基于软件指令定位的新型高阶侧信道分析方法

郭志鹏[1] 唐 明[1),2)] 胡晓波[3)] 李煜光[1)] 彭国军[1)] 张焕国[1)]

[1)](武汉大学国家网络安全学院空天信息安全与可信计算教育部重点实验室 武汉 430072)

[2)](巴黎高科国立高等电信学校 巴黎 75634 法国)

[3)](北京智芯微电子科技有限公司电力芯片设计分析国家电网公司重点实验室 北京 100080)

摘 要 现有的大多数高阶掩码方案都采用了软件实现方式,这样可以防止硬件电路毛刺产生的安全泄漏,同时不会受到硬件平台资源不足的限制.在目前针对高阶掩码方案的多种分析方法中,高阶侧信道分析是最有效的分析方法之一.即使是满足理论安全性的高阶掩码方案,高于理论安全阶数的高阶侧信道分析依然可以对其进行攻击.然而,当掩码方案的阶数很高时,由于高阶分析的时间复杂度和数据复杂度非常高,这使得高阶分析方法难以成功实施.在该文中,作者基于指令识别提出了一种特征点选取方法,称作指令定位特征点选择方法(Instruction Recognition-based Points of Interest Selection,IR-PoIS).通过定位与敏感信息相关的指令,IR-PoIS 方法可以确定高阶掩码方案中每个秘密共享因子对应功耗出现的具体位置,从而降低后续高阶分析方法的时间复杂度.对于一个有 n 个秘密共享因子的高阶掩码方案,IR-PoIS 方法可以将高阶分析方法的时间复杂度水平从功耗曲线点数的 n 次方降低到线性水平,大大提高了高阶分析方法的效率.并且由于目标 CPU 的指令集是已知的,攻击者可以事先对敏感指令进行建模.在 SASEBO-W 开发板上的实验中,作者对 *LDD* 指令进行了定位.实验结果表明,只需十条左右的功耗曲线,IR-PoIS 的定位成功率就可以达到 100%.这表明 IR-PoIS 方法是一种非常高效的特征点选择方法.在此基础上,该文成功地攻击了三阶 Coron14 掩码方案的软件实现,验证了基于 IR-PoIS 的高阶侧信道分析方法的有效性.

关键词 侧信道分析;高阶分析;高阶掩码方案;指令识别;软件实现

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2019.00929

Instruction-Location-Based Analysis Against Software Implementation of Higher-Order Masking

GUO Zhi-Peng[1)] TANG Ming[1),2)] HU Xiao-Bo[3)] LI Yu-Guang[1)]

PENG Guo-Jun[1)] ZHANG Huan-Guo[1)]

[1)](*Key Laboratory of AIS& TC, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan* 430072)

[2)](*The Telecom ParisTech, Paris* 75634 *France*)

[3)](*State Grid Key Laboratory of Power Industrial Chip Design and Analysis Technology, Beijing Smart-Chip Microelectronics Technology Co., Ltd., Beijing* 100080)

Abstract Masking is a very common and popular countermeasure against side-channel attacks. In a masking scheme, any sensitive variables in a cryptographic implementation are randomized by mask sequences. In order to resist higher-order attacks, masking schemes have been developed into higher-order masking schemes. A software implementation is the most common design of

收稿日期:2017-08-02;在线出版日期:2018-07-19.本课题得到国家自然科学基金(61472292,61332019)、密码科学技术国家重点实验室开放课题基金(面上项目)(MMKFKT201821)、湖北省技术创新专项(2018AAA046)和密码芯片防护设计下的侧信道分析检测技术研究(2018J-10)资助.**郭志鹏**,博士研究生,主要研究方向为密码学、侧信道分析与防护.E-mail:whu_guozhipeng@163.com.**唐 明**(通信作者),博士,教授,主要研究领域为密码学、侧信道分析与防护.E-mail:m.tang@126.com.**胡晓波**,学士,高级工程师,主要研究方向为密码芯片设计.**李煜光**,硕士研究生,主要研究方向为密码学、侧信道分析与防护.**彭国军**,博士,教授,主要研究领域为网络与信息系统安全.**张焕国**,教授,主要研究领域为信息安全、可信计算.

higher-order maskings for overcoming the glitch weakness and resource limitation. Until now, higher-order SCAs have been the only challenge to higher-order masking schemes, which were proven to be theoretically secure. However, owing to the large time and data complexities, higher-order analyses are sometimes regarded as infeasible when the order is very high. Thus, it is very important to identify the interesting tuples (or to narrow down a window of time samples as much as possible) prior to key recovery in order to maintain the computational complexity of a multivariate attack at a feasible level. Nevertheless, the existing PoI methods are based on exhaustive searching, and the time complexity is determined by the size of the sample window, the number of samples, and the order of the masking. In this paper, we propose a PoI selection method called IR-PoIS to locate the interesting points for the higher-order analysis because it is based on instruction recognition. By targeting the locations of the instructions corresponding to the sensitive information, IR-PoIS can find the precise locations of the different shares in a masking scheme. It is noted that IR-PoIS can decrease the time complexity from polynomial of degree n to linear in the number of points of a power trace, where n is the number of shares in the masking, which is a notable improvement in the higher-order analysis. As the RISC set is very popular in most existing CPUs, it is reasonable for an analyzer to build all the templates for sensitive instructions before collecting several power traces to analyze. In order to demonstrate the practicality and effectiveness of IR-PoIS, we physically evaluated the locations of 10*LDD* instructions in a test program and acquired a 100% success rate within dozens of traces and a stable 100% success rate within one thousand traces. For the locations of the other instructions, we found that it may lose several interesting points when the sample frequency is not sufficiently high and improved the frequency of the *STD* instruction to acquire a 100% success rate for the location. Furthermore, we analyzed a software implementation of Coron14 masking with the open code on GitHub(https://github.com/coron/htable/), and the results of the analysis verified that IR-PoIS can target the precise locations of sensitive instructions within limited time and data complexities. The IR-PoIS method verified that the instructions executed in software are not hidden by the existing masking schemes, and an adversary can build templates of the sensitive instructions at first and successfully complete a higher order analysis. It is obvious that the location of the instruction is another word for targeting sensitive data. On the basis of the results obtained by the IR-PoIS method, one need rechecking the difficulty and security of the existing higher-order masking schemes.

Keywords SCA; higher-order analysis; higher-order masking; instruction recognition; software implementation

1 引 言

侧信道分析(Side-Channel Analysis,SCA)可以利用一个密码设备运行过程中泄漏的时间、功耗以及电磁等物理特征来恢复秘密信息.侧信道分析攻击已经成为密码设备安全的一个严重威胁,抵御侧信道分析的重要性日益突出.

面对侧信道分析时,掩码防护是一种非常常用的防护方法.在掩码方案中,密码算法中的每一个敏感变量都被拆分成一个随机的掩码序列[1].目前的掩码方案已经发展到了高阶掩码防护方案[2-9].高阶掩码防护方案的主要思想是将密码算法中任意一个敏感变量 x 拆分成 n 个随机的秘密共享因子 x_i.在一个 $d+1$ 阶掩码方案中,任意 d 个中间变量值的联立都和敏感变量相互独立.在现有的多种高阶掩码方案中,敏感变量的拆分方式多种多样[10-14],常见的拆分方式有乘法掩码($x=\otimes_{i=1}^{n}x_i$)和布尔掩码($x=\oplus_{i=1}^{n}x_i$)两种,其中布尔掩码是高阶掩码方案中最为常用的拆分方式[15].

