 盒的执行顺序进行置乱。实际上, 在每次加密过程的首轮与末轮中会用到两个独立的乱序数组 $Sf0[]$ 和 $Sf10[]$ 。具体来说, 在第一轮与第十轮的 16 个输入状态字节分别根据 $Sf0[]$ 和 $Sf10[i], i \in [0, 15]$ 的索引顺序来进行非线性层的置换操作。

通过对上述三个 RSM2.0 中主要数组的描述, 现在可以继续阐述算法中使用的 5 大基本功能模块。在这些模块中, 掩码补偿模块是 2.0 方案中新引入的计算模块, 它的作用是去除用以进行敏感中间值保护的随机掩码在经过线性层与非线性层变换之后产生的中间掩码结果, 从而确保加密算法可以最终得到正确的密文输出。

➤ 轮密钥加 (AddRoundKey, AR):

对经过随机掩码保护的状态字节与每轮中的轮密钥字节执行按位异或操作, 该操作与标准 AES-128 中的操作保持一致。

➤ 带掩码的字节代换 (MaskedSubBytes, MS):

作为非线性层变换, 该操作包含 16 个不同的掩码 S 盒变换。其中的每个 S 盒都由标准 AES-128 中的 S 盒衍生而来以此保证 S 盒输出掩码的可追踪特性。换言之, 每一个重构掩码 S 盒的功能是将一个被掩中间值(如 $X \oplus M$)映射为一个使用新的输出掩码进行保护的输出中间值(如 $X' \oplus M'$)。这样的设计目标能够通过使用下述公式进行描述:

$$MaskedSubByte_{MM'}[X] = SubByte[X \oplus M] \oplus M',$$

这其中 $SubByte[]$ 代表标准 AES-128 中的 S 盒变换, X 代表作为重构 S 盒输入的受保护中间值, M, M' 分别代表用以进行敏感中间值保护的输入掩码与输出掩码值。特别地, 在 RSM2.0 设计中, M 和 M' 被

① 依照最新版本的官方源代码^[18]

设计为固定掩码数组 $M[]$ 中相邻的两个元素。因此, 新构造的掩码 S 盒总共有 16 种形式, 即 $MaskedSubByte_{M[i]M[i+1]}$ (简记为 $MaskedSubByte_i$), $i \in [0, 15]$ 。其中最后一个固定掩码元素 $M[15]$ 与第一个固定掩码元素 $M[0]$ 构成了最后一个重构掩码 S 盒的输入输出掩码, 这样一来 16 个重构 S 盒对于固定掩码表中掩码元素的使用构成了一个环形闭路。而这也正是“旋转 S 盒”防御方案名称的来由。

➤ 行移位 (ShiftRows, SR):

将第 i 行中的 4 个元素向左循环左移 i 个字节, 其中 $i \in [0, 3]$ 。该操作与标准 AES-128 中的行移位操作保持相同。

➤ 列混淆 (MixColumns, MC):

执行列混淆操作的最小数据单元为 AES 状态矩阵中的一列四个元素。由于列混淆层的线性特性, 受掩码保护的一列输入元素将会变换为受另外一组计算中间掩码保护的输出元素, 如下列的公式所示:

$$MC(X1 \oplus M1, X2 \oplus M2, X3 \oplus M3, X4 \oplus M4)^T \\ = MC(X1, X2, X3, X4)^T \oplus MC(M1, M2, M3, M4)^T$$

这其中 X_i 和 M_i , $i \in [1, 4]$ 分别指的是 4 个输入字节和 4 个输入掩码值。同时由于列混淆层的线性特性, 当输入掩码已知的情况下输出字节的掩码值可以按照上述公式计算出来, 正因为如此, 后续的掩码补偿过程才能得以正确的执行。

➤ 掩码补偿 (MaskCompensation, MCP):

作为相邻加密轮之间掩码转换的重要环节, 掩码补偿在列混淆步骤之后进行。除了以循环使用固定掩码表中掩码值的方式构造掩码 S 盒之外, RSM2.0 的另外一大特点是每轮中用以进行输入状态矩阵保护的输入掩码值。在 RSM2.0 中, 每轮的 16 个输入字节被设计为分别与掩码值 $M[O[i]+r]$, $i \in [0, 15]$ 进行异或, 其中 $r \in [0, 9]$ 代表加密算法的轮数, 即第一轮到第十轮。本章中将这种在第 $r+1$ 轮中使用的“输入掩码状态矩阵”记为 $Mask_r$, 该矩阵的定义如下所示:

$$\begin{bmatrix} M[O(0)+r] & M[O(4)+r] & M[O(8)+r] & M[O(12)+r] \\ M[O(1)+r] & M[O(5)+r] & M[O(9)+r] & M[O(13)+r] \\ M[O(2)+r] & M[O(6)+r] & M[O(10)+r] & M[O(14)+r] \\ M[O(3)+r] & M[O(7)+r] & M[O(11)+r] & M[O(15)+r] \end{bmatrix}$$

为了完成这一设计目标, 掩码补偿操作(MCP)可以被划分为如下三个步骤: 首先每轮中受保护中间值经过列混淆之后得到的输出掩码结果被首先计算出来。此后, 下一轮中根据 RSM2.0 设计要求所使

用的输入掩码状态矩阵与输出掩码结果进行异或, 从而获得完整的补偿掩码状态矩阵, 本章中记为 $MaskCompensation_r$ (缩写为 MCP_r), 其中 $r \in [0, 8]$:

$$MCP_r = MC(SR(Mask_{r+1})) \oplus Mask_{r+1}$$

最终, 通过将第 $r+1$ 轮中的列混淆输出值与上述补偿掩码状态矩阵 MCP_r 进行异或, 当前轮中的列混淆输出掩码值被有效消除而剩余的敏感中间值部分同时又被下一轮输入掩码值加以保护, 从而完成轮与轮之间掩码值的转换, 继而能够满足 RSM2.0 方案的设计要求。下面的公式是第 $r=9$ 轮掩码补偿层的完整公式推导, 其中 $(X_r \oplus Mask_r)$ 以及 K_r 分别代表受保护的输入状态矩阵以及当前轮的轮密钥。

$$\begin{aligned} & MC(SR(MS(K_r \oplus X_r \oplus Mask_r))) \oplus K_{r+1} \oplus MCP_r \\ & = MC(SR(SubBytes(K_r \oplus X_r) \oplus Mask_{r+1})) \oplus K_{r+1} \oplus MCP_r \\ & = MC(SR(SubBytes(K_r \oplus X_r))) \oplus MC(SR(Mask_{r+1})) \\ & \oplus K_{r+1} \oplus MC(SR(Mask_{r+1})) \oplus Mask_{r+1} \\ & = MC(SR(SubBytes(K_r \oplus X_r))) \oplus K_{r+1} \oplus Mask_{r+1} \end{aligned}$$

需要特别关注的问题在于, 在最后一轮加密轮 ($r=9$) 中不存在列混淆操作。此外, 第十轮的补偿掩码状态矩阵不需要包含下一轮的输入掩码状态部分, 以便在进行最后一轮的掩码补偿转换之后能够获得正确的 AES-128 密文信息。因此, 第十轮的补偿掩码状态矩阵 $MaskCompensation_9$ 与先前轮中的形式均不相同, 该状态矩阵可以由下面的公式进行描述:

$$MaskCompensation_9 = ShiftRows(Mask_{10})$$

这样, 第十轮的掩码补偿过程可以被一般化的描述为如下过程:

$$\begin{aligned} & SR(MS(K_9 \oplus X_9 \oplus Mask_9)) \oplus K_{10} \oplus MaskCompensation_9 \\ & = SR(SubBytes(K_9 \oplus X_9) \oplus Mask_{10}) \oplus K_{10} \oplus SR(Mask_{10}) \\ & = SR(SubBytes(K_9 \oplus X_9)) \oplus K_{10} \\ & = 128\text{-bit Ciphertext} \end{aligned}$$

2.2 RSM2.0 算法流程

通过对上面 5 大基本功能模块的描述, RSM2.0 的具体算法流程可以如算法 1 中一样描述。值得关注的是, 在每次加密操作开始之前, RSM2.0 方案的密钥扩展部分首先被调用, 而该部分运算不含任何保护措施, 因此包括文献[19-21]在内的多种攻击方式均会对该实现方式造成实际的安全威胁。由于这部分内容并不是本文关注的重点, 因此这里不展开进行讨论。

算法 1. RSM2.0 低熵掩码方案.

输入:

16 字节明文输入: $X = [X_0, X_1, \dots, X_{15}]$;

随机偏移数组: $O(i), i \in [0, 15]$;

乱序数组: $Sf0[i], Sf10[i], i \in [0, 15]$;

输出:

16 字节密文输出 $X = [X0, X1, \dots, X15]$;

1: 在线密钥扩展过程, 获得 $RoundKey_r$, $r \in [0, 10]$,

2: $RoundKey_0 \leftarrow RoundKey_0 \oplus Mask_0$

3: $X = X \oplus RoundKey_0$

/*除最后一轮外的每轮轮函数操作*/

4: FOR {each $r \in [0, 8]$ }

5: IF { $r = 0$ }

6: FOR { $i \in Sf0[0, 15]$ }

7: $X_i = MaskedSubBytes_{O(i)+r}(X_i)$

8: ENDFOR

9: ELSE

10: FOR { $i \in [0, 15]$ }

11: $X_i = MaskedSubBytes_{O(i)+r}(X_i)$

12: ENDFOR

13: ENDIF

14: $X = SR(X)$

15: $X = MC(X)$

/* 当前轮的轮密钥加操作 */

16: $X = X \oplus RoundKey_{r+1}$

17: $MCP_r = MC(SR(Mask_{r+1})) \oplus Mask_{r+1}$

18: $X = X \oplus MCP_r$;

19: ENDFOR

/* 最后一轮 */

20: FOR { $i \in Sf10[0, 15]$ }

21: $X_i = MaskedSubBytes_{O(i)+9}(X_i)$

22: ENDFOR

23: $X = SR(X)$

24: $X = X \oplus RoundKey_{10}$

/* Ciphertext unmasking */

25: $MCP_9 = SR(Mask_{10})$

26: $X = X \oplus MCP_9$;

非模板攻击中, 攻击者主要利用真实能量泄露中的数据依赖性能量泄露分量或者操作依赖性能量泄露分量来开展实际的分析, 而无法对目标密码芯片的真实能量泄露模型进行事先刻画学习。而在模板类攻击中, 攻击者有能力准确刻画密码算法计算中间值与其所对应的真实能量泄露之间的映射关系, 因此模板类分析方法拥有比非模板类分析方法更强的实际攻击能力。

从官网上公开可查的攻击方案描述上来看, 最早提交攻击方案的匿名参赛者通过利用官方组委会提供的预处理数据事先对密码设备进行了泄露刻画。随后, 在攻击阶段, 该参赛者在先后破解随机偏移数组与首轮乱序数组之后分别利用标准 CPA 与标准模板攻击对算法主密钥进行了破解。Liu Junrong 与 Li Yang 等人^[22]的攻击思路同样类似, 通过利用模板攻击分别恢复随机偏移、乱序索引以及 16 个掩码 S 盒的输出值, 攻击者可以直接推导出每个 S 盒中包含的子密钥信息。2015 年 8 月份由另一个匿名参赛人员提交的攻击方案则较为新颖。该方法采用 K-means 聚类算法对能量曲线中的泄露进行事先聚类并构造电码本, 随后在实际攻击阶段通过匹配真实能量泄露与电码本现有记录的方式来对所需的算法敏感信息进行恢复。综合来看, 上述四种攻击方案均需要对随机化掩码 S 盒执行顺序时使用的乱序数组进行实际破解。

而 Zdenek Martinasek 等^[23]研究人员首先使用标准的多变量高斯模板对 RSM2.0 中的偏移数组进行恢复。随后在方案描述中, Martinasek 明确阐述了其所在团队通过利用行移位操作位置的能量泄露来直接恢复算法主密钥, 从而绕过乱序防御方案的攻击方法。然而, 通过查看 RSM2.0 的汇编实现源代码, 我们发现该绕过方案是不可行的。其主要原因在于, 在具体实现中, RSM2.0 方案的行移位操作同样受到乱序操作保护, 因此针对行移位操作的攻击并无法绕开乱序防御的影响。而之所以该攻击团队能够以 104 条能量曲线成功破解 RSM2.0 方案, 其原因在于所提交的攻击方案实际上利用了顺序执行的列混淆操作的能量泄露(在第 4.3.1 小节中进行了阐述)。另外, Hideo Shimizu 等人直接利用标准模板攻击对偏移数组与掩码 S 盒输出中间值进行了攻击, 其中后者的能量泄露位置尽管没有明确阐述应该同样来自于上述的列混淆操作。

除此之外, Tobias Schneider 等人在文献[24]中研究了 Welch 的 T 测试方案在不同场景下的实际适用程度, 并将一种由此改进的泄露测试方案应用于

3 针对 RSM2.0 方案的系统性研究框架

3.1 已有研究工作与其局限性

3.1.1 现有研究工作

截止目前, DPA Contest 竞赛 V4.2 阶段中针对 RSM2.0 实现共提交了 17 种有效的攻击方案, 其中非模板类型攻击 8 种, 模板类攻击 9 种(数据源自 DPA Contest V4.2 阶段官方网站^[22])。

V4.2 阶段的能量曲线集合中, 用以协助检测 RSM2.0 算法实现中潜在的安全漏洞。不过, 该文章并没有进一步研究如何利用这些泄露来进行实际的攻击。Zdenek Martinasek 则在文献[25]中又提出了一种利用机器学习中的多层感知器算法来进行掩码 S 盒输出位置泄露利用的攻击方案。

除了上述提及的研究之外, 已通过官方组委会评估的其他攻击方案并未提交相应的攻击方案描述, 且到本文撰稿之时也没有公开发表的文献可查, 因此在本文中不对这类攻击方案进行讨论与分析。

3.1.2 当前研究的问题

综合上述针对 RSM2.0 低熵掩码方案的已有研究来看, 除了作者本人所在参赛团队提出的攻击方案之外, 其他已知的攻击方案主要存在以下几点问题:

- 破解乱序数组带来额外计算开销与更多的能量曲线使用量。大多数已提交方案的攻击过程均分为三个步骤, 即首先利用模板攻击方法破解随机偏移数组, 随后破解用以执行掩码 S 盒乱序保护的乱序数组, 最后再对掩码 S 盒输出位置的能量泄露进行能量分析。从官方组委会公布的提交方案性能评估结果^[22]上来看, 这种攻击策略会造成很大的额外计算时间开销, 并且由于乱序数组恢复时引入的偏差可能造成更多的能量曲线使用量。
- 在随机掩码破解阶段, 所提出的绝大多数攻击方案仅针对单一位置的掩码能量泄露点进行建模与攻击, 忽略了对其他计算位置中存在的相关能量泄露进行利用, 从而造成随机掩码破解过程效率低下, 破解正确率较低等问题。在密钥恢复阶段, 已有攻击方案所选择的目标中间值位置过分集中(均利用首轮乱序 S 盒输出位置的泄露进行密钥破解), 而没有对 RSM2.0 中潜在的其他安全漏洞进行系统、全面的挖掘, 因此整体攻击方案从各个角度来看均存在很大的改进空间。
- 已有攻击方案中所利用的安全漏洞并没有触及 RSM2.0 方案的核心设计理念(即 S 盒输入输出掩码位置相邻以及相邻加密轮中掩码轮转使用这一特点), 因此这些攻击只能暂时存在, 容易被后续的改进版本所修复(例如直接针对当前掩码泄露点添加乱序防护即可)。
- 几乎所有的攻击方案在具体实现阶段部分或是全部的使用了基于设备泄露刻画建模的思想。因此, 这些攻击方案实际上均属于模板攻击类

