

抗 SPA 的多点乘算法

程一飞, 冯新亚

(安庆师范学院 计算机系, 安徽 安庆 246011)

摘 要: SPA(Simple Power Analysis)攻击可能通过泄露的信息获取内存受限制的设备中的密钥,它是通过区分一次点乘运算中点加运算和倍点运算进行的。抗 SPA 攻击的点乘算法较多,但对于多点乘算法相关措施较少。Sharmir-NAF 多点乘算法是一个时间和空间效率都非常优秀的多点乘算法。为此提出一种基于 Sharmir-NAF 的抗 SPA 攻击的多点乘算法。新的算法在内存空间消耗和计算速度上较原算法负担增加可以忽略不计,而且能够抗 SPA 攻击。

关键词: 点乘; 多点乘; SPA; Sharmir-NAF

中图分类号: TP309.7

文献标识码: A

文章编号: 1673-629X(2006)05-0106-03

A SPA-Resistant Multiple Scalar Multiplication Algorithm

CHENG Yi-fei, FENG Xin-ya

(Computer Science Dept. of Anqing Teachers College, Anqing 246011, China)

Abstract: The Simple Power Analysis(SPA) attack might break the implementation of elliptic curve cryptosystem on memory constraint devices such as smart cart against. It attacks elliptic curve cryptosystem by distinguishing between point doubling and point addition in a single execution of scalar multiplication. Although there have been many SPA-resistant scalar multiplication algorithms, there are few countermeasures for multiple scalar multiplication. The Sharmir-NAF algorithm is an excellent algorithm in efficiency and memory. A new SPA-resistant multiple scalar multiplication algorithm is proposed, which is based on the Sharmir-NAF algorithm. The computational and memory overheads of the new algorithm are almost negligible, but it is SPA resistant.

Key words: scalar multiplication; multiple scalar multiplication; simple power analysis; Sharmir-NAF

0 引言

自 1985 年 V.S. Miller 和 Neal Koblitz 分别提出了将椭圆曲线应用于密码系统,大量的研究集中于高效安全实现椭圆曲线密码系统(ECC)。而在 ECC 中,最重要的操作是计算点乘 kP ,其中 k 是一个大整数, P 是曲线上的一个点,而且在很多情况下 k 是需要秘密保存的。椭圆曲线密码体制与其它公钥密码系统相比,在相同安全强度条件下具有较短的密钥长度,因此它非常适合应用于资源有限的设备,如 smart 卡及手持设备等。但是这些设备上的密码系统可能会成为边际信道攻击(Side Channel Attack, SCA)的攻击目标。

SCA 由 Kocher 等人提出^[1,2],是一种能够让敌手通过监视电量消耗来知道包含在设备内部的密钥的技术。SCA 一般分为两种:Simple Power Analysis(SPA)和 Differential Power Analysis(DPA),其中 SPA 攻击仅对算法执行一次所泄露的信息进行分析,而 DPA 攻击是先获取算法多次执行所泄露的信息,然后利用统计工具对所获取的信

息进行分析,是一种较 SPA 更高级的攻击方法。

SCA 攻击最初主要集中于对称密码系统,如 DES 上,但公钥密码系统如 RSA 以及椭圆曲线系统也是易受攻击的对象。针对 SPA 攻击,主要有两类措施:

(1)点乘算法规则化,无论加密算法是对何种数据进行,监测到的信息都是一致和规则的。具体实现方法有算法中加入“伪”代码,例如 Coron^[3]提出在二进制算法中,当 k_i 为 0 时,执行一次虚的加法。这样表现出的情况为每次循环时,一个乘法后跟着一个加法运算。还有,就是直接用已有的规则算法来执行点乘运算。这种方法的典型例子就是用 Montgomery ladder 算法^[4]来进行点乘运算;

(2)在标量乘算法中采用无法区分的点加公式和倍点公式。

目前抗 SCA 攻击的措施都是针对点乘算法的^[5-7],还很少有文献针对多点乘算法进行研究,文中主要研究抗 SPA 攻击的多点乘算法。

1 椭圆曲线和点乘

在有限域 $K = F_2^m$ 上 Weierstrass 方程可以转换成形式如 $E: y^2 + xy = x^3 + ax^2 + b$ 的形式,其中 $a, b \in F_2^m$,且 $b \neq 0$ 。通过特定的加法运算, $E(K) = \{(x, y) \in K \times$

收稿日期:2005-08-20

基金项目:安徽省教育厅自然科学基金项目(2005KJ365zc)

作者简介:程一飞(1976-),男,安徽怀宁人,讲师,硕士,研究方向为计算机密码学。

$K \mid y^2 + xy = x^3 + ax^2 + b \mid \cup \{O\}$, 其中 O 表示无穷远点, 构成一个交换群。

设 $E: y^2 + xy = x^3 + ax^2 + b (b \neq 0)$ 是 F_2^m 中给定的一条曲线, $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ 是 E 上的两个点, 且 $P_1 \neq -P_2$, 那么 P_3 点的坐标可以如下计算:

$$x_3 = \lambda^2 + \lambda + a + x_1 + x_2, y_3 = y_1 + \lambda(x_1 + x_3) + x_3$$