高阶侧信道分析是目前针对高阶掩码方案最为有效的分析方法.然而,目前的高阶分析方法仍有一

些局限性，难以在掩码方案的阶数较高时使用. 首先，针对软件串行实现的掩码方案，高阶侧信道分析需要联立多个时刻的功耗，通过分析联立功耗和秘密信息之间的关联性恢复秘密信息，而联立多个功耗区域进行攻击会对噪声更加敏感，随着掩码阶数的增加，联立的功耗与敏感信息的相关性呈指数级别下降. 这使得一次成功攻击需要的功耗曲线数量随着阶数的增大呈指数级别增加. 其次，这样的联立策略导致高阶分析需要遍历功耗曲线中任意 $d+1$ 个样本点的组合. 因此，高阶分析的渐进时间复杂度水平趋近于曲线样本点总数的 $d+1$ 次方. 因此，在高阶掩码方案提高了掩码阶数 $d+1$ 后，高阶分析所需的功耗曲线量和计算时间将大大增加，变得难以实际实施[16].

要提高高阶分析的效率，需要分别针对高阶分析的两个阶段进行分析. 第一个阶段是寻找与敏感中间变量相关的功耗点位置. 这些功耗点位置通常称为特征点. 这个过程可以被称为特征点选取过程. 第二阶段就是将特征点进行联立，通过选择合适的联立函数，使特征点的联立与敏感信息具有尽可能高的相关性[17]. 由于掩码方案的源码和运行平台很有可能是公开的[7]，因此假设攻击者知道算法源码和处理器进行的操作是很合理的. 然而，即使在这个前提下，现有的特征点选取方法依然是高阶分析方法中最耗时的阶段[4]. 在安全评估时，由于测评机构需要在一定时间内完成安全评估[18]，因此分析方法的时间消耗是安全评估有效的最重要因素之一. 在本文中，我们目标是找到一种能够高效分析高阶掩码方案软件实现的高阶侧信道分析方法. 如果高阶分析的时间或数据复杂度不再随着目标掩码方案阶数的提高而迅速增加，那么将大大提高安全测试的效率，从而推动现有可证明安全的掩码方案在软件平台上的安全实现方案的发展.

本文将高阶掩码方案的软件实现作为主要分析对象. 其主要原因是，现在已经发现在高阶掩码硬件实现中存在安全漏洞，比如电路毛刺[14]. 对此，一些基于多方安全计算协议的方案[19]将电路分割成多个秘密子电路分开执行，使每个秘密子电路中的毛刺只能泄漏一个秘密共享因子的信息. 这类方案被称作多方电路实现[14]. 而在软件实现中，每一个指令都占用了一个或数个时钟周期，并且根据掩码方案的算法，单条指令中不可能同时包含不同秘密共享因子. 从这个角度来说，软件实现本身就可以被看作是一种多方电路实现. 这使得掩码方案的软件实现天然地满足电路毛刺下的安全性. 因此现有的可证明安全高阶掩码方案普遍将软件实现作为首选.

为了让高阶分析方法[20-24]的计算复杂度维持在一个可行的水平，确定出有用的特征点组合(或减小包含相关功耗的窗口大小)是非常重要的. 对于一个 n 阶掩码方案，最朴素的特征点选取方法是在整条功耗曲线上遍历任意 $n+1$ 个功耗点的组合，然后判断组合功耗是否包含秘密信息. 很明显，这种方法的渐进时间复杂度是功耗曲线样本点数 N 的 $n+1$ 次方. 实际的掩码方案通常都会使算法运行时间增加，从而使样本点数增多. 因此这种特征点选取方法实际上很难应用. Oswald 等人[25]在实际实施 2 阶 DPA 时先通过经验选取了一个样本窗口，包含了目标操作所在的大致区域，从而避免了在整条功耗曲线上遍历. Reparaz 等人[16]提出了一种基于 MIA 寻找 $n+1$ 个特征点的方法，但这种方法本身的时间复杂度是 N 的 $n+1$ 次方水平. Durvaux 等人[26]提出了一种基于仿射追踪的方法，能够在特征点选取的同时对特征功耗进行降维处理. 但这种方法同样具有 N^{n+1} 水平的渐进时间复杂度，并且由于这种方法计算结果的收敛具有不确定性，因此难以应用于更为高阶的情形. 综合上述我们可以知道，目前的特征点选取方法的时间复杂度水平都是功耗曲线或时间样本窗口中的样本点数的 $n+1$ 次方，在分析阶数较高的掩码方案时存在困难.

虽然掩码方案的软件实现可以看作是一种可以对抗电路毛刺泄漏的多方电路，但在软件实现中，每次执行的指令并没有被隐藏. 很显然，算法中每一个敏感变量都会被确定的指令进行操作. 因此，这些指令的位置其实就表明了敏感变量出现的位置. 基于这个思想，本文提出了基于指令识别的特征点选取方法来提高特征点定位的准确性，并且降低特征点选取过程的时间复杂度.

指令识别技术已经有了一些应用，例如嵌入式程序的逆向工程，代码完整性验证，伪造检测，恶意软件检测和攻击预备[27]等. 其中有一些分析方法利用了侧信道信息，但就我们所知，目前尚未出现在高阶分析的特征点选取中应用指令识别的方法. 本文提出了基于指令定位的特征点选取方法来准确地定位出掩码方案中每个秘密共享因子. 指令定位方法需要的功耗曲线量很少，计算复杂度与掩码方案的阶数无关. 本文随后对一些常用指令的定位实验和

对 Coron14 方案的攻击实验实际验证了新方法的有效性.本文方案使得高阶功耗分析的时间复杂度不再和掩码阶数相关,这导致现有高阶掩码方案的实际安全性需要被重新考虑.另外,基于指令定位的特征点选取方法适用于不同芯片及软件平台,可以作为针对掩码方案软件实现的高阶功耗分析的第一步,从而能够和各种高效的高阶功耗分析结合使用,具有很高的通用性.

本文第 2 节分析高阶掩码方案的软件实现中可能存在的安全漏洞;第 3 节中详细描述基于指令定位的特征点选取方法和高阶分析方法,并且验证新的特征点选取方法的有效性;第 4 节用新方法分析 Coron14 方案;最后我们在第 5 节总结本文的工作.

2 高阶掩码方案的软件实现

2.1 软件实现中的安全问题分析

现有可证明安全的高阶掩码方案对于硬件电路实现来说资源消耗过大.同时,Mangard 等人[28]发现掩码方案的硬件实现容易出现电路毛刺造成泄露.除此之外,还有一些安全漏洞威胁着掩码方案硬件实现,比如并行执行或传播延迟等.而与硬件相比,软件实现并不会受到这些因素的影响.因此,现有多数高阶掩码方案[3-5,7-8]都选择了软件实现.

然而,由于现有软件平台上的一些性能优化技术,掩码方案软件实现也可能会出现安全漏洞.在这些技术中,最有可能造成安全泄漏的性能优化技术是指令并行执行和数据寄存器复用.我们接下来分别对这两种技术可能造成的泄漏进行分析.

(1)在现在的通用 CPU 架构中,超标量流水[29]和指令乱序执行[30]被广泛使用,以提高 CPU 的执行效率.指令流水线实现了一定程度的并行处理,被称为指令并行.指令并行使指令不再完全串行执行,而是在前一条指令完成之前就开始执行下一条指令.这种技术要求每个指令都被分为多个阶段.不同指令的不同阶段可以并行执行,这就使得前一条指令在执行的过程中,下一条指令可以进行预取,达到提高效率的目的.为了减少指令流水中的阻塞现象,一些 CPU 会使用指令乱序执行的技术.这一系列技术使得 CPU 中出现了多条指令并行执行的情况.并行执行可能导致某一瞬间的功耗值与多个中间变量值产生相关性.以 3 个 8 bits 秘密共享因子为例,并行执行产生的功耗泄漏与 3 个秘密共享因子异或值之间的互信息为 $I[HW(x_1)+HW(x_2)+HW(x_3);x_1\oplus x_2\oplus x_3]=2.5442$ bits①.因此,当出现几个共享因子的相关指令并行执行的情况时,一个软件平台上实现的高阶掩码方案就会出现高阶泄漏.然而幸运的是,这种高阶泄漏的出现需要满足两个条件.首先需要这些共享因子相关的指令之间不能有数据依赖关系.其次这些指令在汇编代码中相邻或相距很近.而在汇编代码中,读取数据的指令前后通常会有很多内存地址计算、指令跳转和条件判断等等与数据无关的指令.这就使得这一条件在实际的软件实现方案中很难出现.因此指令并行造成的高阶泄漏在目前的高阶掩码方案软件实现中很难发生.