型(或者称为刻画攻击, profiled attack)。换言之, 尽管这些攻击方法切实有效, 但它们均没有对 DPA Contest 官方组委会所宣称的 RSM2.0 具备非模板攻击免疫力这一安全声明构成实际的影响。

另一方面, 在目前 RSM2.0 低熵掩码方案已经存在较多实际攻击案例的背景下, 加紧研究并提出有针对性的改进方案同样应当也是系统性研究 RSM2.0 掩码方案的重要组成部分。然而, 从现有已公开发表的参考文献中我们并未找到任何与该主题相关的研究成果。

3.2 系统性研究框架

为了解决上述现有攻击方案中存在的诸多问题, 本文在文献[26-27]工作的基础上进一步提出了针对 RSM2.0 的系统性漏洞检测方法, 对泄露指纹技术的资源开销情况以及相应的优化技术进行了全面的阐述与分析, 同时针对本文所提出的一系列攻击方法给出了全面的防御对策分析, 形成了包含 RSM2.0 方案的模板类攻击技术研究, 非模板类攻击技术研究, 以及相应的防御方案研究这三大类主题在内的系统性研究框架。

从模板类攻击技术的研究角度来看, 针对上一节中提到的破解乱序数组所带来的较大计算开销这一问题, 我们首先分析了乱序操作在 RSM2.0 方案中的有效保护范围。实际上, 通过对算法实现细节的初步分析后我们发现, 在算法第一轮加密中执行的乱序防护只针对掩码 S 盒与行移位变换这两个操作展开, 而在随后进行的列混淆、轮密钥加、掩码补偿操作中, 未受乱序操作保护的中间值被加载与使用。基于此, 本文首先充分发掘了算法实现过程中存在的未受乱序方案保护的敏感信息泄露位置, 进而利用这些隐藏的能量泄露巧妙规避对乱序数组进行直接破解的需求。

针对随机掩码破解阶段所利用泄露点位置单一, 所使用攻击方法掩码恢复准确率较低等问题, 我们的研究思路是从整条能量曲线的多个泄露位置尽可能多的寻找彼此相关联的掩码泄露信息, 并配合使用模板攻击方案来提升准确率。通过利用这一系列的相关泄露信息, 攻击者可以使用诸如多数表决等类似的修正算法在单一泄露位置恢复准确率偏低的情况下提高最终随机掩码识别的准确率, 同时也能提高攻击方案在其他低信噪比环境下的鲁棒性与通用性。此外, 我们考虑到的一点是, 我们所寻找的多位置掩码泄露信息应当尽可能从 RSM2.0 特有的算法设计流程中进行发掘, 这样一来, 想要防止这种

攻击方式的防御方案对策将需要从根本上修改 RSM2.0 的算法设计理念, 如此我们所提出的攻击方案的实际威胁性才能得以最大化。

实际上, 上述提到的模板类攻击方案与文献[28]中提出的针对高阶防护^[29]的攻击方案在思路上不谋而和, 两者均属于水平能量攻击的范畴。主要的不同点在于本文充分利用了 RSM2.0 轮间掩码使用上的设计缺陷展开攻击, 因此具有更好的针对性以及攻击效果, 而后者则是一种通用性更强的利用水平方向上目标中间值拆分量直接相关的联合能量泄露而展开的极大似然模板攻击方案。

从非模板类攻击技术的研究角度来看, 本质上 RSM2.0 方案属于一阶掩码方案, 而针对一阶掩码方案最为典型的非模板攻击技术即为二阶攻击方案。通过事先对使用了相同掩码进行保护的两个中间值的能量泄露进行二阶预处理, 攻击者可以在后续攻击阶段中使用经典的一阶攻击方法如 DPA, CPA 等直接对算法主密钥进行恢复。

针对 RSM2.0 方案进行二阶攻击的主要难点在

于: 首先由于每个状态字节位置使用独立的随机掩码进行保护, 因此不存在可以被二阶攻击利用的两个使用了相同掩码值进行保护的中间值泄露。另一方面, 由于乱序防护的存在, 掩码 S 盒位置的常见泄露点无法被二阶攻击加以利用。因此, 我们提出的另一种可能的二阶攻击思路是在掩码 S 盒泄露位置之外寻找不受乱序保护但被掩码保护的敏感中间值, 同时寻找该中间值中所使用掩码的自身能量泄露信息。通过联合处理这两部分能量泄露, 传统的二阶攻击方案即可再次生效。

最后, 从防御方案的研究角度来看, 我们的提出的防御技术分为针对模板类泄露指纹攻击技术的防御对策与针对非模板类攻击的防御对策。他们主要针对已有的攻击手段以及本文在 RSM2.0 方案研究中新发现的潜在模板与非模板类漏洞展开针对性的方案改进, 如可以改进乱序防护的作用范围, 避免单次加密过程中多个位置的相关掩码信息泄露等。

综上所述, 本文针对 RSM2.0 实现的系统性研究框架如图 1 所示。

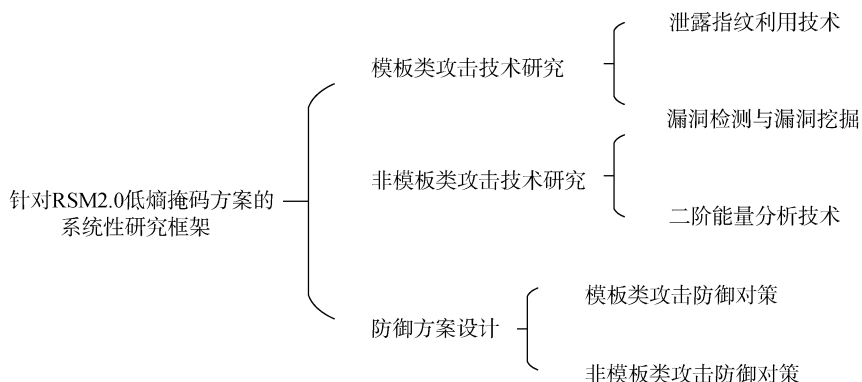


图 1 针对 RSM2.0 防御方案的系统性研究框架

Figure 1 Systematic research framework aiming at RSM2.0 countermeasure

3.3 RSM2.0 安全漏洞检测

在开始研究针对 RSM2.0 方案的模板类与非模板类攻击之前, 本节首先阐述了一种通用的侧信道漏洞检测方法。利用该检测方法, 攻击者可以快速缩小算法中潜在漏洞的搜索范围, 有效系统地定位防御方案中存在的潜在安全隐患。

由于 RSM2.0 方案设计人员对 AES-128 中常用的目标中间值、能量泄露位置均进行了安全防护, 我们不仅需要寻找全新的可以用以进行掩码消除的攻击点, 并且该攻击点应当尽量绕开 RSM2.0 方案中的乱序防护部分, 以此来规避高开销的能量分析攻击过程。

为了完成上述目标, 我们从 RSM2.0 方案的算法

设计以及代码实现这两个层面来执行全面的漏洞检测。为了获得最优的攻击性能并且能够系统性的检测出潜在的大多数漏洞, 本文提出了如下三条漏洞检测准则来指导整个漏洞搜索过程:

- 漏洞搜索范围应当包括含有 8 比特子密钥的所有算法中间值。尽管直接针对更大密钥量的侧信道攻击^[30]是实际可行的, 但其较高的资源开销(如需要 GPU 加速), 以及过大的内存占用量等问题使得这种攻击方式效率低下, 并不适合在 DPA Contest 国际竞赛平台中使用。因此, 为了获得更加高效的攻击性能, 本漏洞检测准则只将关注点放在 8 比特的子密钥恢复上。
- 漏洞搜索过程应当重点关注作为非线性层变换

结果的密码算法计算中间值位置。非线性层变换的混淆特性导致了这样一个攻击结果, 即在能量分析攻击中对于目标子密钥任意一比特值的猜测错误都将导致猜测中间值中平均一半数量比特位的预测错误, 从而使得正确密钥与错误密钥之间的可区分性更加明显。这样一来, 攻击者就能够使用尽可能少的能量曲线来恢复对应的正确密钥值。

- 此外, 为了进行掩码保护而在 RSM2.0 中新增的补偿掩码生成、掩码补偿过程可能引入全新的安全漏洞, 因此针对这些额外操作的漏洞检测也同样需要单独进行。

基于上述的漏洞搜索原则, 针对 RSM2.0 掩码方案的模板类与非模板类漏洞检测可以被有效的限制在图 2 所示的敏感区域中。

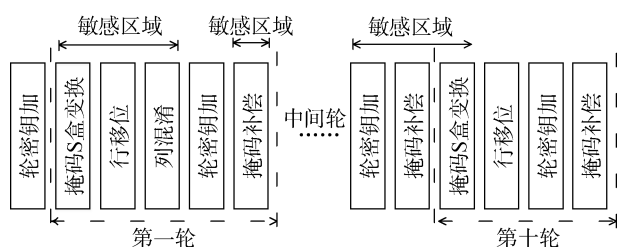


图 2 基于漏洞检测准则的敏感区域定位

Figure 2 Sensitive region location based on vulnerability detection criteria

4 模板类安全漏洞分析

本章首先针对 3.1.2 节中阐述的当前 RSM2.0 模板类攻击方案中存在的诸多问题, 提出了一种泄露指纹分析技术用以破解 RSM2.0 防御方案中使用的随机掩码以及随机偏移索引。

随后, 考虑到该方案可能引起的计算资源开销等问题, 本章随后又对泄露指纹方案的优化技术展开了较为深入的分析。

最后, 基于 DPA Contest 国际侧信道大赛平台, 本文展示了这种泄露指纹利用技术在实际攻击中的可行性与高识别准确性。竞赛组委会的官方评估结果证实了本章中所提出的整套攻击理论的合理性与正确性。

4.1 泄露指纹利用技术

在本节中, 我们首先给出了密码算法在实际运行中存在的“泄露指纹”的准确定义。随后我们通过构建数学通用模型的方式对泄露指纹的利用过程进行了一般性的公式化阐述。最后, 针对软件实现的

RSM2.0 密码算法, 本节详细阐述了泄露指纹利用技术给该防御方案所带来的模板类安全威胁。

4.1.1 泄露指纹通用模型

众所周知, 人体指纹是一种可以用作个人身份识别的生物学纹理特征。类似的, 本章中提出的“泄露指纹”被定义为单条能量曲线中一系列相关泄露信息所构成的一种泄露模式或者是泄露序列, 而这种泄露序列能够用来唯一的识别这些相关泄露所对应的敏感信息。该泄露指纹的一个主要特征在于任何单独时刻点的信息泄露均不足以恢复其所对应的敏感信息, 然而当一系列时刻点上的泄露信息得以累积后, 相应敏感信息的识别就变得清晰与准确。

为了一般化泄露指纹的利用过程, 这里首先将单次算法加密过程中存在相应泄露的可利用中间值定义为 $f_j(X_j, S_j)$, $j \in [0, n-1]$, 其中 j 代表第 j 个能量泄露位置, $X_j \in \{x_1, x_2, \dots, x_p\}$ 以及 $S_j \in \{s_1, s_2, \dots, s_q\}$ 分别代表第 j 个位置上攻击者已知的信息(如明文或密文)以及未知的敏感信息(如子密钥, 随机掩码等)。基于上述定义, 我们将一段长度为 n 的泄露序列定义为如下形式:

$$T_{fp} = L \circ Seq\{f_j(X_j, S_j) \mid j \in [0, n-1]\} \\ + Seq\{\varepsilon_j \mid j \in [0, n-1]\}$$

其中, $L: Z \rightarrow R$ 代表目标密码设备的真实泄露函数, 该函数将集合 $Seq\{\}$ 中的每一个计算中间值映射为设备的真实泄露消耗, 而 ε_j 则表示第 j 个能量泄露位置的随机噪声。

对于一个可利用的泄露序列, S_j 应当满足下面的递推关系:

$$S_j = g_{j-1}(S_{j-1}), \quad j \in [1, n-1]$$

其中, g_{j-1} 指的是相邻位置的敏感信息间存在的公开已知的函数关系。在这样的条件下, 一方面 S_0 唯一的决定了 T_{fp} 中的所有未知变量值(假设不考虑噪声的影响), 另一方面对于敏感信息 S_0 的识别过程又在极大程度上依赖这一泄露指纹序列, 因此本章中称满足上述条件的泄露序列为 S_0 的泄露指纹, 记为 $T_{fp(S_0)}$ 。

现在从攻击者的角度来考虑如何利用泄露指纹来恢复敏感信息。当获取一条能量曲线并且在 X_j , $j \in [0, n-1]$ 公开已知的情况下, 攻击者能够通过穷举 S_0 取值范围内的所有可能取值来计算出 q 种可能

的猜测指纹序列 $W \circ Seq\{f_j(X_j, S_j) | j \in [0, n-1]\}$ 。其中, W 是攻击者猜测的目标设备的能量泄露模式, 用以作为设备真实泄露模式 L 的一种近似估计。更为准确的说, 假设选定 si 作为 S_0 的候选值, 猜测指纹泄露序列 $G_{fp(si)}$ 可以表示为如下形式, 其中, g^{n-1} 表示 $n-1$ 个连续映射 $g_j, j \in [0, n-2]$ 所构成的复合运算操作符:

$$W \circ Seq\{f_0(X_0, si), f_1(X_1, g_0(si)), \dots, f_{n-1}(X_{n-1}, g^{n-1}(si))\}$$

这样, 对于敏感信息 S_0 , 它的识别过程可以使用下面的公式(1)进行描述, 其中 **DIST** 是用于评估设备真实泄露指纹 T 与各个猜测指纹 G 之间相似程度的区分器函数。根据不同的攻击场景, 攻击方式, 该区分器的最终选择可以有多种不同的形式, 如皮尔逊相关系数, 欧氏距离等。

$$S_0 = \underset{si}{\operatorname{argmin}}\{\mathbf{DIST}(T_{fp(S_0)}, G_{fp(si)})\} \quad (1)$$

遵循上面的表述, 我们在后续小节中详细阐述了在 RSM2.0 方案中存在的两种类型泄露指纹, 即掩码指纹以及随机偏移指纹。通过对上述两种泄露指纹的利用以及攻击优化, 攻击者能够以极少的时间开销以及很高的恢复准确率识别出 RSM2.0 中的所有掩码以及索引信息, 从而致使 RSM2.0 防御方案所依赖的随机掩码保护机制全面失效。

4.1.2 掩码指纹

作为 ATmega163 单片机上的软件实现, RSM2.0 防御方案为攻击者提供了大量监测智能卡运行过程内部状态的机会。这种情况主要是由于汇编代码在 CPU 中按照逻辑顺序逐条执行, 因此在执行过程中智能卡芯片会持续不断地向外泄露与指令操作数相关的侧信道信息。另一方面, 大量的前期研究表明, 汉明重量模型(Hamming Weight Model, HW)是一种有效的预测智能卡真实能量泄露 L 的模型, 因此下文中我们将 HW 模型作为上一小节中定义的能量泄露预测模型 W 的一种有效预测。

RSM2.0 方案中存在可利用的掩码指纹泄露主要是由于两方面的原因, 即相邻轮间掩码的使用方案以及固定掩码值的选取上均存在缺陷。具体来说, 作为一个固定且公开的集合, 掩码数组里的元素会根据每次加密时更新的偏移数组 $O()$ 而被随机选取使用。然而, 相邻轮间所使用的输入掩码状态矩阵 $Mask_r$ 存在确定的递推关系。具体来说, 在连续的相邻加密轮中, 输入掩码状态矩阵任意字节位置上使用的掩码按照在固定掩码表中索引位置依次加 1 的顺序进行, 如图 3 所示(图中 $M[i]$ 简记为 M_i):