当 $P_1 \neq P_2$ 时, $\lambda = \frac{y_2 + y_1}{x_2 + x_1}$, 此运算称为点加运算,

而当 $P_1 = P_2$ 时, $\lambda = x_1 + \frac{y_1}{x_1}$, 此运算称为倍点运算。

点乘(标量乘): kP 表示点 P 与自身相加 k 次, 即 $kP = P + P + \dots + P$, 共 k 个 P 相加, 称为点乘或标量乘。

多点乘形如: $k_1P_1 + k_2P_2 + \dots + k_tP_t$, 其中 k_i 是整数, P_i 是椭圆曲线上的点。和单个的点乘一样, 它也是椭圆曲线密码应用中的核心计算。它的运算速度和安全性如同点乘算法一样对许多椭圆曲线密码应用至关重要。

在各种多点乘算法中, Sharmir-NAF 多点乘运算效率非常高, 下面给出 Sharmir-NAF 多点乘算法。

算法 1: Sharmir-NAF 多点乘

输入: $P, Q; u, v$, 其中 $u = (u_{n-1}, \dots, u_0)_2, v = (v_{n-1}, \dots, v_0)_2$, 并且 u_{n-1} 或 v_{n-1} 为 1。

输出: $R = uP + vQ$

// 预计算

$P + Q; P - Q;$

// 主计算

分别计算 u 和 v 的 NAF 表示: $u = (u'_{n-1}, \dots, u'_0)_2, v = (v'_{n-1}, \dots, v'_0)_2$

$R \leftarrow u'_{n-1}P + v'_{n-1}Q;$

for i from $n-2$ downto 0 do

$R \leftarrow 2R;$ // 倍点运算

if $(u'_i, v'_i) \neq (0, 0)$ then

$R \leftarrow R + (u'_iP + v'_iQ);$ // 点加运算

return R 。

2 抗 SPA 的 Sharmir-NAF 多点乘算法

2.1 新的算法

算法 1 执行过程如下: (1) 执行一次倍点运算; (2) 当 $(u'_i, v'_i) \neq (0, 0)$ 时, 执行一次点加运算, 这使得攻击者很容易分辨出 $(u'_i, v'_i) = (0, 0)$ 。为了防止这种攻击, 文中提出一种抗 SPA 攻击的算法, 见算法 2。

算法 2: 抗 SPA 的 Sharmir-NAF 多点乘

输入: $P, Q; u, v$, 其中 $u = (u_{n-1}, \dots, u_0)_2, v = (v_{n-1}, \dots, v_0)_2$ 并且 u_{n-1} 或 v_{n-1} 为 1。

输出: $R = uP + vQ$

// 预计算

$P + Q; P - Q; 3P + Q; 3P - Q;$

// 主计算

$R \leftarrow O;$

if u is odd then $u \leftarrow u + 2$

else $u \leftarrow u + 1;$

if v is odd then $v \leftarrow v + 2$

else $v \leftarrow v + 1;$ // 使 u, v 为奇数

$i \leftarrow u, v$ 的长度中的大者 $\text{len};$

while $i > 2$ do

$t_1 \leftarrow u[i-1] \parallel 1;$

if $u[i] = 0$ and $i < \text{len}$ then $t_1 \leftarrow t_1 - 4;$

$t_2 \leftarrow v[i-1] \parallel 1;$

if $v[i] = 0$ and $i < \text{len}$ then $t_2 \leftarrow t_2 - 4;$

$R \leftarrow 4R;$

$R \leftarrow R + (t_1P + t_2Q);$

$i \leftarrow i - 2;$

$t_1 \leftarrow u[i-1 \rightarrow 0];$

if $u[i] = 0$ then $t_1 \leftarrow t_1 - 2^i;$

$t_2 \leftarrow v[i-1 \rightarrow 0];$

if $v[i] = 0$ then $t_2 \leftarrow t_2 - 2^i;$

$R \leftarrow 2^i R;$

$R \leftarrow R + (t_1P + t_2Q);$

if u is odd then $\{u \leftarrow u - 2; R \leftarrow R - 2P;\}$

else $\{u \leftarrow u - 1; R \leftarrow R - P;\}$

if v is odd then $\{v \leftarrow v - 2; R \leftarrow R - 2Q;\}$

else $\{v \leftarrow v - 1; R \leftarrow R - Q;\}$

return R 。

算法 2 首先置标量 u, v 为奇数, 而对奇数标量 u 按照类似 NAF 算法转换成 $0x, \dots, 0x$ 的形式, 其中 x 为奇数, 且 $x \in \{\pm 1, \pm 3\}$ 。

具体方法: 如果 $u[i] = 0$, 则 $t = u[i-1] \parallel 1 - 4$; 如果 $u[i] = 1$, 则 $t = u[i-1] \parallel 1$, 其中 $a \parallel b$ 表示将 a 和 b 联接, 这样保证转换后的结果成为 $0x, \dots, 0x$ 的形式^[5]。