(2)当掩码阶数增加时,很难想象掩码方案不会复用一些操作来降低资源消耗.在软件平台上,由于通用 CPU 只有有限的数据寄存器,因此一定会出现某个寄存器会在不同时刻依次存储多个秘密共享因子.如果某个寄存器出现连续存储秘密共享因子的情况,那么这个寄存器的翻转功耗就会泄漏这两个秘密共享因子的异或值,造成高阶泄漏.然而寄存器复用至多只能泄漏两个变量的联立信息.由于现有可证明安全的高阶掩码方案[31]满足秘密共享因子数量 $n>2d$,因此软件实现中出现的寄存器复用并不能使 d 个功耗点的联立功耗泄漏敏感变量的任何信息.

根据上面的分析,虽然指令并行和寄存器复用策略在理论上有可能会造成高阶泄漏.但是由于这两种安全隐患需要的条件很苛刻,因此在现有的可证明安全高阶掩码方案中,尚未发现这样的高阶泄漏使其安全阶数降低的情况.因此在本文中,基于指令定位的高阶分析方法假定现有的高阶掩码方案软件实现没有出现上述高阶泄漏的情况.也就是说对于一个 d 阶掩码方案的软件实现,只有 $d+1$ 阶攻击才能成功.

2.2 软件实现中的指令

掩码方案的主要思想是向原始加密算法中引入随机数据.但并没有通过随机数据来隐藏加密算法中的操作.软件实现中的固定指令就会使掩码方案面临着新的安全威胁.通常来说,一个指令的执行包

① 我们遍历了 x_1,x_2,x_3 的所有可能取值,分别计算两个随机变量 $X=HW(x_1)+HW(x_2)+HW(x_3)$ 和 $Y=x_1\oplus x_2\oplus x_3$ 的取值.根据这两个随机变量的取值,可以计算出 X 和 Y 之间的互信息取值为 2.5442 bits.

括 5 个步骤：指令预取，指令译码和寄存器预取，执行，访问存储，以及寄存器写回. 在指令预取过程中，指令码会被载入到指令寄存器. 因此，指令预取过程会泄漏指令码的功耗数据. 我们利用这一泄漏去直接恢复每一个指令的类型，构成指令执行序列，再对照掩码方案的汇编代码序列，就可以确定每个敏感变量所在的位置. 这就避免了遍历 d 个功耗点的组合. 也就使特征点定位的复杂度变为曲线样本点数的线性关系，并且与掩码阶数无关. 这一原理为本文提出的基于指令识别的特征点选取方法奠定了基础.

3　基于指令定位的高阶侧信道分析

3.1　符号说明

本节在介绍基于指令定位的侧信道分析方法的细节前，首先进行符号描述. 我们将功耗曲线集合用 L 表示，包含的曲线总量记为 N. 曲线集合 L 中的第 i 条功耗曲线记为 L_i，其中 $i\in[1,N]$. 每条曲线包含 M_P 个功耗样本点. 曲线 L_i 上第 t 个样本点记为 $L_i(t)$，其中 $t\in[1,M_P]$. 我们将功耗曲线 L_i 上从样本 $L_i(a)$ 到 $L_i(b)$ 的一段向量记为 $L_i[a,b](a<b)$. 为了提高对指令的建模和匹配的效率及成功率，需要首先使用降维算法来压缩功耗曲线. 本文主要使用的降维算法为主成分分析(Principal Component Analysis，PCA). 曲线 L_i 经过降维得到的功耗曲线记为 L_i'. 这个降维过程记为 $L_i'=\mathrm{DR}(L_i)$.

随后对降维后的曲线进行功耗建模的过程与文献[32]中构造模板 q 的方法相似. 在模板 q 中，均值向量记为 $\boldsymbol{Q}$，协方差矩阵记为 $\boldsymbol{C}$. 在匹配过程中，一条新得到的功耗曲线 $\boldsymbol{x}$ 是否符合模板 q 需要通过计算后验概率 $p(q|\boldsymbol{x})$ 来验证. 该后验概率的计算公式如下：

$$p(q|\boldsymbol{x}) = p(\boldsymbol{x}|q)\Pr(q) \tag{1}$$

其中 $\Pr(q)$ 表示 q 的概率，$p(\boldsymbol{x}|q)$ 满足如下公式：

$$p(\boldsymbol{x}|q)=\mathcal{N}(\boldsymbol{x}|\boldsymbol{Q},\boldsymbol{C})$$
$$=\frac{1}{\sqrt{(2\pi)^D|\boldsymbol{C}|}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{Q})^{\mathrm{T}}\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{Q})\right) \tag{2}$$

其中 D 是 $\boldsymbol{x}$ 的样本点数. 越大的后验概率说明假设密钥正确的可能性越大. 由于 $\Pr(q)$ 是一个正实数，因此在模板匹配过程中只需要计算和比较不同假设密钥的 $p(\boldsymbol{x}|q)$ 即可.

3.2　基于指令定位的高阶分析方法流程

3.2.1　指令定位

本节对基于指令定位的特征点选取方法(Instruction Recognition-based Points of Interest Selection，IR-PoIS)进行详细描述. IR-PoIS 方法的第一步是选择需要定位的目标指令. 目标指令是指那些对秘密共享因子进行操作的指令，这通常包含了数据运算指令和数据存取指令. 在布尔掩码方案中，与共享因子相关的运算指令只有异或和逻辑与运算，而逻辑与运算在内存地址计算中非常常见，这就使得测试者难以从众多逻辑与运算执行的位置中精确定位出秘密共享因子所在的时钟周期. 因此逻辑与运算是不推荐作为目标指令的. 那么剩下的目标指令就只有异或和数据存取指令了. 由于数据存取指令一定包含了所有共享因子. 因此我们一般选择数据存取指令作为目标指令. 我们将整个程序中目标指令出现的次数记为 m，共享因子个数记为 n. 这些指令出现的次数和在整个指令序列中的位置可以从掩码方案的汇编代码中得到.

为了建立指令的模板，首先需要采集建模功耗曲线. 采集这些曲线时，为了方便采集，我们将指令固定，操作数随机. 另外，为了降低指令流水线的影响，需要随机化目标指令执行前后的指令. 我们将建模功耗曲线集合记为 T，其中的一条曲线记为 T_i，其中 $i\in[1,N_T]$. 集合 T 中每条曲线的样本点数量记为 M_T. 类似的，我们将匹配时采集的功耗曲线集合记为 P，其中的曲线记为 $P_i(i\in[1,N_P])$. 每条曲线上有 M_P 个样本点数.