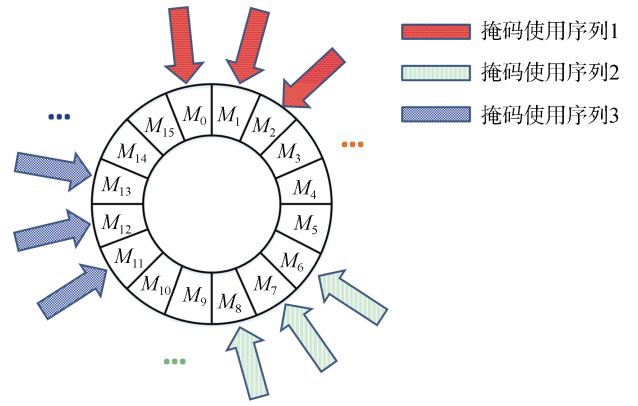


图 3 起点随机的轮间掩码使用序列

Figure 3 Mask usage sequence with random start

以第 i 个状态字节位置为例, 在 RSM2.0 的十轮加密过程中, 一段最大长度为 11 的泄露指纹 $T_{fp(M[O(i)])}$ 可以从单条能量曲线中被提取出来, 并可以表示成如下的形式:

$$L \circ Seq\{M[O(i) + r] | r \in [0, 10]\} + Seq\{\varepsilon_r | r \in [0, 10]\}$$

其中, $M[O(i) + 0]$ 首先被加载进寄存器用以对首轮轮密钥的相应字节进行随机化保护, 而 $M[O(i) + r], r \in [1, 10]$ 则在每轮的掩码补偿阶段被依次加载, 以此来构造当前轮的补偿掩码状态矩阵 $MCP_r, r \in [0, 9]$ 。

而使得这一掩码指纹序列进一步可利用的另一个关键因素在于固定掩码表中的不同元素的汉明重量值不尽相同, 且不存在明显的周期性规律。这一特性使得所有可能的猜测指纹候选序列两两间并不相同。换言之, 任意两个候选序列之间存在可区分性。下面的表格中列出了 $M[O(i) + 0]$ 的 16 种可能取值, 以及他们的相应 HW 值。

表 1 固定掩码数组汉明重量对照表

Table 1 Hamming weight mapping of fixed mask array

$M[]$	0x03	0x0c	0x35	0x3a	0x50	0x5f
$HW \circ M[]$	2	2	4	4	2	6
$M[]$	0x66	0x69	0x96	0x99	0xa0	0xaf
$HW \circ M[]$	4	4	4	4	2	6
$M[]$	0xc5	0xca	0xf3	0xfc		
$HW \circ M[]$	4	4	6	6		

基于表 1, 以 $M[O(i) + 0]$ 为起始位置的 16 种可能的猜测指纹序列可以被攻击者预先计算出来。这些猜测指纹的表示形式为:

$$Seq\{HW \circ M[O(i) + r] | r \in [0, 10]\}$$

其中 $M[O(i) + r], r \in [0, 10]$ 元素在固定掩码数组 $M[]$

中按照从左至右的顺序循环使用各个固定掩码元素。在下表中我们按照最长子序列匹配的原则列举出所有这些猜测指纹序列, 显然两个最长的不可区分序列以 $M[2]$ 和 $M[8]$ 为序列起始点并且包含 6 个元素的序列长度(如表中红色阴影部分所示)。

表 2 16 个掩码指纹猜测序列

$r =$	0	1	2	3	4	5	6	7	8	9	10
$G_{fp}(M[0])$	2	2	4	4	2	6	4	4	4	4	2
$G_{fp}(M[1])$	2	4	4	2	6	4	4	4	4	2	6
$G_{fp}(M[4])$	2	6	4	4	4	4	2	6	4	4	6
$G_{fp}(M[10])$	2	6	4	4	6	6	2	2	4	4	2
$G_{fp}(M[3])$	4	2	6	4	4	4	4	2	6	4	4
$G_{fp}(M[9])$	4	2	6	4	4	6	6	2	2	4	4
$G_{fp}(M[2])$	4	4	2	6	4	4	4	4	2	6	4
$G_{fp}(M[8])$	4	4	2	6	4	4	6	6	2	2	4
$G_{fp}(M[7])$	4	4	4	2	6	4	4	6	6	2	2
$G_{fp}(M[6])$	4	4	4	4	2	6	4	4	6	6	2
$G_{fp}(M[12])$	4	4	6	6	2	2	4	4	2	6	4
$G_{fp}(M[13])$	4	6	6	2	2	4	4	2	6	4	4
$G_{fp}(M[15])$	6	2	2	4	4	2	6	4	4	4	4
$G_{fp}(M[5])$	6	4	4	4	4	2	6	4	4	6	6
$G_{fp}(M[11])$	6	4	4	6	6	2	2	4	4	2	6
$G_{fp}(M[14])$	6	6	2	2	4	4	2	6	4	4	4

为了在恢复随机值 $M[O(i)+0], i \in [0, 15]$ 的过程中获得较高的准确率, 我们将公式(1)中提出的敏感信息识别过程与 Chari 等^[31]学者在 CHES'02 上提出的经典模板攻击方法结合使用。

实际上, 模板攻击方案通过在建模阶段对密码算法计算中间值所对应的真实能量消耗进行分类刻画, 在模板匹配阶段采用极大似然匹配准则来进行正确中间值区分等方式来完成实际的攻击过程, 因此模板攻击具备将真实泄露指纹 T_{fp} 映射为其对应的汉明重量指纹 M_{fp} 的能力。这样, 敏感信息的识别过程就能够被进一步的细化并定义为:

$$M[O(i)] = \underset{M[J]}{\operatorname{argmin}} \{ \operatorname{DIST}(M_{fp}(M[O(i)]), G_{fp}(M[J])) \}$$

通过上述改进的识别方式, 当在每个泄露位置采用模板攻击方法恢复其所对应中间值的汉明重量时, 如果 HW 值的恢复能够达到 100% 的准确率, 则使用长度大于 6 的掩码泄露指纹就足以恢复攻击者所需的随机掩码值, 即 $M[O(i)+0]$ 。鉴于上文中我们已经提取出了最大长度为 11 个元素的掩码泄露指纹 $T_{fp}(M[O(i)])$, 因此攻击者能够使用这种方式恢复出所有 16 个随机掩码值 $M[O(i)+0], i \in [0, 15]$ 。同样值得注意的是, 在这种攻击场景下, 用来进行 $M[O(i)+0]$ 正误判决的区分器 **DIST** 仅需要使用简

单的字符串匹配函数即可满足要求。

然而在实际的攻击场景中, 由于环境噪声与设备电子噪声等因素的影响, 将 T_{fp} 映射为 M_{fp} 的过程不可避免的存在恢复偏差。

以 $G_{fp}(M[2])$ 和 $G_{fp}(M[8])$ 为例, 在 $G_{fp}(M[2])$ 指纹第 6, 7, 9 轮泄露位置的恢复错误可能会将原先对应于 $M[2]$ 的掩码指纹识别为是对应于 $M[8]$ 的指纹。更糟糕的情况在于, 由于恢复偏差的存在, 提取出的泄露指纹可能被映射为一条中间状态的汉明重量序列, 即该序列与猜测指纹序列中穷举出的全部 16 条序列均不相同。在这种情况下, 基于字符串匹配函数的区分器 **DIST** 不再有效, 而需要一种包含错误容忍能力以及错误修正能力的区分器来取代它。

在引入新区分器之前, 我们先对真实噪声环境下单个泄露位置的汉明重量恢复情况进行简单的实验与统计分析。我们的实验数据源自 DPA Contest V4.2 组委会提供的能量曲线集合中随机挑选的 5000 条曲线, 并提取了它们对应的明密文、密钥和所有随机数信息。图 4 和图 5 分别展示了计算中间值 $M[O(3)+5]$ 在不同汉明重量取值下的均值能量曲线, 以及在随机值 $O(3)$ 已知的情况下, 经典模板攻击方法对于 $HW(M[O(3)+5])$ 的恢复准确率图表。

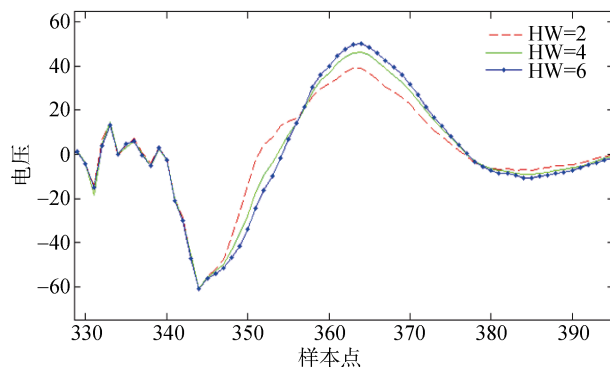


图 4 不同汉明重量等级的均值能量曲线

Figure 4 Mean traces of different HW

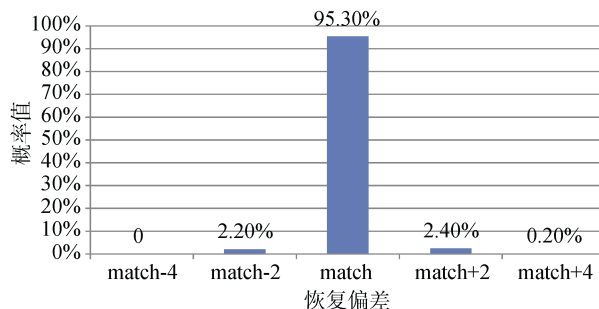


图 5 单点泄露位置恢复偏差

Figure 5 Single position deviation

从图中的实验结果可知:

1. 均值曲线中能量泄露的变化趋势与被处理中间值汉明重量的变化趋势保持一致(正相关或者负相关), 而这也从侧面证实了我们对于 ATmega163 软件平台泄露类型假设的正确性。

2. 从图 5 中可以看出, 针对 $M[O(3)+5]$ 中间值的恢复准确率分布高度集中。即无偏差恢复结果在 5 种可能的恢复偏差分类中成功率最高, 成功率达到 95.3%。同时, 正负偏差在一个间隔单位内的累计恢复成功率则高达 99.8%。

基于上述观察结论, 我们提出使用欧式距离模型来替代无噪声环境下使用的字符串匹配函数, 作为新的泄露指纹识别区分器。这样, 新的泄露指纹识

别过程又可以被进一步的改写为:

$$M[O(i)] = \underset{M[j]}{\operatorname{argmin}} \{ \operatorname{Euclid}(M_{fp(M[O(i)])}, G_{fp(M[j])}) \}$$

直观上看, 欧式距离区分器能够有效降低各个泄露位置的恢复偏差所导致的整体泄露指纹识别错误, 因此, 该区分器具备错误容忍与错误修正的能力。

更进一步, 为了评估在不同的指纹长度与恢复偏差场景下指纹识别框架的识别性能, 我们首先为单个泄露位置的恢复偏差构造了一个粗略的概率模型, 记为 f_p 。更为准确的说, 我们将 $X \in \{2, 4, 6\}$ 定义为对应于某一泄露位置的正确汉明重量值, 并且定义 $Y \in \{2, 4, 6\}$ 为基于该位置泄露而恢复得到的汉明重量结果, 这样 f_p 可以如公式(2)中一样定义, 其中 δ 指的是狄拉克函数。

$$f_p(Y|X) = \begin{cases} (1+p)/2 * \delta(Y-X) + (1-p)/2 * \delta(Y-X-2) & X=2 \\ p * \delta(Y-X) + (1-p)/2 * \delta(Y-X-2) + (1-p)/2 * \delta(Y-X+2) & X=4 \\ (1+p)/2 * \delta(Y-X) + (1-p)/2 * \delta(Y-X+2) & X=6 \end{cases} \quad (2)$$

/*MERGEFORMAT

其中, 以 $p=0.65$ 的偏差模型为例, 相应的偏差模型离散概率分布如图 6 所示。需要特别声明的一点是, 当正确汉明重量变量 X 的取值为其值域范围内的最大或最小元素时(在掩码中这两个值为 2 和 6), 这时的无偏恢复概率值被设计为是 p 与 $(1-p)/2$ 的累积

和。这种构造设计理应是合理的, 原因在于 ATmega163 平台遵循一种线性的能量泄露模式, 因此超出取值范围的恢复偏差在实际模板恢复过程中会被识别为取值范围之内最接近该偏差值的那个元素。

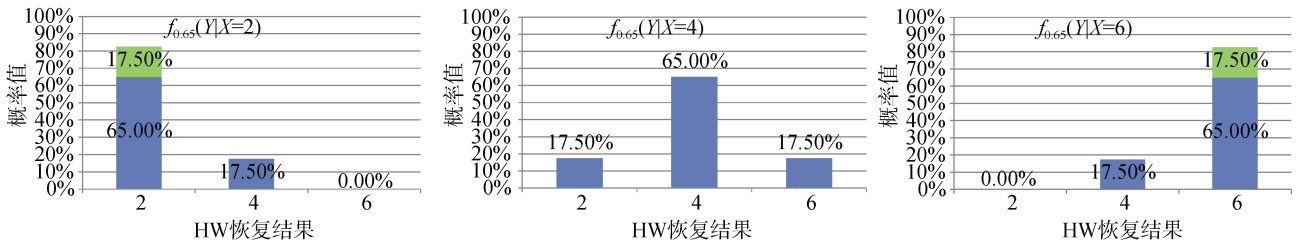


图 6 $p=0.65$ 下的随机掩码恢复偏差模型

Figure 6 Deviation model for HW recovery where $p = 0.65$

基于该恢复偏差概率模型, 我们在三种不同的中心偏差恢复概率下进行了评估实验。这些实验使用 DPA Contest 官方组委会提供的 1000 条随机能量曲线, 并利用泄露指纹识别技术对有偏的汉明重量指纹序列 $M_{fp(M[O(3)+0])}$ 进行识别, 以此来评估在不同的噪声环境下 $M[O(3)+0]$ 值的识别准确率。具体来说, 我们在每个实验中对长度范围从 6 个元素到 11 个元素, 噪声影响下无偏恢复概率 p 从 0.65 到 0.35 范围内的 $M[O(3)+0]$ 无偏识别准确率进行了评估, 实验结果数据记录在表 3 中:

表 3 清晰的显示了泄露指纹识别框架在进行掩码指纹识别时的有效性。此外, 我们所提出的欧氏距

离模型又进一步提升了整个泄露指纹框架的准确性和有效性。即使是在较低信噪比的攻击场景下, 这种识别框架同样能够以较高的概率恢复出与泄露指纹相关的敏感信息, 因此对于所有包含泄露指纹的算

表 3 不同指纹长度与不同偏差模型 $f_p(Y|X)$ 下的 11 随机掩码识别准确率

Table 3 Mask identification accuracy under different length of fingerprint and deviation model $f_p(Y|X)$.

