一种基于模乘相等检测的标量乘碰撞攻击方法

韩绪仓[1,2],曹伟琼[1*],陈 华[1],李昊远[1]
(1. 中国科学院软件研究所可信计算与信息保障实验室,北京 100190;2. 中国科学院大学,北京 100049)

摘 要: 碰撞攻击是针对椭圆曲线密码的主要分析技术之一,其关键取决于对点加、倍点碰撞检测的正确率.由于随机操作数和分支语句的影响,对点加、倍点的碰撞检测几近于随机猜测,因而如何对点加、倍点进行碰撞检测成为亟需解决的问题.本文以Weierstrass曲线中基于雅可比坐标的点加、倍点为分析对象,提出了基于模乘相等的标量乘碰撞攻击方法.首先,结合点加和倍点的运算流程,从中识别出了有助于碰撞检测的模乘,并在模乘间构造了新的碰撞关系,将攻击转化为模乘碰撞检测.其次,本文发现在雅可比坐标中存在由坐标Z完全决定的模乘,基于此首次提出了模乘相等检测,将攻击转化为判断模乘两个操作数是否相同,从而避免了随机操作数的影响.最后,本文对一款硬件实现芯片进行碰撞检测实验,通过对曲线基于主成分分析进行压缩处理,将点加和倍点碰撞检测的准确率提高到了99%.本文提出的碰撞检测方法对采用了随机掩码和分支平衡的标量乘实现仍有效.

关键词: 标量乘;雅可比坐标;模乘碰撞检测;模乘相等检测
基金项目: 国家自然科学基金(No.62172395)
中图分类号: TN918;TP309 **文献标识码:** A **文章编号:** 0372-2112(2024)11-3865-12
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20230795

A Collision Detection Method Based Similarity Detection of Modular Multiplication on Scalar Multiplication

HAN Xu-cang[1,2], CAO Wei-qiong[1*], CHEN Hua[1], LI Hao-yuan[1]
(1. *Trusted Computing and Information Assurance Laboratory*, *Institute of Software*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*;
2. *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

Abstract: Collision attack is one of the main analysis techniques for scalar multiplication, and its success rate depends on the correction rate of collision detection in operations such as point addition and multiplication. Due to the influence of random operands and branching statements, collision detection almost approaches random guessing. How to detect collisions for point addition and point doubling effectively has become an urgent problem to be solved. To solve this problem, we focus on point addition and doubling on Jacobian coordinates in Weierstrass curves, and propose a collision detection method for scalar multiplication based on modular similarity detection. Firstly, according to the operation process of point addition and point doubling, the modular multiplication used in collision detection are identified, and a new collision relationship is constructed between the modular multiplications, which converts attack into modular multiplication collision detection. Secondly, we find that there are modular multiplications which are completely determined by the coordinate Z in the Jacobi coordinates. With the help of this finding, we propose modular similarity detection, and convert attack into detecting whether the two modular multiplication operations are the same, thereby avoiding the influence of random operands on the collision detection. Finally, we conduct collision detection experiments on a hardware-implemented scale multiplication. By compressing the curve based on principal component analysis, the accuracy of collision detection for point addition and doubling is improved to 99%. The proposed collision detection method remains effective for scalar multiplications with masking and branch balancing measures.

Key words: scalar multiplication; Jacobi coordinates; modular multiplication collision detection; modular similarity detection

Foundation Item(s): National Natural Science Foundation of China (No.62172395)

收稿日期:2023-08-10;修回日期:2024-02-18;责任编辑:李勇锋
*通讯作者:曹伟琼

1 引言

椭圆曲线密码[1](Elliptic Curve Cryptosystem,ECC)是一种主流的公钥密码算法,具有同等安全强度下密钥长度短、计算速度快、存储空间小等优点.我国于2012年发布了SM2椭圆曲线公钥密码算法[2],已应用于智能卡、安全芯片、加密机等密码产品中.因而,研究针对ECC实现的安全分析方法具有重要的现实意义和需求.

标量乘kG是ECC中的核心运算,其实现易受侧信道分析的威胁[3,4],通过分析算法运行过程中泄露的功耗、电磁等信息进行以恢复密钥.如利用点加和倍点对应不同的功耗特征进行简单功耗分析[5](Simple Power Analysis,SPA),基于中间数据与功耗曲线之间相关性进行差分功耗分析[6](Differential Power Analysis,DPA),通过输入特殊点$(0,y)$或$(x,0)$的修正功耗分析[7]及零值功耗分析[8].此外,对于SM2签名或ECDSA等签名算法的标量乘实现,若由侧信道分析仅能得到部分标量比特,那么可基于格基约减技术恢复出整个密钥.如利用模板攻击恢复了部分nonce比特[9],或将ECDSA算法中模加运算中的分支语句的执行转化为关于nonce的不等式[10],或根据标量乘运行时间的不同确定nonce的若干比特的攻击[11,12].

针对侧信道分析已经形成了多种针对性的防护方法.首先,采用分支平衡的标量乘实现方法以隐藏功耗泄露,如基于原子化实现方式[13]使用同一种方式实现点加、倍点两种运算;采用执行路径与密钥比特无关的标量乘算法结构,如在蒙哥马利阶梯算法[14](Montgomery Power Ladder,MPL)或基于随机点初始化的BRIP[15]等算法,对于每个标量比特都固定执行倍点和点加.此外,还可采用掩码防护[3]掩盖中间数据与功耗曲线之间的关联.如对输入点进行掩码防护可保证与选择特殊点输入的攻击无法实施,坐标随机化进一步增加了DPA等攻击的难度,而标量掩码在很大程度上将攻击限定在单条曲线上,无法实施如DPA等垂直侧信道分析.这些掩码防护方法已广泛应用于标量乘的安全实现中,可有效抵抗常规的垂直侧信道分析,但并不能抵抗水平侧信道分析[16].

水平碰撞攻击[16]主要利用两个操作是否有相同操作数(即碰撞检测)进行分析.该攻击不需要假定功耗模型或预测中间数据,因此可用于分析采用了掩码和分支平衡的标量乘实现.Hanley等[17]首先论证了水平碰撞攻击的可行性,只要攻击者可对相关操作进行碰撞检测,那么就能够恢复出密钥.之后,Jin等[18]利用读取预存点的功耗泄露进行碰撞攻击,成功获取ECDSA中nonce的部分比特值;Nascimento等[19]针对蒙哥马利阶梯算法中的条件交换操作成功实施了攻击;Jin等[20]对利用碰撞攻击对固定基的comb算法进行了分析.然而这些攻击针对的操作只存在于特定实现中,无法扩展为对标量乘实现中的点加和倍点的有效碰撞检测.

Abarzua等[21]总结了近年来碰撞检测的研究成果,得出结论"由于点加和倍点为不同的运算,因此并不能直接对二者进行碰撞检测".尽管如此,近些年仍出现了针对某些特定点加、倍点实现的碰撞攻击研究.Bauer等[16]等提出了对Edwards曲线下的三种统一点加倍点实现的碰撞攻击,将碰撞检测转化为两个模乘的碰撞检测.Murdica等[22]提出了相同中间值攻击,通过选择特殊点作为输入使得在点加和倍点运算内的某些特定中间值固定,以突破掩码的影响,但该法对于采用了点随机化防护的实现[3]并不可行.Weierstrass曲线是目前应用最广泛的椭圆曲线,也是SM2和ECDSA所基于的曲线类型.然而针对该曲线中的标量乘运算,目前缺少利用其倍点和点加进行有效的碰撞攻击方法.针对该问题,本文则考虑将Weierstrass曲线中的点加、倍点碰撞转化为模乘碰撞检测问题.