模板建立完成后，我要对攻击目标功耗进行模板匹配. 整个指令定位的过程如算法 1 所示. 由于噪声的影响，单次匹配可能会出现偏差. 为了提高指令模板匹配的准确性，算法 1 在进行模板匹配之前首先设置了一个概率阈值 t. 随后的模板匹配过程选出所有匹配概率高于 t 的位置(数量大于 LDD 的数量 m)作为候选位置. 然后构造一个集合 I，记录功耗曲线上每一个位置被选为候选位置的次数. 对于每一条匹配曲线 $P_i(i\in[1,N_P])$，重复这一过程，并对集合 I 中的次数进行累加，最后得到的集合 I 中就记录了经过多次匹配后每个位置被选为候选位置的总次数. 我们将集合 I 某个位置 k 被选中的次数记为 I_k. 最后，算法选择出现次数最多的 n 个位置作为最后的结果输出. 输出结果集合记为 $\mathbb{L}$. 这种提高定位准确性的思路与 Ou 等人[33]提出的 FIPS 方法思路类似，常用于模板攻击中，可以大大降低单条功耗曲线对模板匹配结果的偶然性影响. 应用在算法 1 中可以提高指令定位的健壮性，同时降低模板匹配过程中所需要的功耗曲线量.

算法 1. 指令定位算法(IR-PoIS).

输入:M_T,M_P,t,$T_i(i\in[1,N_T])$,$P_i(i\in[1,N_P])$

输出:$\mathbb{L}$

1. $T'=\mathrm{DR}(T)$; //对每一条曲线 T_i 进行降维
2. $(\boldsymbol{Q},\boldsymbol{C})\leftarrow$buildTemplate($T'$); //利用 T' 建立模板
3. FOR $i=1$ to N_P
4. FOR $k=1$ to M_P-M_T
5. $P'_k=\mathrm{DR}(P[k,k+M_T])$;
6. $\mathrm{Pr}_k=\mathcal{N}(P'_k|\boldsymbol{Q},\boldsymbol{C})$; //根据式(2)计算概率
7. END FOR
8. IF $\mathrm{Pr}_k>t$ THEN
9. $I_k=I_k+1$; //k 被选出的次数增加 1
10. END IF
11. END FOR
12. $\mathbb{L}\leftarrow\{k|I_k$ is the top m in $I\}$; //选出 I 中被选中次数最多的前 m 个位置的下标 k
13. RETURN $\mathbb{L}$;

算法 1 中的第 1~2 行完成了目标指令的建模过程.这个过程的时间复杂度只依赖于 N_T,而与 M_T 和 M_P 无关.因此,随着 M_P 的增加,建模过程的时间复杂度可以看成是一个常数.指令模板的建立过程通常只需要采集数千条功耗曲线即可完成[27],随后的模板匹配过程则只需要扫描一遍匹配功耗曲线就可以完成匹配.整个指令定位过程的时间复杂度为 $\mathcal{O}(M_P)$.因此,指令定位方法要比现有的特征点定位方法(基于穷举的定位方法[16,25]或依赖于上下文的定位方法[26])高效得多.表 1 中总结并对比了现有的特征点定位方法和指令定位方法的时间复杂度.

表 1 现有不同特征点定位方法的时间复杂度对比(d 是高阶攻击的阶数,W_{len} 是滑动窗口的大小,M_P 是攻击曲线上的样本总点数)

特征点定位方法	所需时间估计	时间复杂度
文献[25]中的方法	$\binom{M_P}{d}$	$\mathcal{O}((M_P)^d)$
文献[16]中的方法	$\binom{M_P}{d}$	$\mathcal{O}((M_P)^d)$
文献[26]中的方法	$\binom{M_P/W_{len}}{d}$	$\mathcal{O}\left(\left(\frac{M_P}{W_{len}}\right)^d\right)$
指令定位方法	M_P-M_T	$\mathcal{O}(M_P)$

3.2.2 密钥恢复

这一小节详细介绍恢复秘密信息(例如密钥)的方法.在完成特征点定位以后,攻击者得到了与不同秘密共享因子相关的多个时钟周期.显然这些共享因子对应的功耗需要联立才能恢复出秘密信息.因此攻击者需要选择一个合适的功耗联立函数来计算联立功耗.我们将用于密钥恢复的功耗曲线记为 $X_i(i\in[1,N_X])$,将与第 j 个共享因子相关的功耗曲线段记为 $X_i[k_j](k_j\in\mathbb{L},j\in[1,n])$.高阶分析中通常选择的功耗联立函数是这些功耗曲线段的正规积[24],其计算公式如下:

$$X_i^*=\prod_{j\in[1,n]}(X_i[k_j]-E(X_i[k_j])) \tag{3}$$

其中,$E(X_i[k_j])$是功耗曲线段 $X_i[k_j](i\in[1,N_X])$的数学期望.得到联立功耗后,只需要对联立功耗实施经典的 DPA 或 CPA 攻击即可恢复出原始秘密信息(如子密钥 key).整个密钥恢复过程如算法 2 所示.

算法 2. 密钥恢复算法.

输入:$\mathbb{L}$,$X_i(i\in[1,N_X])$, plaintext

输出:key

1. FOR $j=1$ to n
2. $E_j=\frac{1}{N_X}\sum_{i\in[1,N_X]}X_i[k_j]$;
3. END FOR
4. FOR $i=1$ to N_X
5. $X_i^*=\prod_{j\in[1,n]}(X_i[k_j]-E_j)$;
6. END FOR
7. key$\leftarrow$Attack(plaintext,X^*); //对功耗数据 X^* 进行 DPA 或 CPA 攻击
8. RETURN key;

3.3 基于指令定位的高阶侧信道分析方法的有效性分析

本节分析基于指令定位的高阶侧信道分析方法的有效性及其影响因素.该方法的有效性主要依赖于两个因素:一个是特征点选取阶段的准确率,另一个是联立功耗和秘密信息之间的相关性.

为了验证 IR-PoIS 方法的有效性,我们编写了一个含有 10 个 LDD 指令的测试程序.之所以选择这个指令,是因为在高阶掩码方案中,所有秘密共享因子都会通过 LDD 指令从内存读取到寄存器中.因此只要定位出 LDD 指令,就可以找出每个共享因子出现的位置.为了方便起见,我们让测试程序中每个 LDD 指令的前后指令都相同并且固定.这样可以降低指令流水线的影响.测试程序汇编代码如图 1 所示.

测试程序的功耗数据采集所使用的示波器型号为 Agilent DSO-X 3034A,采样率为 200 MSa/s.采集功耗数据时使用的功耗放大器型号为 PA303(3 dB,0~3 GHz).测试程序的运行平台为 SASEBO-W 开发板上的一张 ATMega-163 智能卡.这组实验一共采集了 20000 条功耗曲线.其中,10000 条用于建立指令模板,另外 10000 条功耗曲线用于测试指

```
164:          state->s[1] = state->s[0] + state->s[0];
+0000039C:    8188        LDD         R24,Y+0         Load indirect with displacement
+0000039D:    0F88        LSL         R24             Logical Shift Left
+0000039E:    8389        STD         Y+1,R24         Store indirect with displacement
166:          __asm__ __volatile__(
+0000039F:    0000        NOP                         No operation
+000003A0:    0000        NOP                         No operation
+000003A1:    0000        NOP                         No operation
+000003A2:    0000        NOP                         No operation
+000003A3:    0000        NOP                         No operation
+000003A4:    8188        LDD         R24,Y+0         Load indirect with displacement
+000003A5:    0000        NOP                         No operation
+000003A6:    0F88        LSL         R24             Logical Shift Left
+000003A7:    0000        NOP                         No operation
+000003A8:    8389        STD         Y+1,R24         Store indirect with displacement
+000003A9:    0000        NOP                         No operation
+000003AA:    8188        LDD         R24,Y+0         Load indirect with displacement
+000003AB:    0000        NOP                         No operation
+000003AC:    0F88        LSL         R24             Logical Shift Left
+000003AD:    0000        NOP                         No operation
+000003AE:    8389        STD         Y+1,R24         Store indirect with displacement
+000003AF:    0000        NOP                         No operation
+000003B0:    8188        LDD         R24,Y+0         Load indirect with displacement
+000003B1:    0000        NOP                         No operation
+000003B2:    0F88        LSL         R24             Logical Shift Left
+000003B3:    0000        NOP                         No operation
+000003B4:    8389        STD         Y+1,R24         Store indirect with displacement
+000003B5:    0000        NOP                         No operation
+000003B6:    8188        LDD         R24,Y+0         Load indirect with displacement
+000003B7:    0000        NOP                         No operation
+000003B8:    0F88        LSL         R24             Logical Shift Left
+000003B9:    0000        NOP                         No operation
+000003BA:    8389        STD         Y+1,R24         Store indirect with displacement
+000003BB:    0000        NOP                         No operation
```

图 1　测试程序的汇编代码

令定位方法实际的准确率. 我们将建模曲线和测试曲线分别记作 T_i ($i \in [1, 10\,000]$), X_j ($j \in [1, 10\,000]$).