$p \setminus l$	6	7	8	9	10	11
0.65	72.7%	83.5%	86.3%	88.0%	94.3%	95.9%
0.50	50.3%	59.3%	65.2%	68.7%	75.5%	82.1%
0.35	34.0%	43.2%	47.8%	55.7%	59.6%	68.4%

法实现来说, 该识别技术构成了一个简单易用的, 高风险侧信道安全漏洞。

为了抵抗这种类型的攻击策略, 一种简单有效的防御对策是重新选择固定掩码数组 $M[]$ 中的元素, 使得新选择的 16 个元素的汉明重量彼此相同。如此, 当攻击者再次穷举猜测指纹序列时, 所有序列完全相同无法区分, 因此无法用来进行泄露源识别。但是, 正如文献[17]中所述的那样, 为了最小化单变量中间值与实际能量消耗之间的互信息泄露并让 RSM 系列方案具备 4 阶以下安全性, 掩码元素的选择存在限制条件, 因此上诉防御对策的实现在可行性上会受到制约。

4.1.3 随机偏移索引指纹

与掩码指纹可以通过均衡猜测指纹候选集合的方式来进行削弱或者消除不同, 随机偏移索引指纹(后文简称为索引指纹)更容易被设计者忽视, 同时也更加难以进行实际的防护^①。

索引指纹的产生同样源自 RSM2.0 方案设计中对于固定掩码元素的循环利用规律。正如 4.1.2 中阐述的一样, 在 10 轮范围内的掩码使用构成了掩码指纹。然而在掩码指纹产生之前, 16 个掩码索引值 $[O(i)+r]\%16$, $i \in [0,15]$ 被分别单独计算并加载进寄存器中, 其中 $r \in [0,10]$ 。事实上, 在 RSM2.0 方案的实现代码中, 表 4 的三条指令按顺序相继执行, 从而计算出对应于每个掩码值的最终索引:

表 4 汇编代码中存在敏感信息泄露的指令

Table 4 Sensitive instructions in assembly code

汇编代码中执行的相关指令 (1071-1073 行)	寄存器中的对应存储值
ld H1,X+	$O(i)$
add YL,H1	$O(i)+r$
andi YL, 0x0F	$[O(i)+r]\%16$

上面表 4 中所列出的这些汇编指令代码分别以 $L \circ O(i) + \varepsilon$, $L \circ (O(i) + r) + \varepsilon$, $L \circ ([O(i) + r]\%16) + \varepsilon$ 的形式泄露能量。本节中我们主要关注后两个指令的泄露, 且 10 轮加密过程中这两者的所有泄露信息构成了如下两种索引指纹形式:

1. $T_{fp(O(i)+0)} = L \circ Seq\{[O(i)+r] \mid r \in [0,10]\}$
 $+Seq\{\varepsilon_r \mid r \in [0,10]\}$
2. $T_{fp([O(i)+0]\%16)} = L \circ Seq\{[O(i)+r]\%16 \mid r \in [0,10]\}$
 $+Seq\{\varepsilon_r \mid r \in [0,10]\}$

与 4.1.2 小节中提到的一样, 使得这些索引指纹可利用的主要原因在于在汉明重量模型假设下, 不同的猜测索引序列间可区分。作为 RSM2.0 中的掩码索引值, $O(i)$ 值取遍 0 到 15 的整数。这里将该取值范围内的元素用数组 $\theta[]$ 表示, 其中 $\theta[i]=i$, $i \in [0,15]$ 。下面的表 5 列出了 $\theta[]$ 数组中每个元素对应的汉明重量值。

表 5 随机索引取值范围汉明重量对照表

Table 5 Hamming weight mapping of random index

$\theta[]$	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
HW $\circ\theta[]$	0	1	1	2	1	2	2	3
$\theta[]$	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
HW $\circ\theta[]$	1	2	2	3	2	3	3	4

基于上述 θ 元素汉明重量对照表, 两个以 $[\theta(i)+0]$ 以及 $[\theta(i)+0]\%16$ 为起始点的索引指纹猜测序列集合可以分别由下列公式形式表示, 其中每个集合里包含 16 种可能的猜测索引序列且 $i \in [0,15]$:

$$G_{fp(\theta(i))} = Seq\{HW^\circ[\theta(i)+r] \mid r \in [0,10]\}$$

$$G_{fp(\theta[i]\%16)} = Seq\{HW^\circ[\theta(i)+r]\%16 \mid r \in [0,10]\}$$

下面我们索引指纹 $G_{fp(\theta[i]\%16)}$ 为例, 按照最长子序列匹配的序列列出了其所对应的所有猜测指纹候选序列。显然, 两个最长的不可区分序列分别以 $[\theta[4]\%16]$ 以及 $[\theta[8]\%16]$ 为起始点, 并且不可区分序列的总长度为 4 个元素。

表 6 随机偏移识别中的 16 个猜测索引序列

Table 6 16 guessing sequences of offset fingerprint

$r =$	0	1	2	3	4	5	6	7	8	9	10
$G_{fp(\theta[0]\%16)}$	0	1	1	2	1	2	2	3	1	2	2
$G_{fp(\theta[1]\%16)}$	1	1	2	1	2	2	3	1	2	2	3
$G_{fp(\theta[2]\%16)}$	1	2	1	2	2	3	1	2	2	3	2
$G_{fp(\theta[4]\%16)}$	1	2	2	3	1	2	2	3	2	3	3
$G_{fp(\theta[8]\%16)}$	1	2	2	3	2	3	3	4	0	1	1
$G_{fp(\theta[3]\%16)}$	2	1	2	2	3	1	2	2	3	2	3
$G_{fp(\theta[5]\%16)}$	2	2	3	1	2	2	3	2	3	3	4
$G_{fp(\theta[9]\%16)}$	2	2	3	2	3	3	4	0	1	1	2
$G_{fp(\theta[6]\%16)}$	2	3	1	2	2	3	2	3	3	4	0
$G_{fp(\theta[10]\%16)}$	2	3	2	3	3	4	0	1	1	2	1
$G_{fp(\theta[12]\%16)}$	2	3	3	4	0	1	1	2	1	2	2
$G_{fp(\theta[7]\%16)}$	3	1	2	2	3	2	3	3	4	0	1
$G_{fp(\theta[11]\%16)}$	3	2	3	3	4	0	1	1	2	1	2
$G_{fp(\theta[13]\%16)}$	3	3	4	0	1	1	2	1	2	2	3
$G_{fp(\theta[14]\%16)}$	3	4	0	1	1	2	1	2	2	3	1
$G_{fp(\theta[15]\%16)}$	4	0	1	1	2	1	2	2	3	1	2

① 为了实际表述的需要, “mod 16” 运算操作将在本小节与下一小节中明确标出

当攻击者计算出所有的猜测索引序列并且获得真实泄露指纹序列后, 对于随机索引 $[O(i)+0]$ 的识别过程分为以下两个步骤:

- 步骤 1: 将随机索引指纹 $T_{fp([O(i)+0])}$ 或者 $T_{fp([O(i)+0]\%16)}$ 利用模板攻击方法映射为汉明重量指纹序列 $M_{fp([O(i)+0])}$ 或者 $M_{fp([O(i)+0]\%16)}$ 。
- 步骤 2: 将计算出的 16 个猜测指纹序列与 $M_{fp([O(i)+0])}$ 或 $M_{fp([O(i)+0]\%16)}$ 序列进行比较, 并且使用下面的判决准则来进行敏感信息 $[O(i)+0]$ 的识别:

$$O(i) = \underset{\theta[j]}{\operatorname{argmin}} \operatorname{Euclid}(M_{fp(O(i))}, G_{fp(\theta[j])})$$

$$O(i) = \underset{\theta[j]}{\operatorname{argmin}} \operatorname{Euclid}(M_{fp([O(i)+0]\%16)}, G_{fp([\theta[j]+0]\%16)})$$

这样一来, 在中间值汉明重量的恢复准确率能够达到 100% 的前提下, 一条长度大于 4 的真实指纹序列就足以用来进行敏感索引值识别, 即识别

$$f_p(Y|X) = \begin{cases} (1+p)/2 * \delta(Y-X) + (1-p)/2 * \delta(Y-X-1) & X=0 \\ p * \delta(Y-X) + (1-p)/2 * \delta(Y-X-1) + (1-p)/2 * \delta(Y-X+1) & X=1,2,3 \\ (1+p)/2 * \delta(Y-X) + (1-p)/2 * \delta(Y-X+1) & X=4 \end{cases} \quad (3)$$

/*MERGEFORMAT

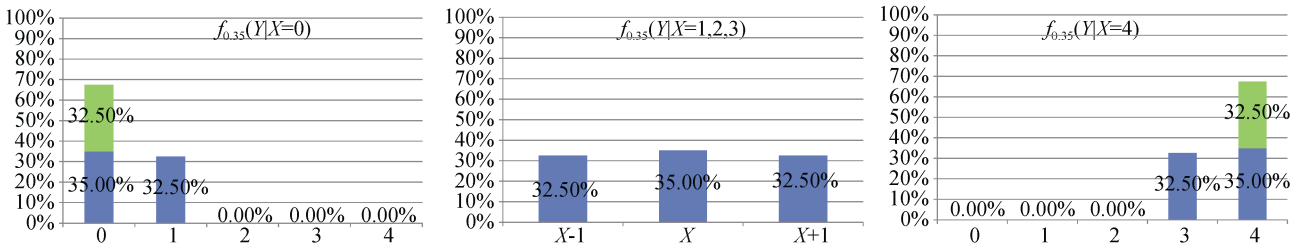


图 7 $p = 0.35$ 下的随机偏移索引恢复偏差模型

Figure 7 Deviation model for HW recovery where $p = 0.35$

这里需要特别强调的是, 当随机变量 X 恰好取到偏移索引取值范围内的最小值或者最大值时(这里分别指的是 0 和 4 这两个值), 这时候的无偏猜测概率被设计为是 p 和 $(1-p)/2$ 两个值的累加值。这样的概率模型构造同样是合理的, 原因已在 4.1.2 小节中针对掩码值构造的偏差概率模型中进行了阐述。最后, 三组对比实验的最终结果记录在表 7 中。

从大体上来看, 表中体现出的变化趋势与掩码指纹识别中的结果大致相同。也就是说, 随着单点泄露位置汉明重量恢复成功率的上升以及泄露指纹长度的增加, 利用泄露指纹框架对随机索引值进行识别的准确率就越高。除此之外, 对比表 3 可知在同样的中心偏差概率与同样的指纹长度下, 随机索引的识别相比随机掩码的识别准确率更高。这种情况

$O(i)+0, i \in [0,15]$ 。又因为我们能够从 RSM2.0 算法中提取出长度为 11 个位置的真实泄露指纹, 因此我们所构造出来的两种随机索引泄露指纹能够用来进行敏感信息识别。

为了进一步验证我们所提出的索引指纹识别框架的健壮性, 我们在不同水平的噪声环境下进行了三组对比实验。每组实验中, 我们同样对不同的指纹长度给识别准确率带来的影响进行了评估。

与 4.1.2 小节中一样, 我们首先对单点泄露位置的汉明重量恢复偏差构建概率模型 f_p 。更为准确的说, 我们将变量 $X \in \{0,1,2,3,4\}$ 定义为对应于某个确定泄露位置的正确汉明重量值, 并且将 $Y \in \{0,1,2,3,4\}$ 定义为基于该位置泄露恢复得到的猜测结果值, 这样 f_p 可以由公式(3)进行定义, 其中 δ 表示的是狄拉克函数。

其中, 中心恢复偏差为 0.35 的概率分布模型如图 7 所示:

主要是由于随机索引变量中更大的取值范围以及索引指纹中更短的可区分指纹长度所导致的。

表 7 不同指纹长度与不同偏差模型 $f_p(Y|X)$ 下的随机索引识别准确率

Table 7 Offset identification accuracy under different length of fingerprint and deviation model $f_p(Y|X)$

$p \backslash l$	6	7	8	9	10	11
0.65	89.9%	91.6%	94.4%	96.8%	97.1%	98.2%
0.50	75.2%	84.5%	88.5%	93.7%	95.0%	97.7%
0.35	60.5%	72.1%	77.5%	86.2%	89.8%	93.4%

更为重要的是, 正如表 7 中最后一列所示, 当指纹长度达到最长的 11 个元素时, 单点泄露位置的恢复偏差对于最终的指纹识别准确率影响较小。即

使是在 $p=0.35$ 这样的强噪声环境下, 随机索引的识别准确率依然能够高达 93.4%。上述的实验数据表明, 泄露指纹恢复方案在针对随机索引的识别过程中不但有效而且具备很强的鲁棒性。

4.2 资源开销与泄露指纹优化

本节主要从泄露指纹利用技术在实际攻击中的计算以及存储资源开销角度进行分析, 对该项技术的优化方案展开讨论, 并提出了相关的评价指标对不同优化策略下的方案性能进行量化评估。

4.2.1 掩码指纹优化

从攻击者的角度来看, 尽管利用更多位置的能量泄露可以更为精确地对掩码泄露源进行恢复, 但更长的泄露指纹同时也意味着计算资源和存储开销的显著增大。实际上, 在掩码指纹的实际利用过程中, 攻击者需要首先对每个指纹泄露位置分别进行汉明重量建模, 并将构建好的各个泄露位置的汉明重量模板提前存储在模板数据库中。随后在模板匹配阶段, 攻击者将新获取的能量泄露与先前构建好的不同等级的汉明重量进行逐一匹配以便获得与当前能量泄露最相近的汉明重量等级。因此, 在资源极端受限的攻击场景中, 在保证掩码指纹猜测序列可区分的前提下尽可能的缩短泄露指纹的长度就成为这类攻击者的核心关注点。

针对上述泄露指纹攻击过程的资源开销优化问题, 我们利用表 2 对 11 个掩码指纹泄露位置中的任意 n 个位置的所有组合情况进行穷举搜索, 其中 $n \in [1, 11]$ 。穷举搜索的实验数据如表 8 所示, 实验结果表明, 在保证 16 个掩码指纹猜测序列可区分的前提下, 最短的可利用泄露指纹长度为 4, 且该长度下的可区分泄露位置组合并不唯一。当利用表 8 中找到的两种长度为 4 的泄露位置集合展开泄露指纹的攻击时, 相比于从起始位置 $M[O(i)+0]$ 开始的长度为 7 的最短可区分泄露指纹, 攻击者的计算开销和存储开销可以分别降低约 42.86%。

表 8 掩码指纹最短可区分位置搜索		
Table 8 Shortest distinguishable position search in mask fingerprint		
Len	长泄露位置 组合总数	可区分的泄露位置 组合
Len=1	11	无
Len=2	55	无
Len=3	165	无
Len=4	330	{0,4,5,9} 以及 {1,5,6,10}
Len=5	462	{0,1,2,4,6} 等 138 种 位置组合

表 9 以长度为 4 的最短泄露位置组合之一的 {0,4,5,9} 为例, 同样按照最长子序列匹配的原则列举出了该位置组合下的 16 个掩码指纹猜测序列。从该表中可以看出不存在两个完全相同的猜测指纹序列, 因此长度为 4 的最短可利用泄露指纹是真实存在的, 这也进一步验证了表 3 中所得到的泄露位置搜索结果的正确性与有效性。

表 9 泄露位置组合 {0,4,5,9} 下的 16 个掩码指纹猜测序列				
Table 9 16 guessing sequences for mask fingerprint within index positions {0,4,5,9}				
泄露位置 猜测序列	0	4	5	9
$G_{fp(M[0])}$	2	2	6	4
$G_{fp(M[4])}$	2	4	4	4
$G_{fp(M[1])}$	2	6	4	2
$G_{fp(M[10])}$	2	6	6	4
$G_{fp(M[12])}$	4	2	2	6
$G_{fp(M[13])}$	4	2	4	4
$G_{fp(M[6])}$	4	2	6	6
$G_{fp(M[8])}$	4	4	4	2
$G_{fp(M[3])}$	4	4	4	4
$G_{fp(M[2])}$	4	4	4	6
$G_{fp(M[9])}$	4	4	6	4
$G_{fp(M[7])}$	4	6	4	2
$G_{fp(M[15])}$	6	4	2	4
$G_{fp(M[5])}$	6	4	2	6
$G_{fp(M[14])}$	6	4	4	4
$G_{fp(M[11])}$	6	6	2	2