除了碰撞检测的转化研究外,如何度量两个模乘对应功耗曲线的相似程度是碰撞攻击另一关键点.目前的研究大都是针对RSA模幂运算中的模乘,主要的度量方法包括相关系数、聚类分类以及基于深度学习的分类等.如Bauer等[16]通过计算两个模乘对应曲线的相关系数作为度量指标.Heyszl等[23]则利用聚类技术对模乘对应曲线进行分类,同一类曲线对应的模乘则有相同的操作数;而Perin等[24]则提出了更完整的基于聚类的分析框架;Zaid等[25]则提出了一种新的损失函数,通过融合多个模型预测结果进行自集成;Saito等[26]则对RSA-CRT模幂对应的功耗曲线,从中识别出伪操作.这些度量方式同样可以扩展应用于ECC的标量乘的碰撞检测中.

针对上述问题,本文以基于Weierstrass曲线的标量乘实现作为分析对象,分析在最常用的雅可比坐标下的实现特性,提出了对点加、倍点转化为模乘相等的有效碰撞攻击方法,并以一款芯片中的标量乘掩码实现作为实验对象,结合深度学习中常用的主成分分析(Principal Component Analysis,PCA)技术,验证了攻击方法的有效性.该方法具有以下优点:

(1)提出了在雅克比坐标下通用标量乘中点加、倍点转化为模乘相等的碰撞检测新方法.该方法打破所有常见的随机掩码和分支平衡影响.相比以往对特殊曲线的特定转换,该方法的适用性更强.

(2)基于模乘相等的碰撞检测具有更高的准确率.雅克比坐标下的点加、倍点中存在仅由坐标Z决定的特殊模乘,而该模乘不受随机操作数的影响,碰撞检测难度更低.本文在实验中采用PCA对功耗曲线进行预处

理,有效提高了攻击的成功率,点加、倍点中不受随机数影响的特殊模乘碰撞检测的正确率可高达99%.

2 基础知识

2.1 椭圆曲线运算

令 F_p 为有限域,其中 p 为大于3的素数. 定义椭圆曲线 $E(F_p)$ 为满足 Weierstrass 方程 $y^2=x^3+ax+b \bmod p$ 的所有点 (x,y) 以及无穷远点 ∞ 的集合[27].

令 P_1 和 P_2 为椭圆曲线中的点. 对于 $P_3=P_1+P_2$,若 $P_1=P_2$,称其为倍点;若 $P_1\neq P_2$,则称其为点加. 定义 $kP=P+\cdots+P$ 为 k 个 P 相加的运算,称为标量乘. 若 n 为满足 $nP=\infty$ 的最小非零正整数,定义 n 为点 P 的阶.

为了避免求逆运算,通常在投影坐标下计算点加和倍点,其中在雅克比坐标下点被表示为 (X,Y,Z). 同一个点对应不同的雅克比投影坐标表示,如 (X,Y,Z) 和 (Xr^2,Yr^3,Zr) 表示同一个点,其中 $r\neq 0$.

令 $P_i=(X_i,Y_i,Z_i)$,其中 $i=1,2,3$. 雅可比坐标下的点加运算 $P_3=P_1+P_2$ 为

$$\begin{cases} X_3=u^2-v^3-2X_1Z_2^2v^2 \\ Y_3=u\left(X_1Z_2^2v^2-X_3\right)-Y_1Z_2^3v^2 \\ Z_3=vZ_1Z_2 \end{cases} \tag{1}$$

其中 $u=Y_2Z_1^3-Y_1Z_2^3$,$v=X_2Z_1^2-X_1Z_2^2$. 雅可比坐标下的倍点 $P_3=2P_1$ 为

$$\begin{cases} X_3=\left(3X_1^2-3Z_1^4\right)^2-8X_1Y_1^2 \\ Y_3=\left(3X_1^2-3Z_1^4\right)\left(4X_1Y_1^2-X_3\right)-8Y_1^4 \\ Z_3=2Y_1Z_1 \end{cases} \tag{2}$$

表1和表2分别为点加和倍点的一种典型实现流程,将点价和倍点转化为模乘、模加和模减等运算. 易知,点加包括17次模乘运算,倍点包括8次模乘运算.

表1　雅可比坐标下的点加运算

序号	运算表达式	序号	运算表达式
1	$T_1=Z_1Z_1$	13	$T_6=T_1+T_3$
2	$T_2=T_1Z_1$	14	$T_7=T_5T_6$
3	$T_1=T_1X_2$	15	$T_8=T_3T_5$
4	$T_2=T_2Y_2$	16	$T_9=T_5T_1$
5	$T_3=Z_2Z_2$	17	$T_{10}=T_4T_9$
6	$T_4=T_3Z_2$	18	$T_2=T_2-T_4$
7	$T_3=T_3X_1$	19	$T_3=T_2T_2$
8	$T_4=T_4Y_1$	20	$X_3=T_3-T_7$
9	$T_1=T_1-T_2$	21	$T_5=T_5-X_3$
10	$Z_3=Z_1Z_2$	22	$T_4=T_2T_5$
11	$Z_3=T_1Z_3$	23	$Y_3=T_4-T_{10}$
12	$T_5=T_1T_1$	—	—

表2　雅可比坐标下的倍点运算

序号	运算表达式	序号	运算表达式
1	$T_1=Z_1Z_1$	10	$T_4=T_4+T_5$
2	$T_2=X_1+T_1$	11	$T_5=T_1X_1$
3	$T_3=X_1-T_1$	12	$T_3=T_5+T_5$
4	$T_4=T_2T_3$	13	$Y_3=T_4T_4$
5	$T_5=T_4+T_4$	14	$X_3=Y_3-T_3$
6	$T_1=Y_1+Y_1$	15	$T_3=T_1T_2$
7	$Z_3=T_1Z_1$	16	$T_2=T_5-X_3$
8	$T_2=T_1Y_1$	17	$T_5=T_4T_2$
9	$T_1=T_2+T_2$	18	$Y_3=T_5-T_3$

2.2 标量乘实现

标量乘可以被转化为点加和倍点. 如对于经典的二进制算法,当密钥比特为0时,仅执行倍点运算;而当密钥比特为1时,则执行倍点和点加两个运算. 攻击则通过区分点加和倍点,进而反推出对应的密钥比特.

为了抵抗此类侧信道分析,标量乘采用执行路径与密钥比特无关的实现算法,即点加和倍点的执行顺序与密钥比特无关,从而消除执行操作与密钥之间的关联. 算法1[5]为基于总加结构的标量乘算法,通过引入伪运算在每个循环内固定执行倍点和点加,是一种典型的分支平衡实现算法.

算法1　Double-And-Add-Always(**总加标量乘算法**)

输入: $k=\left(k_{l-1},\cdots,k_1,k_0\right),P$

输出: kP

1. $Q[0]=P$
2. $i=l-1$ to 0,重复执行
   $Q[0]=2Q[0]$
   $Q\left[1-k_i\right]=Q[0]+P$
3. 返回 $Q[0]$

目前已经提出了大量的分支平衡实现算法,如 Montgomery Power Ladder[14],基于随机点初始化的 BRIP[15],基于查找表的算法[18]等. 此外,掩码则通过将中间数据随机化,以抵抗如DPA等利用中间数据与功耗曲线之间统计特性的分析方法. 典型掩码防护包括[3]:

(1)标量掩码:对 k 进行随机化,典型方式如 $k'=k+r*n$,其中 r 为随机数.

(2)点掩码:用 $P'=P+R$ 代替 P 参与运算,其中 R 为随机点.

(3)坐标随机化:将点 $P(x,y)$ 由仿射坐标转为雅克比坐标 (xr^2,yr^3,r),其中 $r\neq 0$ 为随机数.