首先,对 LDD 指令的功耗进行建模. 由于 LDD 指令是一个双周期指令,因此 LDD 指令的功耗会持续两个时钟周期. 我们使用了 PCA 算法对这组功耗曲线进行了降维处理. 在这次实验中,PCA 算法输出的维数为 8 维. 然后,我们按照算法 1 的流程对 X_j ($j \in [1, 10\,000]$) 执行 IR-PoIS 方法. 我们使用不同大小的候选集合$\mathbb{L}$进行了两次实验,一次是 20 个候选位置,一次是 10 个候选位置. 正确位置出现在候选集合$\mathbb{L}$中的个数如图 2 所示.

从图 2 中可以看出,当候选集合大小定为 LDD 指令总数(10 个)时,8 个正确位置出现在了候选集合中. 而如果选择一个更大的候选集合(例如,20 个元素),那么所有正确位置都会出现在候选集合中. 同时,只需要对 1000 条功耗曲线进行匹配,就可以确定出这些正确位置. 因此,如果候选集合大小选择合适,IR-PoIS 方法定位成功所需的功耗数据量是很小的.

然而,对 STD 指令进行同样的实验却出现了

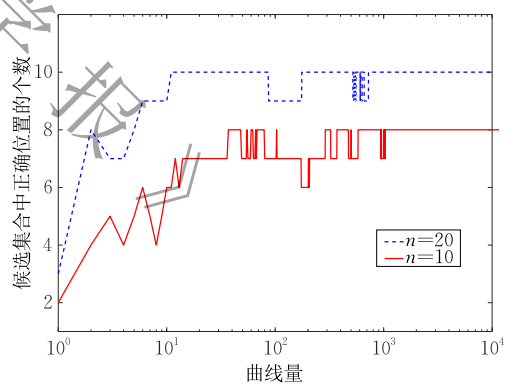


图 2　候选集合$\mathbb{L}$中出现的正确位置个数
(图中的 n 是算法 1 中$\mathbb{L}$的大小)

很低的准确率. 经过对比分析可以发现,采样率会很大程度上影响 IR-PoIS 方法的准确性. 这是因为采样率太低会使功耗曲线中丢失一部分指令的信息.

为了对比采样率对 IR-PoIS 方法有效性的影响,我们使用两种采样率进行了数据采集,采样率分别是 200 MSa/s 和 2 GSa/s. 然后我们将两种条件下 10 个 STD 指令所在的功耗区域分别截取出来进行对比,对比结果如图 3 和图 4.

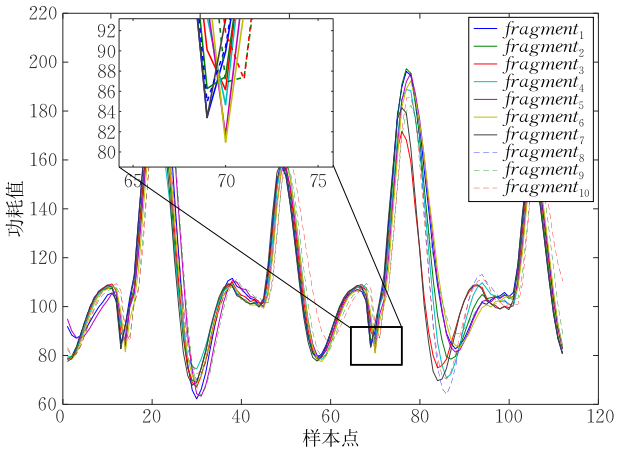


图 3 采样率为 200 MSa/s 时，10 个STD指令所在功耗区域对比（第 i 个 STD 指令对应的功耗区域记为 $fragment_i$）

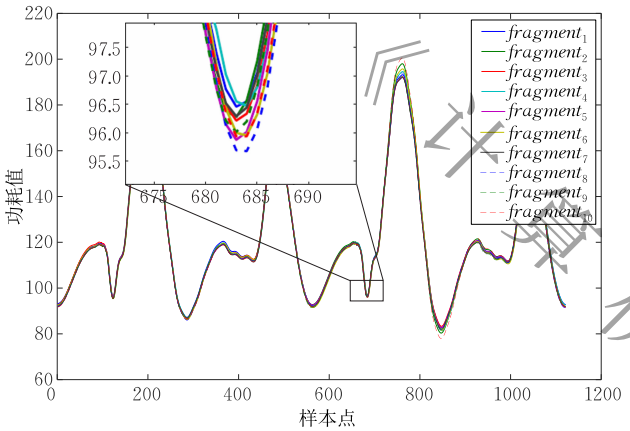


图 4 采样率为 2 GSa/s 时，10 个 STD 指令所在功耗区域对比（第 i 个 STD 指令对应的功耗区域记为 $fragment_i$）

从图 3 和图 4 的对比可以明显地看出，第 2，3，9 和 10 个 STD 指令对应的功耗曲线段 $fragment_2$，$fragment_3$，$fragment_9$，$fragment_{10}$ 在采样率较低的情况下丢失了一部分信息. 我们分别对两种采样率下采集得到的功耗曲线进行了指令定位实验，最终的定位结果如图 5 所示.

图 5 展示了在两种采样率下采集到的功耗曲线上进行指令定位实验的结果，横轴代表所用的功耗曲线条数，纵轴为候选位置集合中正确位置的个数. 从图 5 可以看出，在采样率为 200 MSa/s 的功耗曲线上进行指令定位时，即使候选集合的大小是正确位置总数的两倍(20 个)，依然只有 3 个正确位置出现在候选集合中. 而当采样率提升到 2 GSa/s 时，这 10 个正确位置就全部出现在候选集合中了. 并且在较高采样率条件下，指令定位方法只需要数十条功耗曲线就能成功定位出全部正确位置. 因此在候选集合大小和功耗曲线采样率适当的条件下，本文提

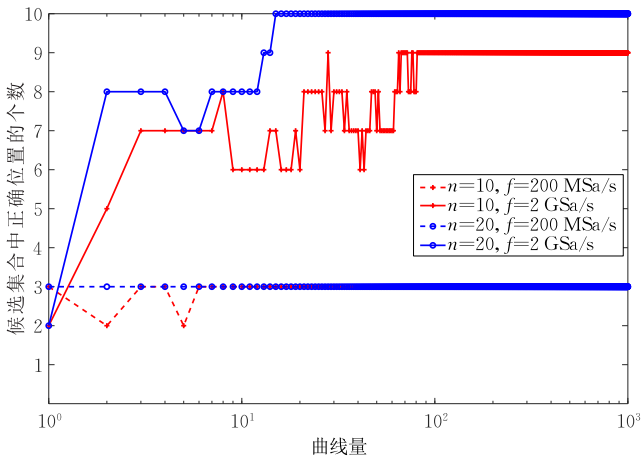


图 5 不同采样率条件下候选集合中出现的正确位置个数比较（图中的 n 是候选集合的大小，f 表示采样率）

出的指令定位方法是非常高效的. 在实际分析过程中，如果发现某个目标指令的模板匹配效果不佳，那么可以尝试提高采样率或者更换目标指令的方法来提高模板匹配效率.