当然, 在实际的攻击场景中, 相比于上面讨论的搜索最短可区分能量泄露位置方案, 一种更为实际的攻击策略是在泄露指纹的长度选取与攻击者所拥有的计算资源以及可接受的计算效率之间寻找一个最佳的平衡点。

然而, 从表 8 中我们看出, 当攻击者根据可接受的资源开销确定了泄露指纹长度之后, 攻击者仍然存在多种可能的位置组合选择。例如, 当选择长度为 5 的泄露指纹展开攻击时, 存在 462 种可能的泄露位置选取方法。因此, 如何从所有这些可能的泄露位置候选集合中选出能够最大化攻击效率的泄露位置组合就成为了攻击者在实际攻击中最关心的问题。

鉴于上述问题, 本文首次提出了“最邻近指纹距离平均值(Mean of Nearest Distance, 下文简称为 MOND)”的评价指标来衡量选定泄露指纹长度条件下, 选取不同泄露位置开展实际攻击的难易程度。鉴

于攻击者在实际的敏感信息识别过程中更可能在差异最小的两个猜测指纹序列间出现敏感信息识别错误, 因此统计所有猜测序列与其差异最小化的猜测序列间的距离值, 并计算出其平均值即可在一定程度上反映出攻击过程中出现敏感信息识别错误的难易程度。具体来说, MOND 指标的计算过程伪代码如算法 2 所示:

算法 2. 掩码指纹 MOND 参数计算

输入:

待评估的指纹长度: N ;

待评估的泄露位置: $\{x_1, x_2, \dots, x_N\}$;

输出:

当前条件下掩码指纹的 MOND 指标;

//计算每个猜测指纹序列的最邻近距离

1: $Sum = 0$;

2: FOR {each $i \in [0, 15]$ }

3: $Min = Maximum$;

4: FOR {each $j \in [0, 15]$ and $i \neq j$ }

//计算两个指纹序列在 N 个泄露位置的曼哈顿距

离

5: $Temp = Manhattan_Dist(G_{fp(M[i])}, G_{fp(M[j])})$

6: IF ($Temp < Min$)

7: $Min = Temp$;

8: ENDIF

9: ENDFOR

10: $Sum + = Min$;

11: ENDFOR

//计算最邻近距离平均值

12: $MOND = Sum * 1.0 / 16$;

13: RETURN $MOND$;

从算法 2 的计算过程以及 MOND 指标的定义可知, MOND 值越大说明攻击者出现敏感信息识别错误的可能性越小, 因此其所对应的泄露位置 $\{x_1, x_2, \dots, x_N\}$ 相比于其他 MOND 较小的泄露位置将是更为有效的攻击位置选择。利用这一评价指标及其计算方法, 我们对指纹长度从 1 到 11 的所有可区分泄露位置进行了 MOND 参数计算, 并按照 MOND 参数值递减的顺序对他们进行了排序。最终, 在每个指纹长度分组中, MOND 极大值所对应的可区分泄露位置如表 10 所示:

至此, 攻击者在实际执行泄露指纹识别的过程中首先可以根据自身拥有的计算资源与存储资源情况来合理的选择后续使用的泄露指纹长度。随后, 根据上述对照表, 攻击者利用该指纹长度下 MOND 极

大值所对应的可区分泄露位置来展开实际的攻击过程, 进而能够最优化当前指纹长度条件下敏感信息的识别准确率。

表 10 MOND 极大值与对应掩码泄露位置对照表
Table 10 Maximum of MOND and the corresponding index positions of mask leakage

	可区分的泄露位置总数	MOND 极大值及对应的可区分泄露位置
Len=1,2,3	0	无
Len=4	2	2.625 {0,4,5,9} 2.625 {1,5,6,10}
Len=5	138	3.75 {0,4,6,7,10} 3.75 {0,3,4,6,10}
Len=6	299	4.875 {0,1,2,6,9,10} 4.875 {0,1,4,8,9,10}
Len=7	290	6.125 {0,1,2,5,6,9,10} 6.125 {0,1,4,5,8,9,10}
Len=8	161	6.75 {0,1,2,4,6,8,9,10}
Len=9	55	7.5 {0,1,2,3,4,7,8,9,10} 7.5 {0,1,2,3,6,7,8,9,10} 7.5 {0,1,2,4,5,6,8,9,10}
Len=10	11	8.25 {0,1,2,3,4,6,7,8,9,10}
Len=11	1	9 {0,1,2,3,4,5,6,7,8,9,10}

4.2.2 随机偏移指纹优化

与掩码泄露指纹优化技术中讨论的情况相同, 下文首先分析了在资源极端受限的条件下展开攻击时, 攻击者所能够利用的最短偏移指纹长度, 以及该最短长度下筛选出的可区分泄露位置。

我们利用表 6 对 11 个偏移指纹泄露位置中的任意 n 个位置的所有组合情况进行穷举搜索, 其中 $n \in [1, 11]$, 穷举搜索的实验数据如表 11 所示。实验结果表明, 在保证 16 个偏移指纹猜测序列两两可区分的前提下, 最短的可区分偏移指纹长度为 3, 且该长度下的可区分泄露位置组合共有 36 种可能的取值情况。

表 11 最短可区分偏移指纹泄露位置搜索
Table 11 Shortest distinguishable index positions search for offset fingerprint

	Len 长泄露位置组合总数	可区分泄露位置组合
Len=1	11	无
Len=2	55	无
Len=3	165	{0,1,6}, {0,1,7} 等 36 种
Len=4	330	{6,8,9,10} 等 296 种

对比以 $(O[0]+0)\%16$ 为起始位置的最短长度为 5 的连续位置偏移指纹, 表 11 中搜索得到的最短长度为 3 的泄露位置可以将攻击者的计算开销和存储开销降低约 40.0%。为了验证搜索结果的正确性和有效性, 表 12 按照最长子序列匹配的原则列出了所提取的 $\{0,1,7\}$ 泄露位置的 16 种可能的偏移指纹猜测序列。从表中可以看出, 其中的任意两个猜测序列之间至少在一个泄露位置上存在差异, 因此这组位置上的偏移指纹能够用来进行随机偏移值的恢复。

表 12 泄露位置组合 $\{0,1,7\}$ 下的 16 个偏移指纹猜测序列

Table 12 16 guessing sequences for offset fingerprint within index positions $\{0,1,7\}$

$r =$	0	1	7
$G_{fp}(O[0]\%16)$	0	1	3
$G_{fp}(O[1]\%16)$	1	1	1
$G_{fp}(O[2]\%16)$	1	2	2
$G_{fp}(O[4]\%16)$	1	2	3
$G_{fp}(O[8]\%16)$	1	2	4
$G_{fp}(O[3]\%16)$	2	1	2
$G_{fp}(O[9]\%16)$	2	2	0
$G_{fp}(O[5]\%16)$	2	2	2
$G_{fp}(O[10]\%16)$	2	3	1
$G_{fp}(O[12]\%16)$	2	3	2
$G_{fp}(O[6]\%16)$	2	3	3
$G_{fp}(O[7]\%16)$	3	1	3
$G_{fp}(O[11]\%16)$	3	2	1
$G_{fp}(O[13]\%16)$	3	3	1
$G_{fp}(O[14]\%16)$	3	4	2
$G_{fp}(O[15]\%16)$	4	0	2

同样的, 当攻击者根据拥有的计算和存储资源开销选定了待分析泄露指纹的长度之后, 攻击者需要进一步在搜索到的所有可区分泄露位置中筛选出能够最大化敏感信息识别准确率的泄露位置。根据上文在掩码泄露指纹优化方案中的结论, 我们同样利用上文中提出的“最邻近指纹距离平均值(MOND 指标)”对长度从 1 到 11 的所有长度分类下的可区分泄露位置组合进行评估, 并记录下其中 MOND 取得极大值时所对应的随机索引泄露位置, 评估结果如表 13 所示。

如此, 攻击者在实际执行随机索引值识别的过程中可以通过权衡自身能够接受的资源开销(包括计算与存储资源等)来合理的选择后续使用的指纹长度。随后, 利用表 13 中记录的 MOND 极大值及其所对应的可区分泄露位置, 攻击者再利用本文中提

出的索引指纹识别技术来开展实际的随机索引恢复, 进而能够最优化随机索引值这一敏感信息的识别准确率。

表 13 MOND 极大值与对应随机索引泄露位置对照表
Table 13 Maximum of MOND and the corresponding index positions of random offset leakage

	可区分泄露位置总数	MOND 极大值及对应的可区分泄露位置
Len=1,2	0	无
Len=3	36	1.625 {0,4,10} 1.625 {0,6,10}
Len=4	296	2.75 {0,4,6,10} 3.5625 {0,1,3,6,10} 3.5625 {0,1,4,6,10}
Len=5	461	3.5625 {0,4,6,9,10} 3.5625 {0,4,7,9,10} 4.5625 {0,1,3,6,7,10}
Len=6	462	4.5625 {0,3,4,7,9,10} 5.3125 {0,1,2,4,6,7,10}
Len=7	330	5.3125 {0,3,4,6,8,9,10} 6.125 {0,1,2,3,4,6,8,10} 6.125 {0,1,2,4,6,7,8,10}
Len=8	165	6.125 {0,1,3,4,6,7,9,10} 6.125 {0,2,3,4,6,8,9,10} 6.125 {0,2,4,6,7,8,9,10}
Len=9	55	7.125 {0,1,3,4,5,6,7,9,10}
Len=10	11	7.9375 {0,1,2,3,4,5,6,7,9,10}
Len=11	1	7.9375 {0,1,3,4,5,6,7,8,9,10}
		8.75 {0,1,2,3,4,5,6,7,8,9,10}

4.3 密钥恢复与官方评估结果

从实际攻击角度来看, 当利用泄露指纹技术破解了随机掩码 $M[O(i)]$ 或随机偏移数组 $O(i)$ 后, 分析人员往往会将第一轮掩码 S 盒的输出值或者是最后一轮掩码 S 盒的输入值选定为目标中间值展开能量分析。然而, 在 RSM2.0 的方案设计中, 这两部分掩码 S 盒能量泄露已由乱序操作进行保护。

尽管如此, 本节中指出, RSM2.0 方案中针对掩码 S 盒的乱序防护在实现层面存在可以被直接绕过的安全漏洞。利用这一绕过漏洞, 攻击者可以在恢复随机掩码的前提下直接进行 RSM2.0 主密钥恢复, 且实施分析的难度与针对无保护 AES-128 算法实现的分析难度并无区别。

4.3.1 乱序防护绕过漏洞

本小节中介绍的用以绕过掩码 S 盒乱序防护的漏洞出现在 RSM2.0 第一轮加密过程的列混淆操作中。实际上, 在 RSM2.0 列混淆计算过程中, 状态矩阵每列四个元素的列混淆变换通过下列公式分别实现, 其中 $i \in [0,3]$ 且值 $(i+1)$ 定义在模 4 意义下, 而 Wi' 表示更新后当前字节位置的列混淆结果:

$$W_i' = (W0 \oplus W1 \oplus W2 \oplus W3) \oplus 2 * (W_i \oplus W_{i+1}) \oplus W_i$$

在生成中间值 $2 * (W_i \oplus W_{i+1})$ 的过程中(如图 8), W_i 按照字节顺序首先被加载进寄存器中, 产生相应能量泄露。而实际上, 作为列混淆层的输入, 每个

W_i 变量实际上即为掩码 S 盒的输出结果。因此, 在随机掩码已经被破解的前提下, 利用该计算位置的能量泄露可以绕开乱序防护直接对 RSM2.0 进行一阶能量分析。

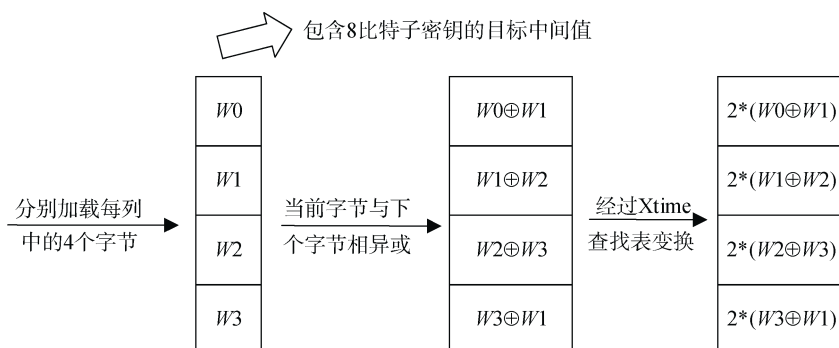


图 8 第一轮中的乱序防御绕过漏洞

Figure 8 Bypassing vulnerability against shuffling defense in the first round

另外值得一提的是, 上面提到的攻击中间值仅包含 8 比特的密钥信息(如图 8 所示), 即攻击者只需要遍历 256 个元素的候选子密钥空间就可以区分出正确的密钥值, 因此攻击者能够高效的恢复所有 16 个主密钥字节。

4.3.2 密钥恢复及官方评估

本小节主要展示利用本章提出的泄露指纹识别技术对 DPA Contest 组委会实现的 RSM2.0 软件方案进行实际破解的实验细节, 并展示了相应的官方评估结果。

官网提供的所有能量曲线均针对智能卡中的 8 比特单片机(ATMega163)进行采集。曲线采集通过力科示波器(型号为 WaveRunner 6100A)配合电磁探头来完成, 示波器的采样率被设置为 500 MS/s, 而采集带宽则被设置为 200MHz。

官方组委会共拥有两套能量曲线集合, 其中一套曲线对所有的参赛团队公开, 供参赛者线下分析使用。而另一套曲线则供组委会对征集到的攻击代码进行性能评估, 且该套能量曲线不对外公开。

公开集合中共包含 16 组能量曲线, 每组能量曲线对应一个随机 128 比特主密钥。每个能量曲线分组中包含 5000 条能量曲线, 而每条曲线则由 1704402 个采样电压值组成, 代表了 RSM2.0 掩码方案在 10 轮加密过程中所产生的电路能量消耗。此外, 与每条能量曲线对应的一些基本参数也提供给了参赛者, 包括: 明密文对, 主密钥, shuffle0 数组, shuffle10 数组, 随机偏移数组等。参赛者可以利用公开的能量曲线集合来对其中的数据进行预处理, 如信噪比评估, 特征点选择, 模板构建等, 以此来为针对

私有能量曲线集合的在线攻击做前期准备。

另一方面, 私有数据集用以对提交代码进行性能评估, 主要评估指标使用 09 年欧密会文章^[32]中提出的成功率(Success Rate, SR)与猜测熵(Guessing Entropy, GE)。根据官网公布的评估参数, 攻击方案最多可以使用 1000 条能量曲线来进行主密钥恢复, 而攻击过程中每条曲线的平均处理时间同样被纳入评估框架, 以此来评判整体方案的计算开销情况。实验平台方面, 所有的提交代码均在相同配置的服务器上执行, 以保证性能评估的公平性。该服务器配备英特尔 Xeon 系列的 E7-8837 型号 CPU, 单核主频为 2.67GHz, 并配有 256GB 的内存空间。

基于在 4.1 以及 4.3.1 小节中描述的 RSM2.0 攻击框架以及绕过漏洞, 下面阐述两种完整的攻击方案, 且这两种方案均已提交并通过了官方组委会的评估。