2.3 碰撞攻击

对于操作 $O(\mathrm{in}_1,\mathrm{in}_2,\cdots,\mathrm{in}_l)$,令 $\mathrm{op}(O)=\{\mathrm{in}_1,\mathrm{in}_2,\cdots,\mathrm{in}_l\}$ 表示该操作的操作数集合. 对于操作

O_1和O_2,若存在操作数$\mathrm{in}\in \mathrm{op}(O_1)$和$\mathrm{in}'\in \mathrm{op}(O_2)$满足$\mathrm{in}=\mathrm{in}'$,则称$O_1$和$O_2$关于in(或in′)碰撞;反之则称$O_1$和$O_2$无关.定义碰撞检测函数:

$$c(O_1,O_2)=\begin{cases}0, & O_1和O_2碰撞\\ 1, & O_1和O_2无关\end{cases} \tag{3}$$

侧信道分析的对象为功耗曲线或电磁曲线等,其可用向量表示.对于曲线$\boldsymbol{t}_1$和$\boldsymbol{t}_2$,定义碰撞系数$\rho(\boldsymbol{t}_1,\boldsymbol{t}_2)$,以度量两条曲线之间的相似程度.通常可基于相关性、欧式距离等指标定义碰撞系数ρ.

Bauer等[16]提出了对标量乘实现的水平碰撞攻击,分为碰撞操作识别和碰撞检测两步.令k_j表示标量k的第j比特.在碰撞操作识别阶段,对于每一个密钥比特k_j,从标量乘实现中识别出关键操作O_1和O_2,使得利用二者碰撞检测结果可以推断出k_j,从而将攻击转化为O_1和O_2的碰撞检测.在碰撞检测阶段,则从标量乘对应的功耗曲线中识别出O_1和O_2对应的功耗曲线段,记为$\boldsymbol{t}_1$和$\boldsymbol{t}_2$;然后计算功耗曲线之间的碰撞系数$\rho(\boldsymbol{t}_1,\boldsymbol{t}_2)$,确定碰撞检测的结果$c(O_1,O_2)$并反推出密钥$k_j$,如图1所示.

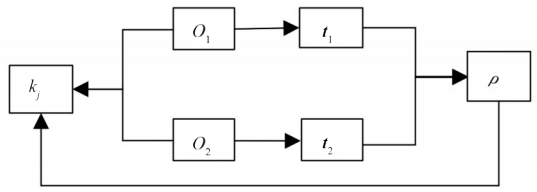


图1 碰撞攻击示意图

攻击者由碰撞系数的大小反推密钥比特,由此碰撞检测的正确率决定了可恢复出密钥比特的比例.根据两个操作的碰撞与否将ρ分为两类:$\rho_i=\{\rho|c(O_1,O_2)=i\}$,其中$i=0,1$.$\rho_0$对应于两个操作碰撞时碰撞系数的分布;$\rho_1$对应于两个操作无关时碰撞系数的分布.定义:$\Delta_\rho=|u_0-u_1|$,其中$u_0$为$\rho_0$的均值,$u_1$为$\rho_1$的均值.$\Delta_\rho$越大,那么碰撞和无关两种情况下的碰撞系数的差别越大,碰撞检测的正确率越高.

3 基于模乘相等的碰撞检测方法

水平碰撞攻击对于采用了掩码和分支平衡等防护的标量乘实现仍然适用,其核心问题是如何对点加、倍点等进行碰撞检测.针对目前缺乏有效的点加、倍点碰撞检测方法,本章提出了一种通用碰撞检测方法,将攻击转化为基于模乘相等检测;然后将该碰撞检测方法应用于基于总加实现的标量乘中以证明该碰撞检测方法的有效性.

3.1 点加倍点碰撞检测

本节以雅可比坐标下的点加和倍点运算为分析对象,从中识别出不受随机数影响的模乘,将点加、倍点的碰撞检测转化为模乘碰撞检测和模乘相等检测两类问题.首先令$MM(R,S,N)=R\times S \bmod N$表示模乘,其中$N$表示模数,$R$和$S$为模乘输入.对于两个模乘$MM(R_1,S_1,N)$和$MM(R_2,S_2,N)$,定义:

模乘碰撞检测:检测$R_1=R_2$是否成立,其中S_1和S_2服从均匀分布.

模乘相等检测:判断$R_1=R_2$和$S_1=S_2$是否成立.

3.1.1 标量乘碰撞分析

点加和倍点是标量乘中的主要运算.由于两个的倍点碰撞在标量乘分析中较少出现,因此通常碰撞攻击主要考虑倍点和点加、点加和点加两类碰撞检测.点加和倍点的操作数为椭圆曲线中的点,在雅可比投影坐标下表示为(X,Y,Z).考虑到椭圆曲线中点在投影坐标下的表示并不唯一,本文将讨论的场景限定为:当点加、倍点碰撞时,碰撞点对应的三个坐标都分别相等.只要坐标不相同,即使其对应同一个点,也将其视为不同的操作数.

碰撞检测是通过计算两个操作(如点加、倍点操作)对应功耗曲线的碰撞系数,以判断二者碰撞与否,即是否存在相同操作数.注意到,点加$P_4=P_1+P_2$包括两个操作数P_1和P_2,记与碰撞检测相关的操作数为碰撞操作数P_c,与碰撞检测无关的操作数则记为无关操作数P_r.在大多数情况中,P_r为标量乘运算的中间结果,通常服从均匀分布.本文分析发现,即使当点加和倍点碰撞时,P_r的随机性将使得点加对应于不同的功耗特征.

如表1所示,点加在第9步计算$T_1-T_3 \bmod p$,其中$T_1=X_2Z_1^2$,$T_2=X_1Z_2^2$,模数为p.该步执行过程对应如下分支语句:

$$\begin{cases}T_1-T_3, & 若T_1\geqslant T_3\\ T_1+p-T_3, & 若T_1<T_3\end{cases} \tag{4}$$

当$T_1<T_3$时,需额外计算一次加法.注意到,T_1和T_3分别由点加的两个输入P_1和P_2共同决定,因此执行哪个分支也由两个输入共同决定.即使点加中与碰撞攻击相关的操作数被固定,那么点加仍可能执行不同的分支,进而产生出不同的功耗特征.类似的,在点加的其它步骤(如模加、模减和模乘)中也存在分支语句,使得点加对应的功耗特征与碰撞与否无关.

综上,无关操作数所具有的不确定性及对分支语句执行的影响,将使得碰撞时两个操作对应的功耗特征可能完全不同,进而使得碰撞检测几近于随机猜测.这是针对标量乘掩码实现碰撞攻击中需要解决的通用问题.为此,本文提出将点加、倍点的碰撞检测转化为模乘相等的碰撞检测,以避免无关操作数和分支语句

对碰撞检测的影响.

3.1.2 点加、倍点碰撞检测

本节通过对点加和倍点中每步运算的操作数进行分析,从中识别出可用于替代点加、倍点进行碰撞检测的子操作,排除无关子操作对碰撞检测的影响,从而提高碰撞检测的正确率.

雅可比坐标下的点加运算为$P_4=P_1+P_2$,其中$P_i=(X_i,Y_i,Z_i)$,$i=1,2,3$. 根据与碰撞相关操作数的不同,点加运算可以分为两类:若碰撞操作数为$P_c=P_1$,那么记该点加为ADD_1;若$P_c=P_2$,那么记该点加为ADD_2. 雅可比坐标下的倍点运算$P_5=2P_3$,其中$P_3=(X_3,Y_3,Z_3)$,记为D.

表1为一种典型的点加实现方式,每一步都可视为点加的子操作. 对于每一步运算,若其操作数或部分比特可由操作数P_c决定,那么该子操作可用于代替点加进行碰撞检测. 反之,该子操作则与碰撞检测无关. 对表1中每一步对应的操作数进行分析,其与P_1相关的子操作为序号1、2、3、4、7、8、10对应的模乘,如表3所示,其均可用于代替点加ADD_1进行碰撞检测. 为了便于对照,表3和表4中的序号为该操作在表1中的序号.

表3　与点加ADD_1相关的子操作

序号	子操作	操作数1	操作数2
1	$T_1=Z_1Z_1$	Z_1	Z_1
2	$T_2=T_1Z_1$	Z_1^2	Z_1
3	$T_1=T_1X_2$	Z_1^2	X_2
4	$T_2=T_2Y_2$	Z_1^3	Y_2
7	$T_3=T_3X_1$	Z_2^2	X_1
8	$T_4=T_4Y_1$	Z_2^3	Y_1
10	$Z_3=Z_1Z_2$	Z_1	Z_2

类似的,与操作数P_2相关的子操作为序号3、4、5、6、7、8、10对应的模乘,如表4所示,其均可用于代替点加ADD_2进行碰撞检测.