4 针对 Coron14 方案的高阶分析实验

4.1 Coron14 高阶掩码方案

本节详细描述基于指令定位的高阶分析方法分析任意阶掩码方案软件实现的过程. 这类掩码方案的理论安全性已经在 ISW 安全框架下得到了证明. Coron14 高阶掩码方案是第一个基于查找表重构的任意阶掩码方案. 本文以 Coron14 高阶掩码方案为例进行说明. 该方案中安全计算 S 盒的部分如图 6 所示.

首先要确定指令定位的目标数据. 针对这种高阶掩码方案，我们选定的目标为 Coron14 算法中第 12 行的 RefreshMasks 算法的输入. 如果能够恢复出这里的 n 个输入共享因子，那么就可以成功恢复出 S 盒的输出 $x=\bigoplus_{i=0}^{n-1} x[i]$. 我们根据 Coron14 方案的作者提供的 C 语言源代码，在 ARMv7 芯片上使用 GCC 进行了编译，得到了 Coron14 方案的汇编代码. 其中，RefreshMasks 算法的汇编代码如图 7 所示.

从图 7 的右半部分可以明显地看出，每个输入共享因子 $x[i]$都会由 LDD 指令读取到 CPU 中. 只要能够定位出全部 LDD 指令的位置，就可以得到每个 $x[i]$相关的功耗曲线区域，也就能够恢复出秘密信息.

RefreshMasks

Input: $z_1, z_2, \cdots, z_n$ $(z = z_1 \oplus z_2 \oplus \cdots \oplus z_n)$
Output: $z_1, z_2, \cdots, z_n$ $(z = z_1 \oplus z_2 \oplus \cdots \oplus z_n)$

1. for i=2 to n do
2. tmp=$rand()$
3. z_1=$z_1 \oplus tmp$
4. z_i=$z_i \oplus tmp$
5. end for
6. return $z_1, z_2, \cdots, z_n$

Coron14 Masking Scheme

Input: $x_1, x_2, \cdots, x_n$ $(x = x_1 \oplus x_2 \oplus \cdots \oplus x_n)$
Output: $y_1, y_2, \cdots, y_n$ $(S(x) = y_1 \oplus y_2 \oplus \cdots \oplus y_n)$

1. for $u \in \{0,1\}^k$ do
2. $T(u) = (S(u), 0, \cdots, 0) \in (\{0,1\}^{k'})^n$
3. end for
4. for $i \in [1, n-1]$ do
5. for $u \in [0, 2^k - 1]$ do
6. for $j \in [1, n]$ do $T'(u)[j] = T(u \oplus x_i)[j]$;
7. end for
8. for $u \in [0, 2^k - 1]$ do
9. $T(u)$=RefreshMasks($T'(u)$)
10. end for
11. end for
12. $(y_1, y_2, \cdots, y_n)$=RefreshMasks($T(x_n)$)
13. return $y_1, y_2, \cdots, y_n$;

图 6　Coron14 高阶掩码方案流程

```
/*C code*/

void RefreshMasks(byte x[], int n)
{
    for (int i=1; i<n; ++i)
    {
        byte tmp=rand();
        x[0]=x[0]^tmp;
        x[i]=x[i]^tmp;
    }
}
```

```
/*segment of assembly code*/

LDD    R24, Z+0     Load indirect with displacement
EOR    R24, R22     Exclusive OR
STD    Z+0, R24     Store indirect with displacement

LDD    R24, Y+1     Load indirect with displacement
EOR    R24, R22     Exclusive OR
STD    Y+1, R24     Store indirect with displacement

/*x[i] (i=1, ···, n-1) is stored into R24*/
```

图 7　RefreshMasks 算法 C 语言源码和对应的汇编代码

为了验证指令定位方法的实际效果，我们在 SASEBO-W 开发板上采集了 3 阶 Coron14 方案的功耗曲线. 该掩码方案程序运行在智能卡上，所使用的 CPU 为 8 位 ATmega163 Atmel AVR 处理器，时钟频率为 3.5712 MHz. 采集功耗曲线时的采样率为 200 MSa/s. 后面的两个小节中将详细介绍指令定位和高阶分析的实际效果.

4.2　指令定位实验

如前一小节中的分析，指令定位的目标是所有与 RefreshMasks 算法输入共享因子对应的 LDD 指令位置. 然而，在实际算法中，很可能会出现一些与输入共享因子无关的 LDD 指令. 而能否正确定位出目标 LDD 指令的位置会直接影响到指令定位后续的高阶分析方法的分析效率. 因此我们采取了一些手段来提高指令定位的准确性.

(1) 对原始功耗曲线进行 PCA 等降维处理，以提高模板建立的效率和模板匹配的准确性.

(2) 利用 LDD 指令执行过程中多个周期的功耗数据. 一条指令执行可能需要几个时钟周期，例如 LDD 指令执行就需要 2 个时钟周期才能完成. 为了提高模板匹配的成功率，我们使用了 LDD 指令执行期间所有的时钟周期进行建模以提高指令定位准确性.

(3) 对连续若干条指令进行指令建模. 可以将目标 LDD 指令前后的若干条指令合并起来一起完成指令建模和指令匹配过程，以区分出无关的 LDD 指令. 在实际攻击中，如果攻击者可以得到掩码方案的源码和实现平台，那么他就可以确定目标 LDD 指令前后的指令序列.

指令模板匹配过程的结果如图 8 所示. 图 8 中显示了每个位置正确的概率，而圆圈出现的位置就是正确位置. 可以看出正确位置得到的匹配概率都远大于错误位置的匹配概率，因此指令定位方法是有效的. 为了评估指令定位方法的效率，我们统计了不同功耗曲线数量条件下得到的指令定位成功率，如图 9 所示.

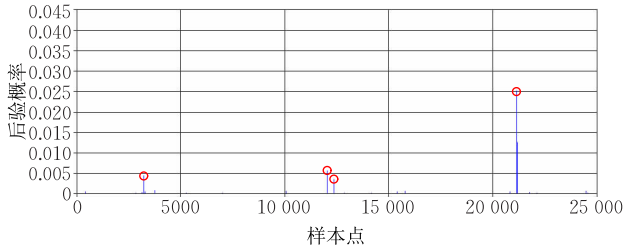


图 8　指令定位得到的每个位置匹配成功的概率
(后验概率是根据式(2)计算出的条件概率)

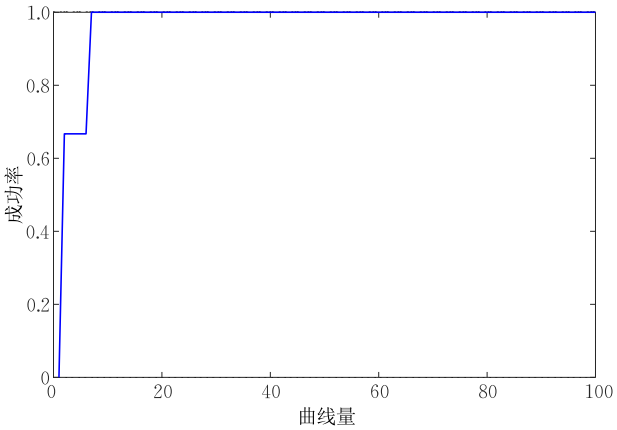


图 9 指令定位方法的成功率

从图 9 可以看出，只需要不到 10 条功耗曲线，指令定位方法就可以达到 100%的成功率. 另外，指令定位方法的时间和数据复杂度并不会随掩码阶数的提高而增加. 因此这种分析方法很容易应用在针对高阶掩码方案实际的攻击中.