我们首先将 16 组公开曲线划分为两个子集合(即刻画数据集以及验证数据集), 其中第一个集合用来进行模板刻画, 另一个集合用来进行敏感信息恢复的准确度验证。随后, 通过使用经典的模板攻击方法并结合本章中提出的泄露指纹识别技术, 我们首先对每次加密过程中持续更新的随机偏移数组进行识别。为了验证实际准确率, 我们从验证数据集中随机提取了 1200 条能量曲线, 并将利用掩码指纹与偏移索引指纹分别得到的识别准确率记录在表 14 以及表 15 中。由于在提取出的能量曲线上各偏移字节位置的泄露水平存在较大差异, 将不同字节的识别准确率进行简单平均并不具有实际意义。因此, 表中列举出的是泄露量最小的偏移字节位置的识别准确率情况。

表 14 基于掩码指纹的随机掩码识别准确率
(最小泄露字节)

Table 14 Identification accuracy of mask fingerprint
for the byte of least leakage

Length	1	2	3	4	5	6
Accur.	17.5%	55%	66.8%	80%	84.8%	92.5%
Length	7	8	9	10	11	
Accur.	99.8%	99.8%	100%	100%	100%	

表 15 基于偏移指纹的随机偏移识别准确率
(最小泄露字节)

Table 15 Identification accuracy of offset fingerprint
for the byte of least leakage

Length	1	2	3	4	5	6
Accur.	27%	63.7%	83.2%	87.3%	97%	97.8%
Length	7	8	9	10	11	
Accur.	98.3%	98.8%	98.8%	100%	100%	

上表中的实验数据表明,即使是针对最难恢复的偏移字节位置,泄露指纹框架仍然能够获得 100% 的敏感信息识别准确率。

在随机偏移数组能够被无偏差破解的前提下,我们随后利用 4.3.1 中指出的漏洞绕过受保护的掩码 S 盒操作,而直接利用列混淆输入中间值进行模板攻击以及相关系数攻击。

针对我们提交的两套方案,官方给出了详细评估报告,其中全局成功率 GSR 的评估结果如图 9 及图 11 所示。该结果表明我们提交的两套方案均实际有效。对于其中利用泄露指纹与相关系数攻击相结合的方案,11 条能量曲线就能够以 80% 的全局成功率破解 RSM2.0 防御方案。同时,8 条能量曲线就足以将部分猜测熵 PGE 的最大值降低至 10 以下。对于另一个泄露指纹与模板攻击相结合的方案,其破解效率更为高效,该方案仅需 4 条能量曲线就能够达到 80% 的全局成功率,且仅需 2 条能量曲线就足够满足部分猜测熵最大值低于 10 的评估指标要求。

从代码执行时间的角度来看,两种方案分别需要 200 毫秒以及 100 毫秒的总时间来相继破解随机掩码防御以及后续的主密钥恢复。因此从执行时间角度上看,同样说明了我们提交方案的高效性。

5 非模板类安全漏洞分析

与上一章中阐述的模板类攻击方法不同,本章重点关注针对 RSM2.0 方案的非模板类攻击方法。

所谓的非模板攻击指的是攻击者无法事先获得从待攻击目标设备中提取出的大量能量曲线数据集,

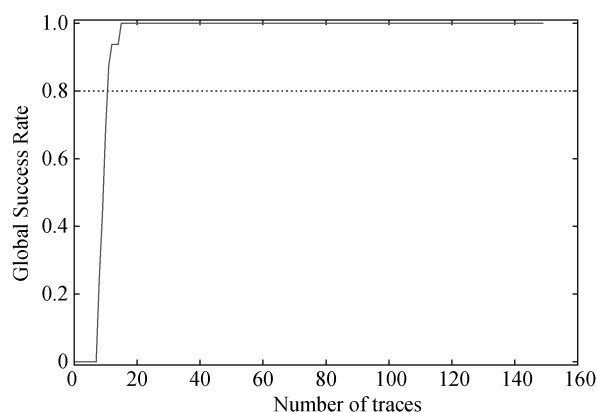


图 9 指纹-CPA 攻击方案的全局成功率评估结果

Figure 9 GSR evaluation for fingerprint-CPA attacking scheme

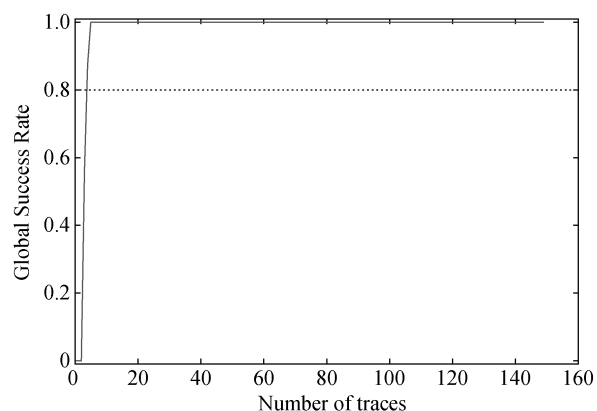


图 10 指纹-TA 攻击方案的全局成功率评估结果

Figure 10 GSR evaluation for fingerprint-TA attacking scheme

从而无法在攻击开始前的阶段对目标设备的真实能量泄漏模型进行精确刻画的一种攻击类型。典型的非模板类型攻击包括 DPA^[8], CPA^[33], MIA^[34]等。

与经典的模板类攻击相比,如标准模板攻击 TA^[31],以及随机模型攻击 SA^[35],非模板类型的攻击通常需要使用更多的能量曲线。但由于非模板攻击是一种对攻击者能力要求更低的假设,因此它往往会造成更大的安全威胁。

5.1 非模板类型攻击漏洞

5.1.1 算法设计层面漏洞

• 不可避免的补偿掩码在线计算过程

基于 3.3 节提出的漏洞检测准则,敏感区域中发现的首个 RSM2.0 非模板漏洞出现在补偿掩码状态 MCP_r 的计算过程中。由于 RSM2.0 升级方案中使用随机偏移数组 OQ 取代了原始方案中的单变量随机索引值,中间值 MCP_r 的派生过程不可避免的需要在线完成。在原始方案中,第 $r+1$ 轮的输入掩码状态

$Mask_{idx,r}$ 唯一的由 4 比特随机索引 idx 决定。更准确

的说, 该输入掩码状态满足下列公式:

$$\begin{pmatrix} M[idx+0+r] & M[idx+4+r] & M[idx+8+r] & M[idx+12+r] \\ M[idx+1+r] & M[idx+5+r] & M[idx+9+r] & M[idx+13+r] \\ M[idx+2+r] & M[idx+6+r] & M[idx+10+r] & M[idx+14+r] \\ M[idx+3+r] & M[idx+7+r] & M[idx+11+r] & M[idx+15+r] \end{pmatrix}$$

由于 $M[]$ 是一个固定的掩码数组, $Mask_{idx,r}$ 状态的所有可能取值只有 16 种。这样, 以 $Mask_{idx,r}$ 为输入值推导生成的补偿掩码状态实际上也只有 16 种可能, 因此该状态矩阵可以通过线下预计算提前求出以便在后续的计算过程中直接使用。然而, 在 RSM2.0 方案中, 输入掩码状态 $Mask_r$ 利用偏移数组 $O()$ 对各字节位置的掩码值进行索引, 数组中的每个索引值 $O(i)$ 独立同分布, 因此导致更新的补偿掩码状态 $Mask_r$ 存在 16^{16} 种候选值。为了存储所有补偿掩码状态的取值, 共需要 16×16^{16} 字节的存储空间。而这一存储开销在资源受限的嵌入式设备中往往无法满足。至此, RSM2.0 的补偿掩码生成过程必须在线生成, 而在执行这一操作过程中产生的所有相关能量泄露都将被记录在能量曲线中。

另一个导致补偿掩码生成过程可利用的重要原因是该阶段的操作不受乱序方案保护。该处防护的缺失意味着在补偿掩码的派生阶段, 包括输入掩码状态 $Mask_r$ 的加载, 补偿掩码第一部分中间值的产生, 完整补偿掩码生成在内的所有处理过程都在固定的时刻点上产生相应的数据依赖性泄露, 图 11 展示了上述的掩码能量泄露。

• 轮密钥加计算过程漏洞

除补偿掩码派生外, RSM2.0 中的轮密钥加操作

同样缺乏乱序保护, 因此导致潜在的二阶攻击威胁。这其中, 由于顺序进行的轮密钥加执行过程而导致的可利用漏洞出现在第九轮末尾。在该位置, 第九轮轮密钥加操作被设计在当前轮的列混淆操作结束之后执行。换言之, 第九轮轮密钥加操作后得到的每个输出字节与该轮中的列混淆输出字节采用相同的掩码进行保护, 如图 12 所示, 其中 X_i 代表第九轮的输入字节而 K_i 以及 K'_i 则分别代表第八轮以及第九轮中的轮密钥字节。特别值得关注的是, 图中红色部分的中间状态掩码同样在补偿掩码状态派生阶段产生过(即图 13 中的第三个泄露点位置)。因此通过预处理这两个位置的能量泄露, 攻击者可以有效规避乱序防护而直接执行传统二阶攻击。

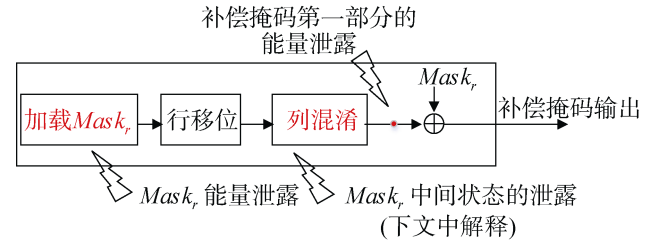


图 11 补偿掩码派生阶段的可利用漏洞

Figure 11 Exploitable vulnerability in the derivation of compensation mask

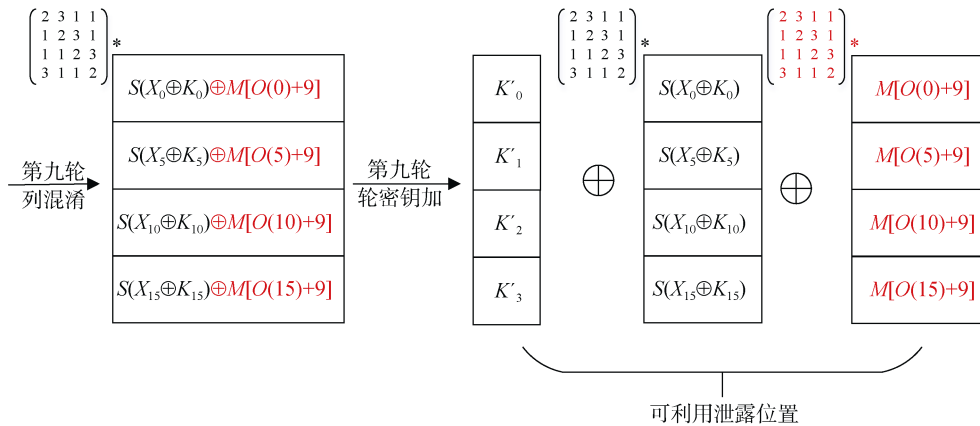


图 12 第九轮轮密钥加中的可利用泄露位置(以状态矩阵第一列为例)

Figure 12 Exploitable leakages in the AddRoundKey operation of the ninth round (Take the first column of the state as an example)

5.1.2 代码实现层面漏洞

• 掩码补偿操作的执行位置漏洞

代码执行层面的漏洞则由掩码补偿操作的执行位置引起。更为准确的说, 在 RSM2.0 源代码中, 当前轮的掩码补偿在轮密钥与下一轮的掩码 S 盒操作之间执行, 如图 13 所示。该看似可忽略的实现顺序却对 RSM2.0 的非模板安全性影响重大。对于第九轮轮密钥加操作产生的中间值, 攻击者能够使用已知密文与

猜测的末轮轮密钥来反向推导得到, 且攻击者每次只需猜测 8 比特子密钥。除此之外, 这里的掩码补偿过程会将其输入值中包含的中间状态掩码值转化为第十轮中的输入掩码状态 $Mask_9$ 。该掩码状态值的能量泄露同样出现在第九轮补偿掩码生成过程中(如图 13)。

因此, 同样通过综合使用这两处的能量泄露, 攻击者能够去除其中随机掩码的影响, 从而设计出另一种新的二阶攻击方案。

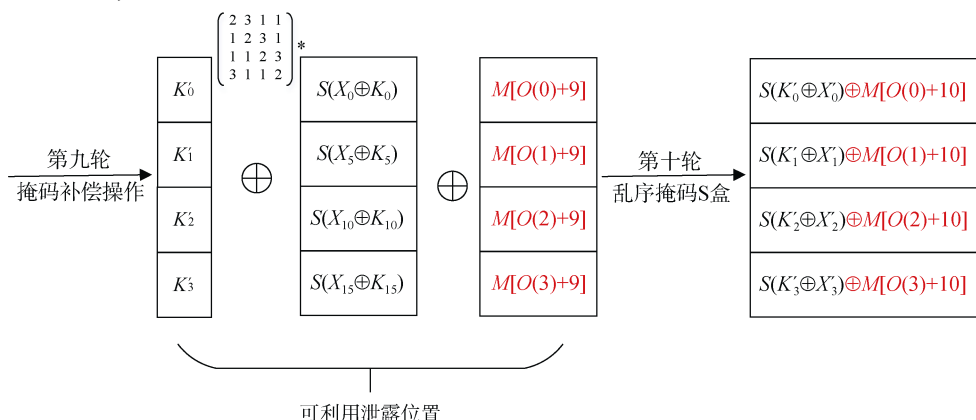


图 13 掩码补偿输出位置的可利用泄露(以状态矩阵第一列为例)

Figure 13 Exploitable leakages in the output position of MaskCompensation operation (Take the first column of the state as an example)

• 线性层变换中的漏洞

更严重的安全漏洞出现在线性层中。由于在针对原始 RSM 方案的竞赛阶段, 几乎所有的提交方案均将非线性层变换, 在进行方案改进时, RSM2.0 方案的设计者过分关注针对该位置 S 盒执行过程的保护, 而忽视了线性层变换中的潜在安全风险。

这里阐述的线性层漏洞主要出现在第一轮加密过程的列混淆操作阶段。而产生该漏洞的主要原因分为以下两个方面。

首先, 在 RSM2.0 的汇编代码实现中, 针对受保护中间值所进行的列混淆操作与补偿掩码派生阶段中执行的列混淆操作采用相同的汇编代码来实现。因此二者在执行过程中唯一的不同点在于输入数据部分, 即两个阶段的输入值分别为:

$ShiftRows(X \oplus K \oplus Mask_1)$ 以及 $ShiftRows(Mask_1)$

此外, 由于列混淆操作的线性变换特性, 这种共享源代码的实现方式意味着当一个受保护的掩码中间值在列混淆变换中被使用或者生成时, 其对应的保护掩码值本身会出现在补偿掩码派生过程中列混淆操作的相应位置。

更糟的是, 汇编代码中实现的列混淆操作以状态字节矩阵的各列为基本操作单元并且按照从左至右的顺序逐列执行列混淆操作过程。这种实现方案意味着攻击者能够在能量曲线的固定时刻点上定位出列混淆中所有计算中间值的能量泄露。

基于以上两方面的原因, 执行传统二阶攻击所需要的所有先决条件均得到了满足。

5.2 非模板类攻击官方评估结果

5.2.1 首轮中的二阶攻击

本小节^①主要阐述针对 RSM2.0 加密首轮中存在的漏洞而设计的两种二阶攻击方案。这两种方案均利用了加密过程和 MCP_r 派生阶段中由于共享列混淆代码而导致的相关联泄露来展开攻击。为了更好的理解下面介绍的非模板类型攻击, 这里首先阐述 RSM2.0 方案中列混淆操作的具体实现方法。