表4　与点加ADD_2相关的子操作

序号	子操作	操作数1	操作数2
3	$T_1=T_1X_2$	Z_1^2	X_2
4	$T_2=T_2Y_2$	Z_1^3	Y_2
5	$T_3=Z_2Z_2$	Z_2	Z_2
6	$T_4=T_3Z_2$	Z_2^2	Z_2
7	$T_3=T_3X_1$	Z_2^2	X_1
8	$T_4=T_4Y_1$	Z_2^3	Y_1
10	$Z_3=Z_1Z_2$	Z_1	Z_2

表2为一种典型的雅可比坐标下的倍点运算流程. 由于倍点只有一个操作数P_3,因此每一个子操作的操作数均由P_3决定. 通过分析每个子操作的操作数与P_3之间的关系,并不能从中找到真正有助于碰撞检测的子操作. 本文中与倍点碰撞的操作仅可能使点加,而点加中与碰撞检测相关的子操作均为模乘运算. 因此只需在表2中找到与表3和表4所列模乘相碰撞的模乘,即以X_3、Y_3、Z_3、Z_3^2和Z_3^3为操作数的模乘,如表5所示. 说明:表5中的序号为该模乘在表2中对应的序号.

表5　与倍点相关的子操作

序号	子操作	操作数1	操作数2
1	$T_1=Z_3Z_3$	Z_3	Z_3
7	$Z_5=T_1Z_3$	$2Y_3$	Z_3
8	$T_2=T_1Y_3$	$2Y_3$	Y_3
11	$T_2=T_1Y_3$	$4Y_3^2$	X_3

尽管在雅可比坐标下点加和倍点的具体实现细节可能与表1和表2有所区别,但上述子操作识别的方法仍然有效. 此外,在实现中可能还会存在一些隐藏的子操作,如读取碰撞操作数P_c. 尽管此类操作也可用于碰撞检测中,但由于其并不具有通用性,因此本文不再讨论.

3.1.3 模乘碰撞检测

本节考虑在模乘间构造新的碰撞检测关系,将点加、倍点的碰撞检测转化为模乘碰撞检测. 考虑一般情况,O_1和O_2为碰撞检测针对的操作,即点加和倍点中的一种;$O_{1,a}$和$O_{2,b}$为从O_1和O_2中识别出的关键模乘. 令prob(x)表示概率计算函数,若:

$$\mathrm{prob}(c(O_1,O_2)=c(O_{1,a},O_{2,b}))\to 1 \tag{5}$$

那么即可利用模乘碰撞检测结果推断点加和倍点的碰撞检测结果. 基于模乘碰撞检测的流程如图2所示.

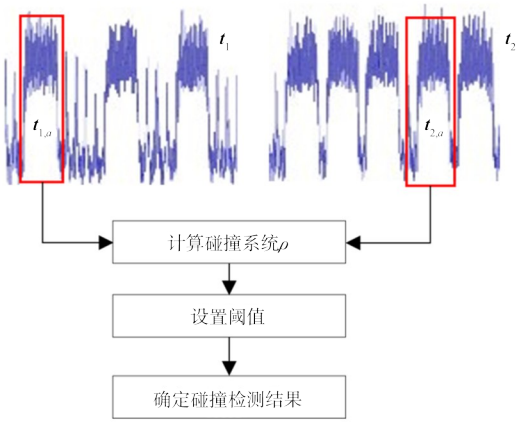


图2　基于模乘碰撞检测的分析

令$\boldsymbol{t}_1$和$\boldsymbol{t}_2$分别为O_1和O_2对应的曲线;$\boldsymbol{t}_{1,a}$和$\boldsymbol{t}_{2,b}$分别为$O_{1,a}$和$O_{2,b}$对应的曲线. 攻击主要分为两步:首先从$\boldsymbol{t}_1$中提取出$O_{1,a}$对应的曲线$\boldsymbol{t}_{1,a}$,从$\boldsymbol{t}_2$中提取出$O_{2,b}$对应的曲线$\boldsymbol{t}_{2,b}$;然后计算碰撞系数$\rho(\boldsymbol{t}_{2,b},\boldsymbol{t}_{2,b})$;最后设置合

适的阈值确定出模乘碰撞检测结果$c(O_{1,a},O_{2,b})$,进而由式(5)可推导出$c(O_1,O_2)$.

下面将该分析方法应用于点加、倍点碰撞检测中.令$P_c=(X,Y,Z)$表示碰撞点,那么有

$$(X,Y,Z)=\begin{cases}(X_1,Y_1,Z_1), & \text{碰撞操作为}\mathrm{ADD}_1\\(X_2,Y_2,Z_2), & \text{碰撞操作为}\mathrm{ADD}_2\\(X_3,Y_3,Z_3), & \text{碰撞操作为D}\end{cases} \tag{6}$$

从表3、表4和表5中识别的模乘可能以X、Y、Z、Z^2和Z^3为操作数.根据操作数的不同,将这些模乘(为方便起见,同时列出了对应的序号)重新排列,如表6所示.

表6 点加、倍点中的模乘碰撞构造

操作数	碰撞检测针对的操作		
	ADD_1	ADD_2	D
X	$7.Z_2^2X$	$3.Z_1^2X$	$11.(4Y_3^2)X$
Y	$8.Z_2^3Y$	$4.Z_1^3Y$	$8.(2Y_3)Y$
Z	$1.ZZ$ $2.Z^2Z$ $10.Z_2Z$	$5.ZZ$ $6.Z^2Z$ $10.Z_1Z$	$1.ZZ$ $7.(2Y_3)Z$
Z^2	$3.Z^2X_2$ $2.Z^2Z$	$7.Z^2X_1$ $6.Z^2Z$	—
Z^3	$4.Z^3Y_2$	$8.Z^3Y_1$	—

当从表6的同一行选择对应的模乘时,那么可将点加、倍点的碰撞检测转化为模乘碰撞检测.下面以确定$c(\mathrm{ADD}_1,\mathrm{D})$为例说明碰撞检测流程,即检测点加$P_4=P_1+P_2$和倍点$P_5=2P_3$关于$P_1=P_3$是否碰撞.在表6的第1行中,$\mathrm{ADD}_1$第7步对应的模乘为$Z_2^2X_1$,倍点D的第11步对应的模乘为$(4Y_3^2)X_3$,二者分别以$X_1$和$X_3$为操作数.易知,攻击可按照如下步骤,将点加和倍点的碰撞检测可被转化为这两个模乘关于$X_1=X_3$的碰撞检测:

(1)从点加对应的功耗曲线中确定出第7步运算对应的功耗曲线,记为$\boldsymbol{tM}_1$;

(2)从倍点对应的功耗曲线中确定出第11步运算对应的功耗曲线,记为$\boldsymbol{tM}_2$;

(3)计算碰撞系数$\rho M=\rho(\boldsymbol{tM}_1,\boldsymbol{tM}_2)$,其中$\rho$可基于相关性或欧式距离等进行定义;

(4)根据ρM的大小将模乘分为两部分,分别对应于两个模乘碰撞和无关;

(5)当两个模乘关于$X_1=X_3$碰撞时,那么$c(\mathrm{ADD}_1,\mathrm{D})=1$,对应的点加和倍点碰撞;反之$c(\mathrm{ADD}_1,\mathrm{D})=0$,对应的点加和倍点无关.

对于其他类型的点加、倍点碰撞检测问题,可采用类似的方式转化为模乘碰撞检测问题.与直接碰撞攻击相比,首先,该方法只利用了特定的模乘,排除了与碰撞检测无关的子操作;其次,该方法将碰撞检测限制在模乘运算之间,解决了点加、倍点中分支语句执行对碰撞检测结果的影响.此外,模乘碰撞检测是针对模幂分析的一种常见问题,其可行性已被相关的研究所证明[23~26].