4.3 高阶侧信道分析实验

在得到了每个共享因子对应的 LDD 指令位置后，可以很容易地利用联立函数对这些功耗曲线段进行高阶分析，进而恢复出秘密信息$\oplus_{i=0}^{n-1} x[i]$. 在 ATmega163 处理器中，LDD 指令执行的第二个时钟周期会包含输入共享因子的值. 我们联立这些功耗曲线段，并进行了 DPA 分析. 图 10 中显示了正确的猜测子密钥对应的差分功耗曲线. 从图中可以看出差分功耗曲线上有明显的尖峰.

随后，我们分别应用了 DPA 和 CPA 来验证高阶分析的实际效果. 我们共进行了 125 组攻击实验，每次实验应用了 5 万条功耗曲线. 我们将多组实验结果进行了统计，得到的 DPA 和 CPA 攻击成功率和子密钥猜测熵分别如图 11 和图 12 所示. 从图 11 和图 12 中可以看出，通过对 5 万条功耗曲线进行分析，高阶分析的成功率达到了 100%，最终得到子密钥的猜测熵取值为 1，说明正确子密钥在全部子密钥中排在了第一位. 这也证实了指令定位得到的位置确实是正确的. 因此，这一系列的实验说明指令定

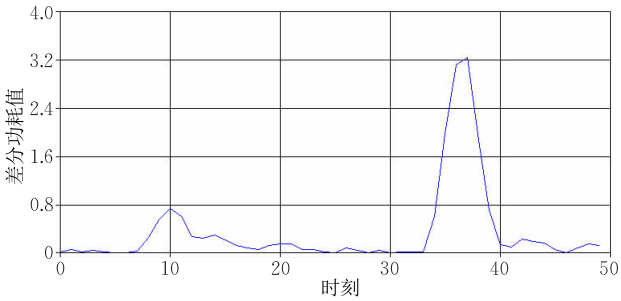
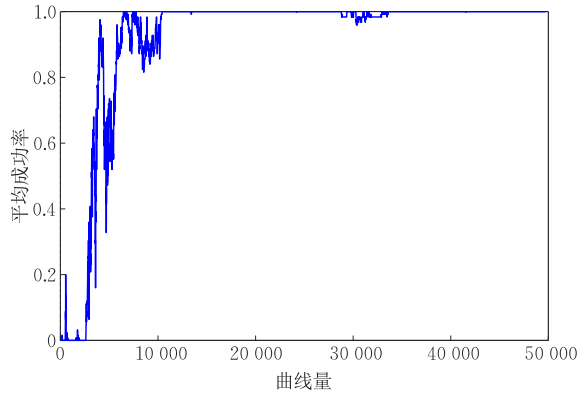
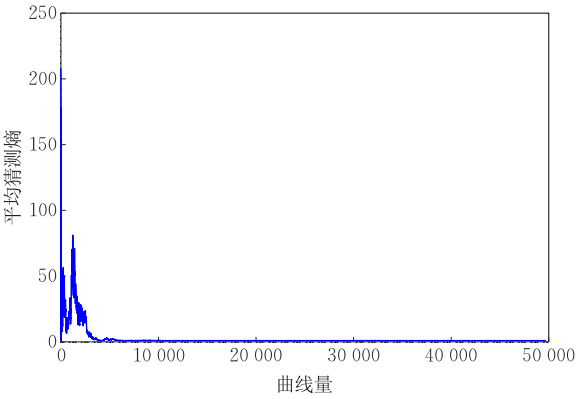


图 10 正确子密钥对应的差分功耗曲线尖峰

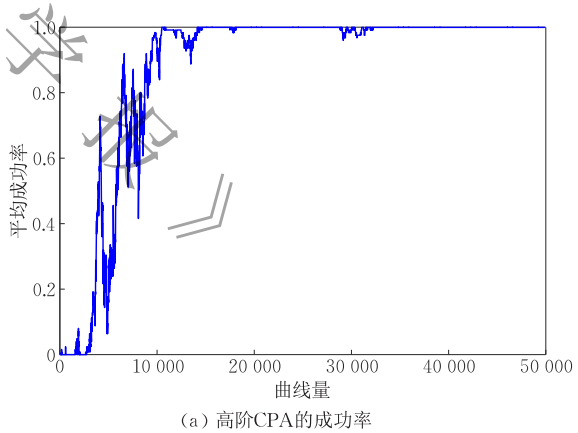


(a) 高阶DPA的成功率

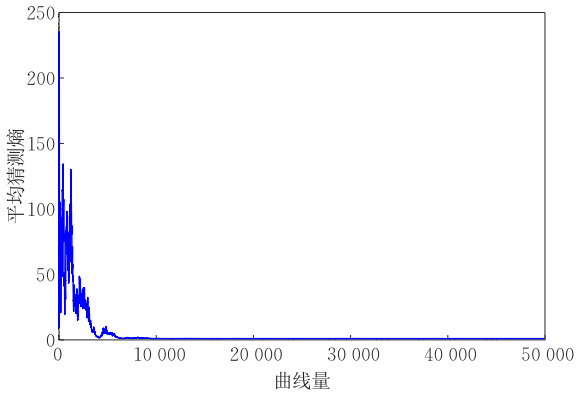


(b) 子密钥猜测熵

图 11 高阶 DPA 的成功率和子密钥猜测熵



(a) 高阶CPA的成功率



(b) 子密钥猜测熵

图 12 高阶 CPA 的成功率和子密钥猜测熵

位方法是一种有效的特征点选取方法，而基于指令定位的高阶功耗分析方法也是很高效的.

5 结　论

ISW 安全框架下的高阶掩码被认为是满足理论安全性的侧信道防护方法. 软件平台是现有的高阶掩码方案最常用的实现平台. 在软件平台上实现可以避免硬件电路中毛刺产生的安全泄漏，并且可以利用软件平台上丰富的存储资源和运算资源.

针对高阶掩码方案，高阶功耗分析通常是比较有效的一类分析方法. 而要成功攻击高阶掩码方案，需要更为高阶的功耗分析. 由于高阶分析方法具有较高的时间和数据复杂度，在阶数较高的情况下，这种分析方法通常需要很长一段时间来采集数据以及实施攻击. 然而在实际的芯片侧信道安全测试过程中，测试人员只有有限的测试时间. 因此高阶分析的效率严重影响了芯片侧信道分析安全性的测试和评估效果.

在本文中，我们提出了新的基于指令定位的特征点定位方法. 这种方法可以将现有高阶分析方法中的特征点定位过程的时间复杂度从功耗曲线样本点数的 n 次复杂度降至线性复杂度. 为了验证指令定位的特征点选择方法的实际效果和效率，本文对测试程序中的 LDD 指令进行了指令定位实验. 实验结果证实本文提出的指令定位方法仅需数十条功耗曲线即可完成特征点定位，成功率接近 100%. 在对其他指令的定位实验中，我们发现功耗曲线的采样率是影响指令定位方法效果的一个重要因素. 采样率不够高会导致功耗曲线中的特征点出现丢失，导致指令模板匹配失败. 在提高了采样率后，指令定位的成功率确实得到了提高. 随后，我们对 Coron14 方案的软件实现进行了实际的攻击实验，其源码是公开的①. 实验结果证实了指令定位方法确实可以以较低的时间和数据复杂度定位出目标指令的位置. 本文提出的指令定位方法的时间复杂度是线性的，和高阶分析的阶数无关.

指令定位方法表明在现有的掩码防护方案软件实现中，只有数据被掩码所保护，而程序中的操作并没有被隐藏. 攻击者可以建立敏感指令的模板，通过对指令的定位完成对敏感信息的定位，进而完成高阶攻击. 基于这一发现，设计者需要重新检视现有高阶功耗分析方法的复杂度和现有高阶掩码方案的安全性. 如何在软件平台上保护高阶掩码方案的安全性将作为我们未来的研究工作之一.