设 $(V_{0,j}, V_{1,j}, V_{2,j}, V_{3,j})^T$ 是列混淆一列 4 字节的输入变量, 则各字节的变换输出值可以形式化的推导并表示为下列形式, 其中 $i \in [0, 3], j \in [0, 3]$:

① 为了表述的需要, 本小节中用到的所有求模操作均明确标出

$$\begin{aligned}
V_{i,j} &= (2 * V_{i,j}) \oplus (3 * V_{(i+1)\%4,j}) \oplus V_{(i+2)\%4,j} \\
&\oplus V_{(i+3)\%4,j} = (2 * V_{i,j} \oplus 2 * V_{(i+1)\%4,j}) \oplus V_{(i+1)\%4,j} \\
&\oplus V_{(i+2)\%4,j} \oplus V_{(i+3)\%4,j} = V_{i,j} \oplus (V_{1,j} \oplus V_{2,j} \oplus V_{3,j} \oplus V_{4,j}) \\
&\oplus 2 * (V_{i,j} \oplus V_{(i+1)\%4,j})
\end{aligned}$$

在具体的源代码实现中, 官方组委会依照公式末行的推导形式来编写列混淆代码。换言之, 计算中间值 $V_{1,j} \oplus V_{2,j} \oplus V_{3,j} \oplus V_{4,j}$ 首先被求出并存放在临时寄存器中。由于列混淆变换时单列四字节中任意字节的更新均需要使用上述计算中间值, 该中间值被循环使用 4 次。随后, 为了派生得到指定字节 $V_{i,j}$, $(V_{i,j} \oplus V_{(i+1)\%4,j})$ 被后续计算产生。该异或结果随后作为查找表 *Xtime* 的输入, 查表获得相应的输出。*Xtime* 查找表的生成规则是在索引值为 X , $X \in [0, 255]$ 的位置中存储有限域 $GF(2^8)$ 意义下的 $2X$ 变量值。因此临时变量 $(V_{i,j} \oplus V_{(i+1)\%4,j})$ 的查表输出结果即为 $2 * (V_{i,j} \oplus V_{(i+1)\%4,j})$ 。最终, 通过分别异或 $(V_{1,j} \oplus V_{2,j} \oplus V_{3,j} \oplus V_{4,j})$, $2 * (V_{i,j} \oplus V_{(i+1)\%4,j})$ 以及初始值 $V_{i,j}$, 指定字节 $V_{i,j}$ 的列混淆更新结果即为所求。

• 针对 *Xtime* 值生成过程的攻击方案

在生成 *Xtime* 输入值 $(V_{i,j} \oplus V_{(i+1)\%4,j})$ 的过程中, 第一个元素 $V_{i,j}$ 被首先加载到临时寄存器中, 因此与该变量相关的芯片能量消耗信息发生了泄露。在针对受保护中间值的加密阶段, $V_{i,j}$ 实际上代表了掩码 S 盒的一个输出字节, 因此所有掩码 S 盒输出字节 $S(X_i \oplus K_i) \oplus M[(O(i)+1)\%16]$, $i \in [0, 15]$ 的信息均发生了泄露。另一方面, 在掩码补偿阶段, $V_{i,j}$ 则代表了输入掩码状态矩阵 $Mask_1$ 中的一个字节。如此一来, 随机掩码值 $M[(O(i)+1)\%16]$ 本身的泄露同样存在。通过对两部分泄露 (Z, Q) 进行“减均值相乘” $(Z - \bar{Z}) * (Q - \bar{Q})$ 的二阶预处理变换, 预处理之后的联合泄露值将于下列计算中间值存在相关性:

$$(S(X_i \oplus K_i) \oplus M[(O(i)+1)\%16]) \oplus M[(O(i)+1)\%16] = S(X_i \oplus K_i), i \in [0, 15]$$

该计算中间值是为仅包含 8 比特子密钥信息的无保护中间值, 因此后续采用相关系数攻击即可逐步恢复其中所有的 128 比特密钥值。

官方评估结果显示, 这里提出的二阶攻击方法切实可行。基于公开的性能评估参数, 该方案只需要 258 条能量曲线(图 14)就能够以 80% 的全局成功率恢复 RSM2.0 方案中所有的 128 比特主密钥信息。而仅仅需要 210 条能量曲线就能将局部猜测熵的最大值降低到 10 以下。这意味着在使用 210 条能量曲线完

成二阶攻击之后, 攻击者后续只需暴力搜索最多 10^{16} 的剩余密钥候选空间即可将正确子密钥的值恢复出来。

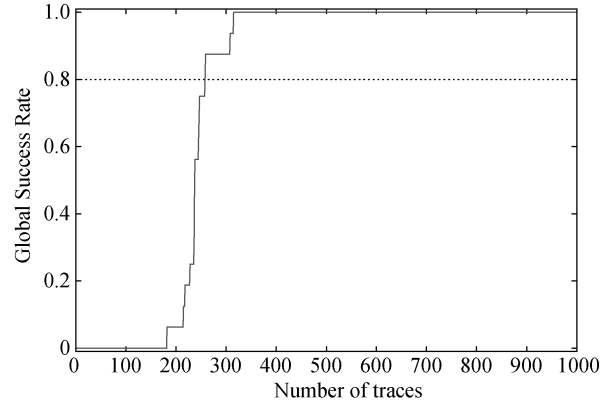


图 14 针对 *Xtime* 输入攻击方案的全局成功率
官方评估结果

Figure 14 Official evaluation result of GSR for the attacking scheme aiming at *Xtime* input

• 优化的链式攻击方案

与上述攻击不同, 这里我们利用了生成 *Xtime* 完整输入值时产生的能量泄露来进行攻击, 该输入值即为 $(V_{i,j} \oplus V_{(i+1)\%4,j})$ 。通过对受保护中间值加密以及补偿掩码计算这两个过程中列混淆泄露的二阶预处理, 联合泄露值将与攻击者可预测的中间值之间存在相关性。而该假设中间值里实际上包含了 16 比特的密钥信息。以 RSM2.0 加密过程中的首列四字节元素为例, 去除掩码保护之后的四个假设中间值分别为:

$$\begin{aligned}
&S(X_0 \oplus K_0) \oplus S(X_5 \oplus K_5), \\
&S(X_5 \oplus K_5) \oplus S(X_{10} \oplus K_{10}), \\
&S(X_{10} \oplus K_{10}) \oplus S(X_{15} \oplus K_{15}), \\
&S(X_{15} \oplus K_{15}) \oplus S(X_0 \oplus K_0)
\end{aligned}$$

这其中, X_i 以及 K_i 代表明文以及主密钥中的第 i 个字节位置。针对这些假设中间值直接进行能量分析攻击是可行的。通过将上述第一个和第三个中间值做为攻击目标或者以第二个与第四个中间值为攻击目标, 攻击者就能获得第一列中包含的所有密钥信息。尽管能够进行实际攻击, 攻击者针对每列攻击时的密钥猜测总空间为 $2 * 2^{16}$, 因此其计算开销仍然过大。

为了进一步降低计算开销, 下面针对上述攻击过程进行优化, 并提出一种全新的链式攻击策略。具体的方案是, 攻击者首先使用在本小节首个攻击方案中提及的能量泄露优先对每列中的首个密钥值进行破解, 如第一列中的 K_0 。随后, 利用攻击中得到的

最大可能性 K_0 猜测值, 攻击者进一步对本方案中展示的目标中间值 $S(X_0 \oplus K_0) \oplus S(X_5 \oplus K_5)$ 进行攻击。在本次攻击中, 由于 K_0 已在先前的攻击过程中进行过确定, 因此这里只需单独对子密钥 K_5 进行猜测破解即可。同样的方法随后针对第二个目标中间值 $S(X_5 \oplus K_5) \oplus S(X_{10} \oplus K_{10})$ 展开执行, 在子密钥 K_5 已被以最大概率恢复的前提下这一次攻击者需要恢复的子密钥目标仅为 8 比特的 K_{10} 。以此类推, K_{15} 同样能够以相同的方法进行有效恢复。通过这种方法, 提取每列四个子密钥所需要的密钥猜测空间由 2×2^{16} 降低至 4×2^8 , 而由于 RSM2.0 方案列混淆操作中的四列状态变换各自独立, 因此针对另外三列中所包含密钥的恢复过程与第一列中的密钥恢复过程完全相同。

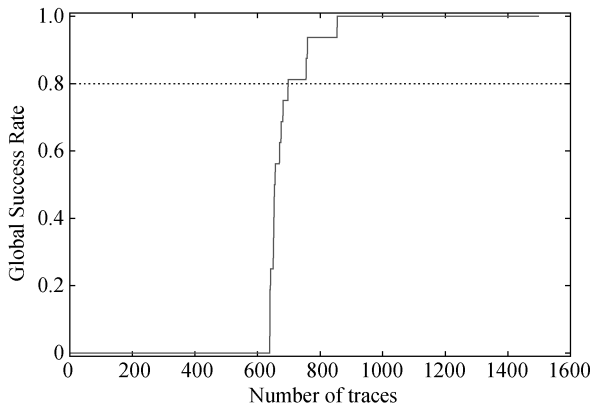


图 15 针对链式攻击方案的全局成功率官方评估结果

Figure 15 Official evaluation result of GSR for the optimized chained scheme

图 15 展示了我们所实现的这种链式攻击方案的官方评估结果。从评估结果来看, 这种攻击方法能够有效破解 RSM2.0 复合防护方案, 并且该方案只需要 565 条能量曲线就能够以 80% 的全局成功率恢复 RSM2.0 方案中包含的所有主密钥信息。

同样值得关注的是, 类似的二阶链式攻击方法还可以针对另外两个计算位置展开, 它们分别是 Xtime 的查表计算结果 $2 * (V_{i,j} \oplus V_{(i+1) \% 4, j})$ 以及列混淆推导公式中 $(V_{1,j} \oplus V_{2,j} \oplus V_{3,j} \oplus V_{4,j})$ 中间值的生成过程。在这两个位置进行链式二阶攻击与上述攻击方案唯一的不同点在于其所攻击的目标中间值有所不同。为了避免内容重复, 针对这两个位置的二阶攻击详情在这里不再赘述。

5.2.2 第九轮中的二阶攻击

• 针对掩码补偿输出的攻击方案

这里介绍的攻击方案主要利用了 5.1.2 小节中阐

述的算法实现层面的漏洞。由于新引入的掩码补偿过程被安排在当前轮的轮密钥加变换操作之后执行, 因此掩码补偿变换的每个输出字节仅包含 8 比特的轮密钥信息(利用密文进行反向推导可得)。另一方面, 由于经过了掩码补偿变换, 这部分中间值实际上是由第十轮的输入掩码值 $Mask_9$ 进行保护的。除此之外, 在生成第九轮补偿掩码的过程中, 当 $Mask_9$ 矩阵被逐字节顺序加载进临时变量寄存器时产生了相应的能量消耗, 如图 11 中所示。因此, 通过结合使用这两个泄露位置的能量消耗, 联合得到的能量泄露与下列未受掩码保护的目标中间值之间存在相关性, 因此成为另一个可以进行二阶攻击的泄露位置:

$$\begin{aligned} & (Sbox^{-1}ShiftRows^{-1}[C \oplus K_L] \oplus Mask_9) \oplus Mask_9 \\ & = Sbox^{-1}ShiftRows^{-1}[C \oplus K_L] \end{aligned}$$

其中, C 表示密文输出结果, K_L 表示最后一轮轮密钥。当 16 字节的 K_L 被逐字节恢复出来之后, 再通过执行反向密钥编排过程既可以获得 128 比特的 RSM2.0 主密钥。

图 16 描述了 DPA Contest 官方组委会对本节所述攻击方案的评估结果。该结果显示, 通过利用实现层漏洞来执行的攻击方案性能高效。只需使用 257 条能量曲线就能够达到 80% 全局成功率的性能指标。除此之外, 为了将部分猜测熵的最大值降低至 10 以下, 该攻击方案仅需要 205 条能量曲线就可以达到要求。

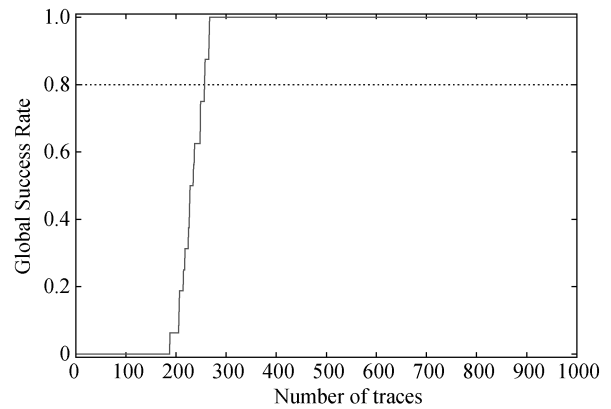


图 16 针对掩码补偿攻击方案的全局成功率官方评估结果

Figure 16 Official evaluation result of GSR for the attacking scheme aiming at Mask Compensation

• 针对轮密钥加输出值的二阶攻击

第九轮轮密钥加操作输出值与第九轮掩码补偿操作输出值之间存在的唯一差别在于用以保护计算中间值的随机掩码不尽相同。对于轮密钥加操作来

说, 它所对应的输出掩码值是输入掩码状态 $Mask_8$ 经过第九轮的掩码 S 盒, 行移位, 列混淆这一系列变换所得到的。因此, 第九轮变换中的轮密钥加操作输出值可以按照如下公式从密文 C 中推导得到:

$$Sbox^{-1}ShiftRows^{-1}[C \oplus K_L] \oplus MixColumns[ShiftRows[Mask_9]]$$

另一方面, 在当前轮补偿掩码的构造过程中, 补偿掩码的第一部分值 $MixColumns[ShiftRows[Mask_9]]$ 需要被首先产生, 如图 11 中所示。类似的, 通过联合利用这两个位置的泄露, 包含在下面目标中间值中的最后一轮轮密钥值 K_L 可以通过二阶攻击的方法恢复出来:

$$Sbox^{-1}ShiftRows^{-1}[C \oplus K_L]$$

此后, 作为最终攻击目标的主密钥信息则可以通过反向密钥编排的方式得以恢复。

官方针对全局成功率参数评估出的最终结果如图 17 所示。该结果表明这种攻击方案能够有效破解 RSM2.0 算法主密钥并且该方案需要使用 698 条能量曲线来达到 80% 的主密钥恢复全局成功率。

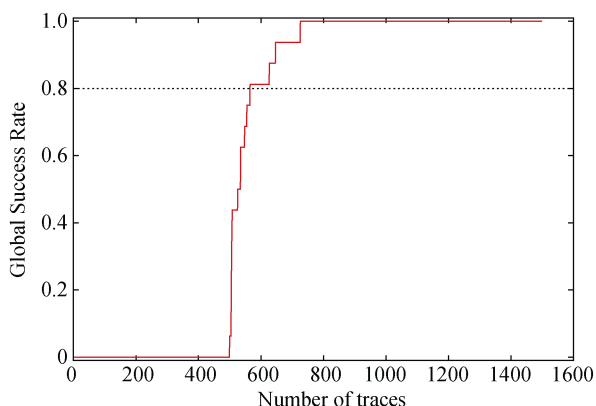


图 17 针对轮密钥加攻击方案的全局成功率
官方评估结果

Figure 17 Official evaluation result of GSR for the attacking scheme aiming at AddRoundKey