3.1.4 模乘相等检测

本文发现在表6的第3行存在如下可由Z坐标决定的模乘:

(1)ADD_1:$1.ZZ$、$2.Z^2Z$;

(2)ADD_1:$5.ZZ$、$6.Z^2Z$;

(3)D:$1.ZZ$.

当点加、倍点等操作碰撞时,二者对应的Z坐标将相等,上述模乘对应的两个操作数均相等.由此,模乘碰撞检测将进一步被转化为模乘相等检测,即检测两个模乘对应的两个操作数是否相等.若仍以确定$c(\mathrm{ADD}_1,\mathrm{D})$为例,攻击即被转化为检测点加的第1个模乘$Z_1Z_1$和倍点的第1个模乘$Z_3Z_3$是否相等.

模乘相等检测是本文发现的一种特殊的模乘碰撞检测.模乘包括两个操作数,其中一个操作数与碰撞检测相关,而另一个则为与碰撞检测无关的随机操作数.模乘碰撞检测的结果通常还会受到随机操作数的影响.特别的,当功耗曲线噪声较大,或实现并行度较高时,模乘碰撞检测的难度将明显变大.而模乘相等检测中不存在随机操作数,当二者碰撞时对应的两个操作数都相等,其功耗变化的区别主要是由外部采集环境或其他模块所引入的.显然,模乘相等检测难度将低于一般的模乘碰撞检测.

模乘碰撞检测对于所有的点加、倍点实现都适用.除雅可比坐标外,本文未发现在其他常见的坐标系中存在此类特殊模乘,可将攻击转化为模乘相等检测.这表明雅可比坐标下的点加、倍点碰撞检测的难度相对较低.

3.2 标量乘碰撞攻击

算法1为一种典型的标量乘实现,将标量k从高到低逐比特运算.本节以该算法为例,讨论对标量乘实现的碰撞攻击.考虑连续两比特密钥k_i和k_{i-1}(其中$i\geqslant 0$).如图3所示,其对应的操作依次为D_i、A_i、D_{i-1}、A_{i-1},其中D_i和D_{i-1}为倍点,A_i和A_{i-1}为点加.

首先,从中可识别出与碰撞攻击相关的点加和倍点.当$k_i=0$时,A_i为伪运算,其输出不会作为后续运算的输入,因而D_{i-1}和A_i有相同的操作数$T[0]$,即$c(A_i,D_{i-1})=1$;当$k_i=1$时,A_i的输出将为D_{i-1}的输入,那么D_{i-1}和A_i则没有相同的操作数,即:$c(A_i,D_{i-1})=0$.因此,有

$$k_i=c(A_i,D_{i-1})\oplus 1 \tag{7}$$

由此,攻击首先利用基于模乘的碰撞检测方法对

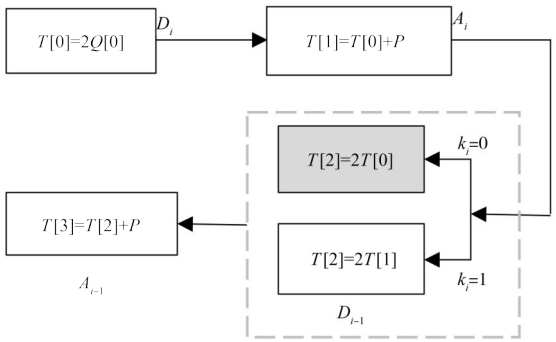


图3　针对总加算法的碰撞分析

D_{i-1}和A_i进行碰撞检测,然后由碰撞检测结果反推对应的k_i. 由于受噪声的影响,碰撞检测的准确率可能无法达到100%,进而导致某些比特出错. 当错误率较小时,对出错位置进行遍历即可恢复出密钥.

3.3　适用性分析

本文以雅可比坐标[27]下的标量乘为分析对象,提出的基于模乘相等的标量乘碰撞检测方法,关键是确定出与碰撞检测相关的模乘. 首先,该方法仅利用两个操作是否有相同的操作数,而并不关注操作数的具体取值. 常规的掩码[3,4]和分支平衡[13~15]等防护并不会改变两个操作碰撞与否,不会影响本文方法的适用性. 其次,本文方法关注更底层的点加、倍点的碰撞检测,不受所采用标量乘实现算法的影响. 对于基于雅可比坐标的标量乘实现,可应用本文的方法将攻击转化为模乘相等检测,即判断两个模乘对应的操作数是否都相等. 而对于基于其他投影坐标的实现,利用与本文方法仅能将攻击转化为模乘碰撞检测,即判断两个模乘中的某个操作数是否相同.

本文以采用了掩码和分支平衡等的标量乘实现为主要分析对象,由此攻击只能在单条曲线中进行. 模乘运算的操作数都为大整数,其在执行过程中将对操作数进行多次处理,因此在模乘运算中普遍存在与之相关的侧信道泄露. 单条曲线分析能否恢复出密钥则取决于采集到的曲线信噪比,对于不同的分析对象可能会有所区别. 只有当信噪比较高时,才能以较高的准确率恢复出密钥. 而对于未采用掩码的实现,则可通过对多条曲线平均的方式进行预处理,攻击难度相对较低,不在本文研究范围内.

3.4　与其他方法比较

由于点加倍点在通用曲线及投影坐标下运算的复杂性,并未发现针对点加、倍点碰撞检测的相关研究. 本文则是首次提出了针对通用曲线及投影坐标下点加倍点的碰撞检测方法,主要针对点加、倍点中的模乘运算. 近期的碰撞攻击研究则以读取操作数[18,20]或条件交换[19]等为分析操作,与本文方法的比较如表7所示,其中"—"表示并不清楚该分析方法是否可用于分析该实现. 由于模乘运算中由于大整数运算产生功耗泄露远大于读取预存点或条件交换等操作,因此本文攻击可利用的泄露相对更大.

表7　不同碰撞检测方法适用性比较

防护方法	本文方法	Hanley[17]	Jin[18]	Nascimento[19]	Jin[20]
坐标随机	√	√	√	√	√
点掩码	√	√	√	√	√
标量掩码	√	×	√	√	√
总加[5]	√	√	×	×	×
MPL[14]	√	√	×	√	×
BRIP[15]	√	√	×	×	×
查找表[18]	√	—	√	×	√

此外,本文攻击方法主要关注利用点加、倍点的实现特点进行碰撞检测,还可与其他分析方法相结合. 首先,针对模乘碰撞检测的方法[23~26]可直接与本文方法相结合. 其次,当碰撞检测的正确率未达到100%时,可以利用已获得的信息对检测结果进行优化,如Perin等[24,28]分别基于聚类和深度学习的技术,提出了提升检测正确率的方法. 此外,还可利用椭圆曲线本身的特性设计密钥恢复算法[29],对碰撞检测结果进行纠错. 这些研究与本文关注攻击的不同阶段,可结合应用以提高攻击效率.

4　实验验证

4.1　实验对象

本文以一款芯片作为实验对象. 该芯片集成了32位CPU核ARM SC100,采用软硬结合的方式实现了标量乘,其中基于硬件模块实现了模乘,基于CPU调用模乘实现了雅可比坐标下的点加和倍点,进而实现了如算法1所示的标量乘总加算法.

本文使用Riscure公司研发的侧信道采集平台从智能卡中采集功耗,如图4所示. 该平台中使用专用读卡器(Power Tracer4)与芯片进行通信,并提供了功耗采集接口,可利用内置的功耗测量装置记录芯片运行期间的功耗变化;同时利用示波器对功耗采集接口的信号进行处理转化为数字信号,保存到计算机中. 本文实验使用的示波器为Lecroy WaveRunner 610Zi,采样频率为1 GHz.

4.2　倍点、加点碰撞检测

本节主要从点加、倍点中提取出与碰撞检测相关模乘对应的功耗曲线,对其分别基于经典碰撞检测、本文提出的基于模乘碰撞的检测和基于模乘相等的检测三种方法进行实验,验证本文方法的可行性.