参 考 文 献

[1] Mangard S, Oswald E, Popp T. Power Analysis Attacks: Revealing the Secrets of Smart Cards vol. 31. New York, USA: Springer Science & Business Media, 2008

[2] Rivain M, Dottax E, Prouff E. Block ciphers implementations provably secure against second order side channel analysis// Proceedings of the International Workshop on Fast Software Encryption. Lausanne, Switzerland, 2008: 127-143

[3] Kim H, Hong S, Lim J. A fast and provably secure higher-order masking of AES S-box//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Nara, Japan, 2011: 95-107

[4] Carlet C, Goubin L, Prouff E, et al. Higher-order masking schemes for S-boxes//Proceedings of the Fast Software Encryption. Washington, USA, 2012: 366-384

[5] Rivain M, Prouff E. Provably secure higher-order masking of AES//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Santa Barbara, USA, 2010: 413-427

[6] Roy A, Vivek S. Analysis and improvement of the generic higher-order masking scheme of FSE 2012//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Santa Barbara, USA, 2013: 417-434

[7] Coron J-S, Prouff E, Rivain M, Roche T. Higher-order side channel security and mask refreshing//Proceedings of the International Workshop on Fast Software Encryption. Singapore, 2013: 410-424

[8] Coron J-S. Higher order masking of look-up tables// Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Copenhagen, Denmark, 2014: 441-458

[9] Grosso V, Prouff E, Standaert F-X. Efficient masked S-boxes processing—A step forward//Proceedings of the International Conference on Cryptology in Africa. Dakar, Senegal, 2014: 251-266

[10] Chari S, Jutla C S, Rao J R, Rohatgi P. Towards sound approaches to counteract power-analysis attacks//Proceedings of the Annual International Cryptology Conference. Santa Barbara, USA, 1999: 398-412

[11] Genelle L, Prouff E, Quisquater M. Thwarting higher-order side channel analysis with additive and multiplicative maskings// Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Nara, Japan, 2011: 240-255

[12] von Willich M. A technique with an information-theoretic basis for protecting secret data from differential power attacks//Proceedings of the IMA International Conference on Cryptography and Coding. Cirencester, UK, 2001: 44-62

① https://github.com/coron/htable/

[13] Goubin L, Martinelli A. Protecting AES with Shamir's secret sharing scheme//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Nara, Japan, 2011: 79-94

[14] Prouff E, Roche T. Higher-order glitches free implementation of the AES using secure multi-party computation protocols//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Nara, Japan, 2011: 63-78

[15] Balasch J, Faust S, Gierlichs B. Inner product masking revisited//Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Sofia, Bulgaria, 2015: 486-510

[16] Reparaz O, Gierlichs B, Verbauwhede I. Selecting time samples for multivariate DPA attacks//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Leuven, Belgium, 2012: 155-174

[17] Prouff E, Rivain M, Bevan R. Statistical analysis of second order differential power analysis. IEEE Transactions on Computers, 2009, 58: 799-811

[18] Veyrat-Charvillon N, Standaert F-X. Generic side-channel distinguishers: Improvements and limitations//Proceedings of the Annual Cryptology Conference. Santa Barbara, USA, 2011: 354-372

[19] Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation //Proceedings of the 20th Annual ACM Symposium on Theory of Computing. Chicago, USA, 1988: 1-10

[20] Waddle J, Wagner D. Towards efficient second-order power analysis//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Cambridge, USA, 2004: 1-15

[21] Moradi A, Mischke O, Eisenbarth T. Correlation-enhanced power analysis collision attack//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Santa Barbara, USA, 2010: 125-139

[22] Clavier C, Feix B, Gagnerot G, et al. Improved collision-correlation power analysis on first order protected AES//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Nara, Japan, 2011: 49-62

[23] Dabosville G, Doget J, Prouff E. A new second-order side channel attack based on linear regression. IEEE Transactions on Computers, 2013, 62: 1629-1640

[24] Gierlichs B, Batina L, Preneel B, Verbauwhede I. Revisiting higher-order DPA attacks//Proceedings of the Cryptographers' Track at the RSA Conference. San Francisco, USA, 2010: 221-234

[25] Oswald E, Mangard S, Herbst C, Tillich S. Practical second-order DPA attacks for masked smart card implementations of block ciphers//Proceedings of the Cryptographers' Track at the RSA Conference. San Jose, USA, 2006: 192-207

[26] Durvaux F, Standaert F-X, Veyrat-Charvillon N, et al. Efficient selection of time samples for higher-order DPA with projection pursuits//Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design. Berlin, Germany, 2015: 34-50

[27] Msgna M, Markantonakis K, Mayes K. Precise instruction-level side channel profiling of embedded processors//Proceedings of the International Conference on Information Security Practice and Experience. Fuzhou, China, 2014: 129-143

[28] Mangard S, Popp T, Gammel B M. Side-channel leakage of masked CMOS gates//Proceedings of the Cryptographers' Track at the RSA Conference. San Francisco, USA, 2005: 351-365

[29] Smith J E, Sohi G S. The microarchitecture of superscalar processors. Proceedings of the IEEE, 1995, 83: 1609-1624

[30] Hwu W M, Patt Y N. HPSm, a high performance restricted data flow architecture having minimal functionality//Proceedings of the ACM SIGARCH Computer Architecture News. New York, USA, 1986: 297-306

[31] Ishai Y, Sahai A, Wagner D. Private circuits: Securing hardware against probing attacks//Proceedings of the Annual International Cryptology Conference. Santa Barbara, USA, 2003: 463-481

[32] Chari S, Rao J R, Rohatgi P. Template attacks//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Hotel Sofitel, San Francisco, USA, 2002: 13-28

[33] Ou C, Wang Z, Ai J, et al. Error tolerance based single interesting point side channel CPA distinguisher//Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. Xi'an, China, 2016: 819-827

GUO Zhi-Peng, Ph. D. candidate. His current research interests include side channel analysis and protection and cryptanalysis.

TANG Ming, Ph. D., professor. Her current research interests include cryptography, and systematic research on side channel analysis and protection.

HU Xiao-Bo, senior engineer. Her research interests include cryptographic design and cryptographic chip.

LI Yu-Guang, M. S. candidate. His research interests include side channel analysis and cryptanalysis.

PENG Guo-Jun, Ph. D., professor, Ph. D. supervisor. His current research interests include network and information system security.

ZHANG Huan-Guo, professor, Ph. D. supervisor. His current research interests include information security and trusted computing.

Background

Masking is an effective and popular countermeasure to several types of side-channel attacks. As for the higher-order masking case, it tends to be more and more generic to resist any order SCAs. Although the ISW model proposed by Ishai, Sahai and Wagner could prove the security of masking schemes theoretically, there remains a gap between the practical security and theoretical security. This paper aims at improving the security of masking implementation, researches the risks of software design with recognition of instructions. a new PoI selection method based on instruction recognition called IR-PoIS to decrease the time complexity of the existing POI methods from n-time polynomial to linear in the number of points of a power trace. The IR-PoIS method verified that the instructions executed in software are not hidden by the existing maskings, and an adversary can build templates of the sensitive instructions at first and successfully complete a higher order analysis. Our analysis shows that the risk occurs in the design of provably security masking scheme. This capability can help designers directly modify their designs to make them more secure.

Our research team has successively used power, electromagnetic and fault analysis different structure of block ciphers in SCA for 10 years. We completed a prototype system about side-channel analysis, protection and testing.

This research is supported by the National Natural Science Foundation of China (Nos. 61472292 and 61332019), Opening Topics of State Key Laboratory of Cryptology (No. MMKFKT201821), Research on Side Channel Analysis and Detection Technology in Cryptographic Chip Protection Design (No. 2018J-10) and in part by the Technological Innovation of Hubei Province (Major Special Project) (No. 2018AAA046).