6 防御方案讨论

针对第四章与第五章中提出的多种模板类与非模板类攻击方案, 本章进一步对可能的防御方案改进策略进行讨论, 以便能够有效缓解甚至彻底消除上文中提出的多种安全威胁。

6.1 泄露指纹长度截断对策

从第四章中针对泄露指纹攻击技术的实验数据以及讨论中可以看出, 一种可行的防御对策思路是尽可能的缩短攻击者能够提取到的泄露指纹长度, 从而使得 16 种泄露指纹猜测序列中更多的出现重复的猜测序列。这样一来, 攻击者在进行随机掩码或者

随机偏移索引这两个敏感信息的识别过程中无法唯一的确定敏感信息的正确取值, 进而能够有效抵御本文中提出的泄露指纹攻击技术。

从缩短泄露指纹对策可能引起的额外资源开销角度来看。要使得攻击者可提取的泄露指纹长度缩短为原来的 $1/n$, 则防御方案中需要使用的随机数将相应的增加为原来的 n 倍。具体来说, 由于我们在 4.2 节的讨论中验证了随机偏移指纹的最短可区分长度为 3, 因此一种绝对安全且兼顾资源开销的泄露指纹截断长度为 2。换言之, 改进之后的防御对策应当在 RSM2.0 算法每执行两轮加密操作之后重新更新 16 个随机偏移索引值 $O(i)$, $i \in [0, 15]$, 从而保证攻击者无法获取有效长度的可区分泄露指纹。而这样一来, 更新的防御方案相比于当前的 RSM2.0 防御需要多引入共 80 个随机数的额外资源开销。

当然由于存在密码芯片噪声、标准模板攻击恢复偏差等诸多方面的原因, 在实际的攻击过程中即使可提取的泄露指纹长度超过了 3 个泄露位置的理论安全长度, 攻击者的敏感信息识别准确度依旧无法做到 100% 的准确且该识别准确度会随着泄露指纹长度的减小而出现显著的下滑。因此, 防御方案设计人员应当根据密码算法在实现中可利用的配置资源情况以及期望该算法实现达到的安全强度来合理的权衡泄露指纹长度的截断程度, 以便能够在缩短泄露指纹长度多带来的算法安全性提升与额外引入的随机数开销之间找到最佳的平衡点。

6.2 指令置换与扩展乱序区域对策

为了有效减轻甚至彻底消除第 5 章中提及的非模板类型攻击威胁, 本小节主要阐述了如下的针对性防御对策。这些对策遵循的基本原则是: 在二阶攻击中利用的两部分能量泄露均需要被有效保护, 相应的保护措施可以是消除其中的任何一个泄露源, 或者是将该泄露源在能量曲线上的出现位置进行随机化处理。

1. 在第九轮中增强能量分析抵抗力: 在第九轮中将轮密钥加操作与掩码补偿操作的操作顺序进行互换可以成为增强 RSM2.0 方案非模板攻击安全性的第一步。实际上, 与上节中介绍的针对掩码补偿输出值泄露的二阶攻击相比, 通过执行上述的改进方案, 攻击者当前需要攻击的目标中间值就变为如下形式, 其中的每个目标字节均包含了 16 比特的子密钥信息: $Sbox^{-1}ShiftRows^{-1}[C \oplus K_L] \oplus K_P$, 其中, 符号 K_L 和 K_P 分别代表了最后一轮与倒数第二轮的轮密钥信息。

通过互换轮密钥加与掩码补偿的改进策略, 上

述两个变换操作的输出值均由 $Mask_9$ 状态进行保护。而为了进一步消除二阶攻击威胁, 顺序执行的 $Mask_9$ 加载操作, 异或补偿掩码操作以及轮密钥加操作都需要添加乱序防护。上述乱序防御方案的实现简单可行, 原因在于这些操作中对于 16 个状态字节的变换处理是相互独立的。

2. 在第一轮中增强能量分析抵抗力: 在补偿掩码派生阶段以及掩码中间值加密阶段共享的列混淆代码造成了上文中提及的加密第一轮中的非模板类攻击。为了对抗这种类型的攻击, 一种可能的防御策略是在两个阶段中分别使用两种不同源代码实现列混淆过程。例如, 将加密阶段的列混淆实现按照下列的公式推导过程进行修改:

$$V_{i,j} = 2 * V_{i,j} \oplus 3 * V_{(i+1)\%4,j} \oplus V_{(i+2)\%4,j} \oplus V_{(i+3)\%4,j}$$

这样一来, 由于在不同阶段的列混淆实现中生成的所有派生中间值不再包含相同掩码值, 上一节中提到的所有链式攻击方法都将失效。

为了进一步防止顺序的列混淆输入加载过程中出现的可利用泄露, 以字节为单位的乱序列混淆变换过程是一种可能的解决方案。

3. 需要特别声明的是: 在具体实现过程中, 上述改进方案里提及的需要添加乱序防护的位置无需引入额外的乱序数组。方案实现人员只需要注意在掩码自身泄露位置与受掩码保护的中间值泄露位置分别使用不同的乱序数组(如使用已有的 $Sf0[]$ 和 $Sf10[]$ 数组)即可在不增加资源开销的前提下有效提高当前 RSM2.0 方案安全性。具体来说, 通过这种改进方案, 原始二阶攻击中利用的两个固定位置的泄露位置现在将会出现在 16^2 种可能的泄露位置上。因此这样一来, 攻击者从目标泄露中提取有效信息的难度将会大大提高。而利用文献[11, 15]中提到的“聚合-结合”型二阶攻击方案来破解上述添加的乱序防护又会引入巨大的计算开销。

7 总结

本文针对当下最新的密码算法防御方案设计理念, 以 DPA Contest 官方组委会所设计并实现的增强型旋转 S 盒掩码方案为具体目标, 首先提出了一种通用的能量分析漏洞检测方案, 利用该方案所检测出的敏感区域, 本文分别研究了 RSM2.0 中存在的模板类与非模板类安全问题, 并分别提出了一系列相应的攻击方案。

在模板类攻击方案中, 本文首次提出了一种泄露指纹利用技术。该技术使用从能量曲线水平方向上提取出的一系列相关泄露来构造指纹序列, 进而

利用该泄露序列来完成密码算法中敏感信息的精确关联与恢复。为了验证泄露指纹技术的健壮性, 本文在不同的噪声水平环境中对该项技术进行了全面评估。同时, 为了进一步降低该项技术的计算与存储开销, 本文还深入讨论了可能的泄露指纹优化方案并首次提出了一种 MOND 量化指标, 用以衡量不同的泄露位置选择下泄露指纹攻击方案的攻击性能优劣。

此外, 在非模板类攻击方案中, 针对 RSM2.0 防御方案中存在的非模板类安全隐患, 本文又提出了一系列的二阶能量攻击方案。这些方案利用乱序防御方案在实现过程中存在的漏洞, 通过结合两部分能量泄露的方式消除了 RSM2.0 方案中随机掩码的保护效果, 进而能够直接使用传统的能量分析方法对 RSM2.0 算法主密钥进行全面破解。

最后, 为了抵抗本文中提出的诸多模板类与非模板类能量分析攻击安全威胁, 本文针对 RSM2.0 的可能改进策略同样展开了讨论。这些改进对策能够部分消除算法中存在的安全漏洞, 降低泄露指纹利用技术的敏感信息识别成功率, 或者是进一步隐藏 RSM2.0 算法在实际执行过程中产生的敏感信息相关的能量泄露。因此, 从总体上来看, 这些新提出的防御对策不仅能够有效提升 RSM2.0 密码算法在实现中的安全性, 同时也能够对同类密码算法防御方案的设计提供重要的指导作用。

参考文献

- [1] T. S. Messerges, "Using second-order power analysis to attack DPA resistant software," In *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES'00)*, pp. 238-251, 2000.
- [2] L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," In *Proc. Public Key Cryptography (PKC'03)*, pp. 199-210, 2003.
- [3] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-Analysis Attack on an ASIC AES implementation," In *Proc. Information Technology: Coding and Computing (ITCC'04)*, pp. 546-552, 2004.
- [4] D. Shanmugam, R. Selvam, and S. Annadurai, "Differential power analysis attack on SIMON and LED block ciphers," In *Proc. International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE'14)*, pp. 110-125, 2014.
- [5] J.-S. Coron, and L. Goubin, "On boolean and arithmetic masking against differential power analysis," In *Proc. Cryptographic Hardware and Embedded Systems (CHES'00)*, pp. 231-237, 2000.
- [6] M. Rivain, and E. Prouff, "Provably secure higher-order masking of aes," In *Proc. Cryptographic Hardware and Embedded Systems*

- (CHES'10), pp. 413–427, 2010.
- [7] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” In *Proc. Cryptographic Hardware and Embedded Systems (CHES'04)*, pp. 16–29, 2004.
 - [8] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” In *Proc. Advances in Cryptology (CRYPTO'99)*, pp. 388–397, 1999.
 - [9] S. Mangard, E. Oswald, and T. Popp, “Power analysis attacks: Revealing the secrets of smart cards,” *Springer Science & Business Media*, 2008.
 - [10] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert, “Shuffling against side-channel attacks: A comprehensive study with cautionary note,” In *Proc. Advances in Cryptology (ASIACRYPT'12)*, pp. 740–757, 2012.
 - [11] S. Tillich, and C. Herbst, “Attacking state-of-the-art software countermeasures—a case study for aes,” In *Proc. Cryptographic Hardware and Embedded Systems (CHES'08)*, pp. 228–243, 2008.
 - [12] S. Bhasin, N. Bruneau, J.-L. Danger, S. Guilley, and Z. Najm, “Analysis and improvements of the dpa contest v4 implementation,” In *Proc. Security, Privacy, and Applied Cryptography Engineering (SPACE'14)*, pp. 201–218, 2014.
 - [13] C. Herbst, E. Oswald, and S. Mangard, “An aes smart card implementation resistant to power analysis attacks,” In *Proc. Applied Cryptography and Network Security (ACNS'06)*, pp. 239–252, 2006.
 - [14] M. Rivain, E. Prouff, and J. Doget, “Higher-order masking and shuffling for software implementations of block ciphers,” In *Proc. Cryptographic Hardware and Embedded Systems (CHES'09)*, pp. 171–188, 2009.
 - [15] S. Tillich, C. Herbst, and S. Mangard, “Protecting aes software implementations on 32-bit processors against power analysis,” In *Proc. Applied Cryptography and Network Security (ACNS'07)*, pp. 141–157, 2007.
 - [16] M. Nassar, S. Guilley, and J.-L. Danger, “Formal analysis of the entropy/security trade-off in first-order masking countermeasures against side-channel attacks,” In *Proc. Progress in Cryptology (INDOCRYPT'11)*, pp. 22–39, 2011.
 - [17] C. Carlet, and S. Guilley, “Side-channel indistinguishability,” In *Proc. the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP'13)*, p. 9, 2013.
 - [18] Implementation of the dpa contest v4.2 on the atmel atmega-163 smart card. http://www.dpacontest.org/v4/data/v4_2/smart_v42_2.zip.
 - [19] S. Mangard, “A simple power-analysis (spa) attack on implementations of the aes key expansion,” In *Proc. Information Security and Cryptology (ICISC'02)*, pp. 343–358, 2002.
 - [20] J. VanLaven, M. Brehob, and K. J. Compton, “A computationally feasible spa attack on aes via optimized search,” In *Proc. Security and Privacy in the Age of Ubiquitous Computing (SEC'05)*, pp. 577–588, 2005.
 - [21] C. Clavier, D. Marion, and A. Wurcker, “Simple power analysis on aes key expansion revisited,” In *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES'14)*, pp. 279–297, 2014.
 - [22] Hall of fame in DPA contest v4.2 competition. http://www.dpacontest.org/v4/42_hall_of_fame.php.
 - [23] Z. Martinasek, F. Iglesias, L. Malina, and J. Martinasek, “Crucial pitfall of dpa contest v4.2 implementation,” In *Proc. Security and Communication Networks (SCN'16)*, pp. 6094–6110, 2016.
 - [24] T. Schneider, and A. Moradi, “Leakage assessment methodology,” In *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES'15)*, pp. 495–513, 2015.
 - [25] Z. Martinasek, P. Dzurenda, and L. Malina, “Profiling power analysis attack based on mlp in dpa contest v4. 2,” In *Proc. Telecommunications and Signal Processing (TSP'16)*, pp. 223–226, 2016.
 - [26] Z.Y. Liu, N. Gao, C.Y. Tu, Y. Ma, and Z.B. Liu, “Detecting Side Channel Vulnerabilities in Improved Rotating S-Box Masking Scheme—Presenting Four Non-profiled Attacks,” In *Proc. International Conference on Selected Areas in Cryptography (SAC'16)*, pp. 41–57, 2016.
 - [27] Z.Y. Liu, N. Gao, C.Y. Tu, J. Zhou, Y. Ma, and Y. Zhao, “Leakage Fingerprints: A Non-negligible Vulnerability in Side-Channel Analysis,” In *Proc. the 11th ACM on Asia Conference on Computer and Communications Security (ASIACCS'16)*, pp. 807–818, 2016.
 - [28] A. Battistello, J. S. Coron, E. Prouff, and R. Zeitoun, “Horizontal side-channel attacks and countermeasures on the ISW masking scheme,” In *Proc. International Conference on Cryptographic Hardware and Embedded Systems (CHES'16)*, pp. 23–39, 2016.
 - [29] M. Rivain, and E. Prouff, “Provably secure higher-order masking of AES,” In *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES'10)*, pp. 413–427, 2010.
 - [30] A. Moradi, M. Kasper, and C. Paar, “Black-box side-channel attacks highlight the importance of countermeasures,” In *Proc. Topics in Cryptology (CT-RSA'12)*, pp. 1–18, 2012.
 - [31] S. Chari, J. R. Rao, and P. Rohatgi, “Template attacks,” In *Proc. Cryptographic Hardware and Embedded Systems (CHES'02)*, pp. 13–28, 2002.
 - [32] F.-X. Standaert, T. G. Malkin, and M. Yung, “A unified framework for the analysis of side-channel key recovery attacks,” In *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'09)*, pp. 443–461, 2009.
 - [33] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis

with a leakage model,” In *Proc. Cryptographic Hardware and Embedded System (CHES'04)*, pp. 16–29, 2004.

- [34] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, “Mutual information analysis,” In *Proc. Cryptographic Hardware and Embedded*

Systems (CHES'08), pp. 426–442, 2008.

- [35] W. Schindler, K. Lemke, and C. Paar, “A stochastic model for differential side channel cryptanalysis,” In *Proc. Cryptographic Hardware and Embedded Systems (CHES'05)*, pp. 30–46, 2005.



刘泽艺 于 2017 年在中国科学院大学信息安全专业获得博士学位。现任中国科学院信息工程研究所助理研究员。研究领域为大数据分析 & 分布式资源管理。研究兴趣包括: 侧信道攻防技术、自然语言处理。Email: liuzeyi@iie.ac.cn



高能 于 2006 年在中国科学院研究生院通信与信息系统专业获得博士学位。现为中国科学院信息工程研究所研究员, 现任信息安全国家重点实验室副主任。研究领域为网络安全、系统安全。研究兴趣包括: 安全事件分析、社交网络隐私。Email: gaoneng@iie.ac.cn



查达仁 于 2010 年在中科院研究生院信息安全专业获博士学位。现任中国科学院信息工程研究所第四工程部副主任。研究领域为大数据存储与融合技术、智能数据处理等。研究兴趣包括: 密码工程与应用, 网络体系结构与安全防护。Email: zhadaren@iie.ac.cn



屠晨阳 于 2016 年在中国科学院大学获得信息安全博士学位。现任中国科学院信息工程研究所助理研究员。研究领域为端系统安全。研究兴趣包括: 侧信道分析与防御等。Email: tuchenyang@iie.ac.cn