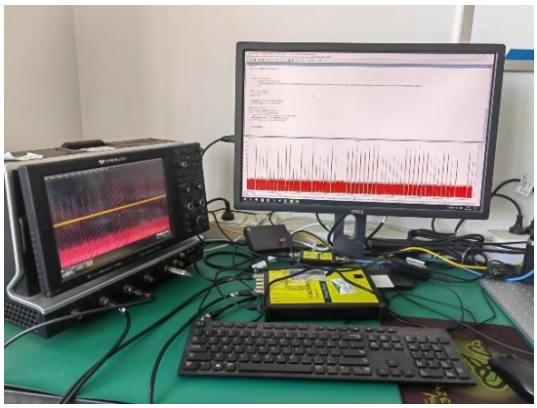


图4 功耗采集环境

4.2.1 功耗特征识别

芯片中可依次执行倍点和点加运算,其对应的功耗曲线如图5上半部分所示.曲线中较高部分的尖峰对应于模乘,较低部分的尖峰则为模加/减.易知,倍点包括8次模乘,点加包括17次模乘.其中灰色区域为点加中的运算,包括1次模乘和模加/减操作.将两条曲线中该区域对应的曲线放大,如图5所示.

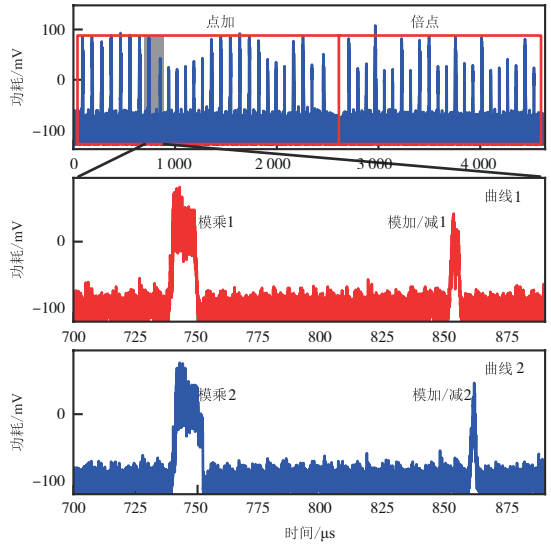


图5 倍点和点加功耗曲线图

在图5中,尽管两条曲线(分别称为曲线1和曲线2)均对应于点加,但其特征则存在明显差异.对于模加/减运算,曲线1中执行了加/减p(模数),因此执行时间长于曲线2;对于模乘运算,曲线2中在模乘最后部分则执行了一次条件约减,因此执行时间则长于曲线1中的模乘.两次点加运算执行了不同分支,使得对应曲线中出现了明显的不对齐现象.如图6所示,本节从点加和倍点分别选择模乘对应的曲线进行碰撞检测,其中上半部分为点加,下半部分为倍点.

在模乘碰撞检测实验中,选择点加的第7个模乘

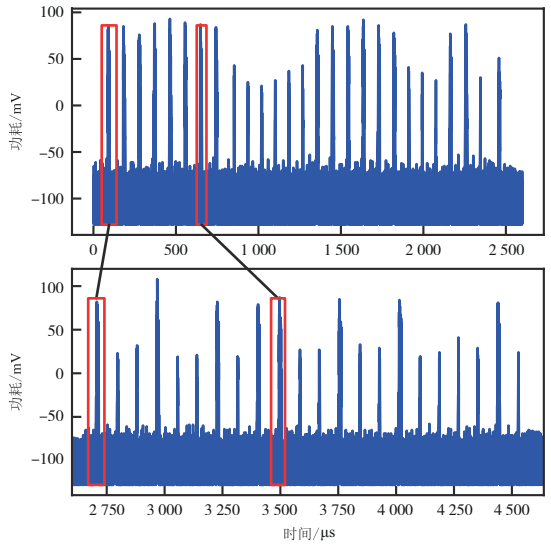


图6 倍点与点加碰撞检测模乘功耗曲线提取

$Z_2^2X_1$和倍点的第11个模乘$(4Y_3^2)X_3$,检测$X_1=X_3$是否成立;在模乘相等检测实验中,选择点加第1个模乘Z_1Z_1和倍点第1个模乘Z_3Z_3,检测$Z_1=Z_3$是否成立.

4.2.2 经典碰撞攻击

碰撞检测即对功耗曲线计算碰撞系数,本文以欧氏距离作为度量两条曲线的碰撞系数.对采集到的1 000条曲线,计算点加和倍点对应功耗曲线之间的欧式距离,结果如图7所示,分别用蓝色和红色标识两个碰撞和无关操作对应的欧式距离.易知,欧氏距离几近于均匀分布.以所有欧式距离的均值作为阈值,当对应的欧式距离小于阈值时,则将对应点加、倍点识别为碰撞;而当对应的欧式距离大于阈值时,则将对应点加、倍点识别为无关.经统计,碰撞检测的正确率约为50.4%.

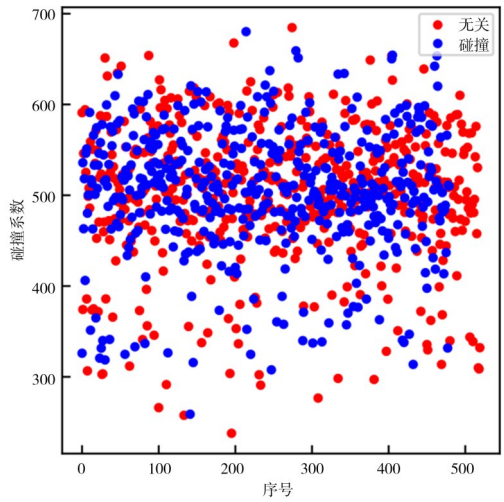
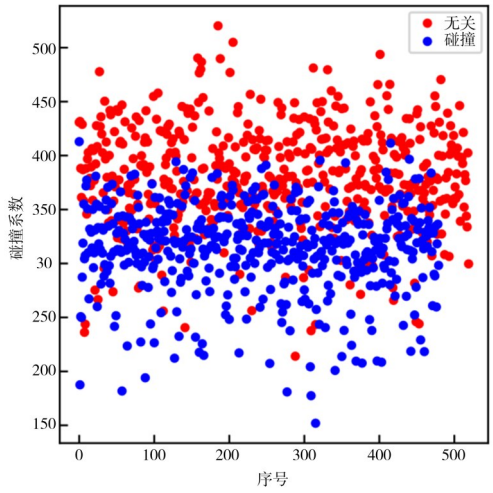


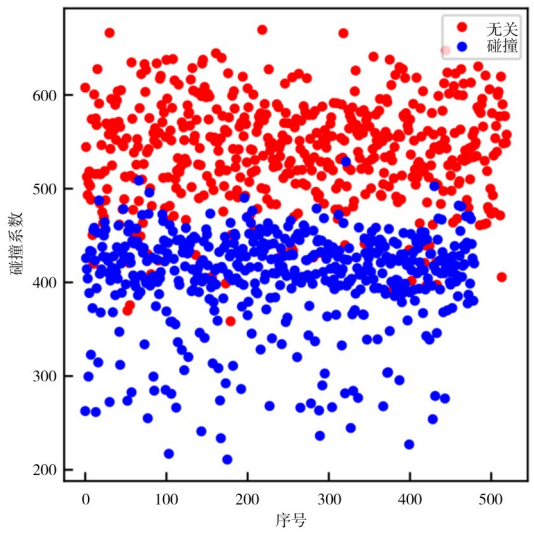
图7 倍点与点加对应曲线的欧式距离分布

4.2.3 基于模乘碰撞的检测

实验中首先从每个点加和倍点中提取出的模乘对应的功耗曲线,计算二者之间的欧式距离;然后以所有欧式距离的均值作为阈值,并根据对应欧式距离与阈值的大小关系确定碰撞检测结果,如图8所示,分别用蓝色和红色标识两个碰撞和无关操作对应的欧式距离.



(a) 模乘碰撞检测



(b) 模乘相等检测

图8　点加、倍点对应碰撞系数的分布

图8(a)为模乘碰撞检测对应欧式距离的分布,碰撞检测准确率为80.4%;图8(b)为模乘相等检测对应欧式距离的分布,检测准确率为93%. 由此表明:两个模乘碰撞与无关两种情况下所对应的欧式距离的分布有明显的区别;模乘相等检测的正确率明显高于模乘碰撞检测的正确率.

为了进一步提升模乘碰撞检测的准确率,本文使用主成分分析[28]对曲线进行降维处理. 首先对两个模乘对应功耗曲线相减,计算出差值曲线;然后对差值曲线基于PCA进行压缩处理,基于第一分量的分布推测碰撞检测的结果,如图9所示. 当该分量大于0时,则判定对应的两个操作碰撞,反之当该分量小于0时,则判定对应的两个操作无关. 经统计,基于模乘碰撞检测的准确率为94.8%,基于模乘碰撞相等的检测准确率为99%.

4.2.4 实验结果比较

本节对雅可比坐标下的点加、倍点的一种典型实现进行了三种碰撞检测实验,如表8所示. 易知:若直接利用点加、倍点对应曲线进行碰撞检测,其结果几近于随机猜测;基于本文提出的基于模乘碰撞检测方法则可准确率超过了80%. 特别的,当对曲线基于PCA进行预处理后,基于模乘碰撞检测和基于模乘相等检测的攻击准确率可达到94.8%和99%. 由此,证明了本文提出的碰撞检测方法的有效性.

表8　不同碰撞检测方法的准确率比较

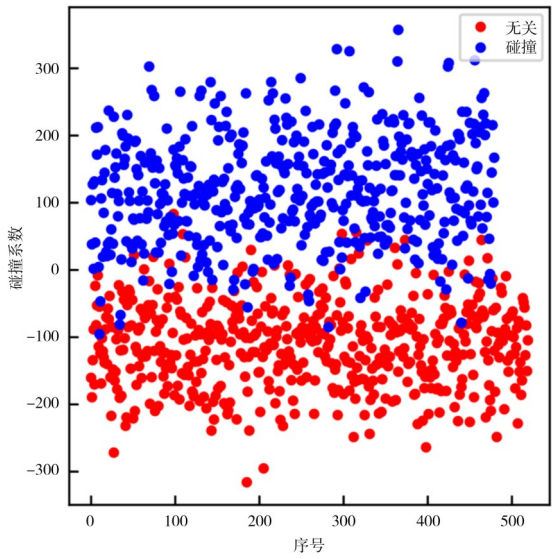
碰撞检测方法	无预处理	基于PCA预处理
经典碰撞检测	50.4%	—
基于模乘碰撞检测	80.4%	94.8%
基于模乘相等检测	93%	99%

Hanley等[17]对标量乘实现进行了经典碰撞攻击,其使用多条曲线进行倍点和点加碰撞检测,准确率平均可以达到96.8%;而基于模乘相等检测的准确率达到了99%,明显优于该方法. 此外,针对模乘碰撞检测的实验结果已经有了若干分析结果,其中基于聚类的碰撞检测[24]实验的最高准确率可达到98.83%;基于深度学习的碰撞检测[28]实验的准确率平均可达到90%,最好情况下可以达到100%. 本文提出的基于模乘相等检测的准确率与这些方法相当.

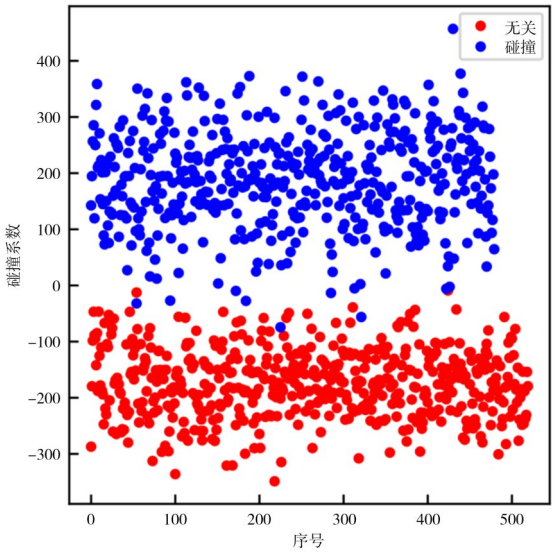
4.3 标量乘碰撞检测

芯片中基于算法1实现了标量乘运算,并采用了64 bit的标量掩码和坐标随机化等掩码防护. 标量乘对应的功耗曲线如图10上半部分所示,其中包括320个重复的功耗特征. 本节将对该曲线进行碰撞攻击. 首先,每个功耗特征包括1次倍点和1次点加. 将图10中红色区域放大,其对应于标量乘中的两个循环,如图10下半部分所示.

碰撞攻击针对算法中相邻的循环i和$i-1$中的点加A_i和倍点D_{i-1}进行碰撞检测. 首先从整条曲线提取出相邻的点加和倍点对应的曲线,如图10下半部分中红色区域所示. 以该段曲线作为模板,通过模板匹配可提取出319条新曲线,每段曲线与标量的1比特相对应. 说明:标量最低比特无法由碰撞攻击获取.



(*a*) 模乘碰撞检测



(*b*) 模乘相等检测

图9 经主成分分析处理后的点加、倍点碰撞检测

基于模乘相等检测的方法对提取出的曲线进行碰撞检测. 首先从这些曲线中提取出对应的模乘,然后使用主成分分析进行压缩处理,第一分量的分布如图11所示. 对于每个分量,当其大于0时则判定对应的点加和倍点两个操作碰撞,对应密钥比特为0;而当其小于0时则识别密钥比特为1. 由此,可初步恢复出整个密钥比特.

由于点加、倍点的碰撞检测中存在错误,进而使得攻击出的标量中同样存在错误比特,利用该标量对输入进行运算得到的结果与输出并不匹配. 本文假定错误比特数小于3,对其中所有比特进行遍历,最终恢复

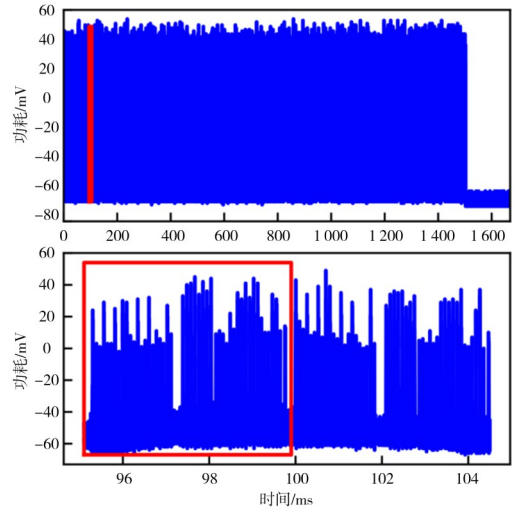


图10 标量乘功耗曲线全貌图

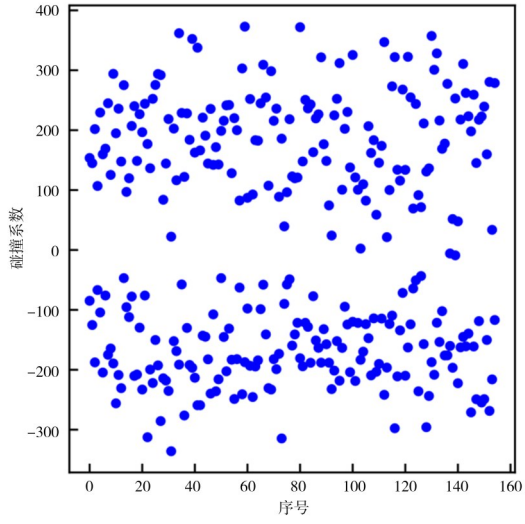


图11 基于模乘相等检测的标量乘碰撞攻击

了正确密钥. 结果表明:碰撞检测中只有1比特出错.

5 总结

本文提出了一种基于模乘的标量乘碰撞检测方法,在功耗曲线的碰撞系数与碰撞检测结果之间建立了新的对应关系. 该方法利用了点加和倍点内模乘间的碰撞关系,排除了无关子操作对攻击的干扰,从而提升了碰撞检测的正确率.

防护中首先考虑使用其他坐标系,以避免威胁最大的基于模乘相等检测攻击. 此外,可对碰撞点进行重掩码,改变两个操作的碰撞关系. 如在点加或倍点执行开始前或完成后,将碰撞点的坐标由(X,Y,Z)重掩码为(Xr^2,Yr^3,r),其中$r\neq 0$为随机数. 由于每个坐标在标量乘中只使用一次,因此任何两个操作中都没有相同的

操作数,从而使得碰撞攻击不可行.

标量乘掩码实现中通常对点加、倍点的实现方式并没有明确的限制,安全性更多的由掩码或标量乘算法所保证. 本文研究表明,点加、倍点的实现将直接影响到标量乘的安全性,雅可比坐标下的安全性相对较低. 此外,攻击中存在多个模乘碰撞检测,如何同时利用这些模乘以提高标量乘碰撞检测准确率是下一步拟研究的问题.

参考文献

[1] KOBLITZ N. Elliptic curve cryptosystems[J]. Mathematics of Computation, 1987, 48(177): 203.

[2] Chinese Cryptography Administration. SM2 elliptic curve public key cryptographic algorithms: GM/T 0003-2012[S]. China, 2010.

[3] PAPACHRISTODOULOU L, BATINA L, MENTENS N. Recent developments in side-channel analysis on elliptic curve cryptography implementations[M]//Hardware Security and Trust. Cham: Springer, 2017: 49-76.

[4] FAN J F, GUO X, DE MULDER E, et al. State-of-the-art of secure ECC implementations: A survey on known side-channel attacks and countermeasures[C]//2010 IEEE International Symposium on Hardware-Oriented Security and Trust. Piscataway: IEEE, 2010: 76-87.

[5] CORON J S. Resistance against differential power analysis for elliptic curve cryptosystems[C]//Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems. New York: ACM, 1999: 292-302.

[6] KOCHER P, JAFFE J, JUN B. Differential power analysis[M]//Advances in Cryptology-CRYPTO 99. Berlin: Springer, 1999: 388-397.

[7] GOUBIN L. A refined power-analysis attack on elliptic curve cryptosystems[M]//Public Key Cryptography — PKC 2003. Berlin: Springer, 2002: 199-211.

[8] AKISHITA T, TAKAGI T. Zero-value point attacks on elliptic curve cryptosystem[M]//Lecture Notes in Computer Science. Berlin: Springer, 2003: 218-233.

[9] DE MULDER E, HUTTER M, MARSON M E, et al. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: Extended version[J]. Journal of Cryptographic Engineering, 2014, 4(1): 33-45.

[10] RYAN K. Return of the hidden number problem[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(1): 146-168.

[11] JANCAR J, SEDLACEK V, SVENDA P, et al. Minerva: The curse of ECDSA nonces[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020 (4): 281-308.

[12] MOGHIMI D, SUNAR B, EISENBARTH T, et al. TPM-Fail: TPM meets timing and lattice attacks[C]//Proceedings of the 29th USENIX Conference on Security Symposium. New York: ACM, 2020: 2057-2073.

[13] 韩晓薇, 乌力吉, 王蓓蓓, 等. 抗简单功耗攻击的SM2原子算法[J]. 计算机研究与发展, 2016, 53(8):1850-1856.
HAN X W, WU L J, WANG B B, et al. Atomic algorithm against simple power attack of SM2[J]. Journal of Computer Research and Development, 2016, 53(8): 1850-1856. (in Chinese)

[14] JOYE M, YEN S M. The montgomery powering ladder[M]//Cryptographic Hardware and Embedded Systems-CHES 2002. Berlin: Springer, 2003: 291-302.

[15] MAMIYA H, MIYAJI A, MORIMOTO H. Efficient countermeasures against RPA, DPA,and SPA[C]//Cryptographic Hardware and Embedded Systems - CHES 2004. Berlin: Springer, 2004: 343-356.

[16] BAUER A, JAULMES E, PROUFF E, et al. Horizontal collision correlation attack on elliptic curves[J]. Cryptography and Communications, 2015, 7(1): 91-119.

[17] HANLEY N, KIM H, TUNSTALL M. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace[C]//Topics in Cryptology-CT-RSA 2015. Cham: Springer, 2015: 431-448.

[18] JIN S, LEE S, CHO S M, et al. Novel key recovery attack on secure ECDSA implementation by exploiting collisions between unknown entries[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(4): 1-26.

[19] NASCIMENTO E, CHMIELEWSKI L. Applying horizontal clustering side-channel attacks on embedded ECC implementations[C]//Smart Card Research and Advanced Applications - CARDIS 2017. Cham: Springer, 2018: 213-231.

[20] JIN S, CHO S M, KIM H, et al. Enhanced side-channel analysis on ECDSA employing fixed-base comb method[J]. IEEE Transactions on Computers, 2022, 71(9): 2341-2350.

[21] ABARZÚA R, VALENCIA C, LÓPEZ J. Survey on performance and security problems of countermeasures for passive side-channel attacks on ECC[J]. Journal of Cryptographic Engineering, 2021, 11(1): 71-102.

[22] MURDICA C, GUILLEY S, DANGER J L, et al. Same

values power analysis using special points on elliptic curves[C]//International Workshop on Constructive Side-Channel Analysis and Secure Design. Berlin: Springer, 2012: 183-198.

[23] HEYSZL J, IBING A, MANGARD S, et al. Clustering algorithms for non-profiled single-execution attacks on exponentiations[C]//International Conference on Smart Card Research and Advanced Applications. Cham: Springer, 2014: 79-93.

[24] PERIN G, CHMIELEWSKI L. A semi-parametric approach for side-channel attacks on protected RSA implementations[C]//International Conference on Smart Card Research and Advanced Applications. Cham: Springer, 2016: 34-53.

[25] ZAID G, BOSSUET L, HABRARD A, et al. Efficiency through diversity in ensemble models applied to side-channel attacks: A case study on public-key algorithms[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(3): 60-96.

[26] SAITO K, ITO A, UENO R, et al. One truth prevails: A deep-learning based single-trace power analysis on RSA-CRT with windowed exponentiation[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(4): 490-526.

[27] HANKERSON D R, VANSTONE S A, MENEZES A J. Guide to Elliptic Curve Cryptography[M]. New York: Springer-Verlag, 2004: 75-109.

[28] PERIN G, CHMIELEWSKI L, BATINA L, et al. Keep it unsupervised: Horizontal attacks meet deep learning[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(1): 343-372.

[29] SCHINDLER W, WIEMERS A. Power attacks in the presence of exponent blinding[J]. Journal of Cryptographic Engineering, 2014, 4(4): 213-236.

作者简介

韩绪仓 男,1987年10月出生于陕西省西安市. 现为中国科学院软件研究所博士研究生. 研究方向为密码算法侧信道分析与防护.
E-mail: xucang2020@iscas.ac.cn

曹伟琼 女,1986年1月出生于广西省桂林市. 现为中国科学院软件研究所助理研究员. 主要研究方向为公钥算法的侧信道分析与防护.
E-mail: caoweiqiong@iscas.ac.cn

陈 华 女,1976年10月生于山东省日照市. 现为中国科学院软件研究所正高级工程师,博士生导师. 研究方向为侧信道分析与防护、密码检测.
E-mail: chenhua@iscas.ac.cn

李昊远 男,1995年11月出生于山东省. 现为中国科学院软件研究所博士研究生. 主要研究方向为密码算法的侧信道分析与防护.
E-mail: haoyuan2019@iscas.ac.cn