2.2 性能分析

(1) 内存空间需求。

由于 x 为奇数, 且 $x \in \{\pm 1, \pm 3\}$, 则需要预计算 $P + Q; P + 3Q; P - Q; P - 3Q; 3P + Q; 3P - Q; 3P + 3Q; 3P - 3Q$ 。因此需要 2 次倍点运算和 8 次点加运算, 需要预存储 8 个点。Sharmir-NAF 多点乘算法预计算需要 2 次点加运算, 预存储 2 个点。

(2) 非零密度。

标量在进行点乘运算时转换成了 $0x, \dots, 0x$ 形式, 故非零密度为 $1/2$, 新的算法需要进行 $n + 2$ 次倍点运算和 $n/2 + 10$ 次点加运算。Sharmir-NAF 多点乘算法需要 $n - 1$ 次倍点和 $(5n/9 + 2)$ 次(平均)点加操作。因此新的算法在时间效率率较 Sharmir-NAF 多点乘算法要快, 当然代价是预计算需要多存储 6 个点。

另外算法 1 由于 NAF 表示形式需要从右到左进行计算(即从标量的最底位向最高位进行), 故需要首先计算存

储标量的 NAF 表示形式。而由于新的算法对标量编码是从最左到右进行,编码和主计算合并,因此不需要存储标量 u 和 v 的新的编码,这对于内存空间受限的设备来说尤其重要。

(3) 安全性分析。

攻击者虽然可以通过测试电量的消耗来区分点加和倍点运算,但由于新提出的算法通过采用固定模式 $0x, \dots, 0x$ 对标量进行处理,他始终得到相同的序列 $DDA \mid DDA \mid \dots \mid DDA$,其中 D 表示倍点运算, A 表示点加运算,因此他通过 SPA 攻击得不到秘密 u 和 v 。故新的算法是抗 SPA 攻击的。

3 结束语

在内存空间和计算时间负担增加不多的情况下,基于 Sharmir-NAF 提出了一个新的抗 SPA 的多点乘算法。虽然本算法是抗 SPA 的多点乘算法,是针对多点乘运算的,但通过同构等方法,本算法也同样适用于安全的点乘运算。

参考文献:

- [1] Kocher P, Jaffe J, Jun B. Introduction to Differential Power Analysis and Related Attacks[EB/OL]. URL: <http://www.cryptography.com/dpa/technical/index.html>, 1998.
- [2] Kocher P, Jaffe J, Jun B. Differential Power Analysis[A]. In Proceedings of CRYPTO'99, LNCS vol 1666[C]. [s. l.]: Springer-Verlag, 1999. 388-397.
- [3] Coron J S. Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems[A]. In Proceedings of CHES'99, LNCS vol 1717[C]. [s. l.]: Springer-Verlag, 1999. 292-302.
- [4] Montgomery P L. Speeding the Pollard and Elliptic Curve Methods for Factorizations[J]. Mathematics of Computation, 1987, 48: 243-264.
- [5] Okeya K, Takagi T, Vuillaume C. On the Exact Flexibility of the Flexible Countermeasure against Side Channel Attacks[A]. In The 9th australasian conference on information security and privacy, ACISP 2004, LNCS vol 3108[C]. [s. l.]: Springer-Verlag, 2004. 466-477.
- [6] Okeya K, Takagi T. The Width- w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks[J]. IEICE Transactions, 2004, E87-A: 75-84.
- [7] Lee M K. SPA-Resistant Simultaneous Scalar Multiplication[A]. IN Approaches or Methods of Security Engineering Workshop, LNCS vol 3481[C]. [s. l.]: Springer-Verlag, 2005. 314-321.

(上接第 102 页)

件。下面来具体看一下这两种方法。

(1) 将数据库的编码格式临时指定为 GBK。首先在建数据库时指定该数据库的编码格式为 GBK,在 MySQL 控制台中写:CREATE DATABASE test character set 'gbk';在 Java 程序中连接这个数据库时,指定 client 端使用的编码格式为 GBK,具体代码如下:

```
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost: 3306/test? user = " + " root&password = password&useUnicode = true&characterEncoding = gbk"); //连接数据库
```

(2) 修改 MySQL 配置文件。将 MySQL 安装目录中 my.ini 文件的[MySQL]和[client]两个区中的 default-character-set = latin1 都改为 default-character-set = gbk,然后重启 MySQL 服务。这样就将 MySQL 默认的编码格式改成了 GBK,可以直接存取中文数据而不会出现乱码问题。针对 Java 程序和数据库之间的中文乱码问题,笔者提出了三种解决方案,读者可以根据自己的实际情况选择最方便、效率最高的方法。

3 结束语

其实 Java 程序的中文乱码问题并没有想像的那么复

杂,经过对 Java 程序转换过程的分析,知道乱码现象存在的根本原因是含有中文字符的字符串(其编码格式为 GBK 或者 GB2312)被当作别的编码格式编码或者解码了。所以只要保证程序入口和出口的汉字信息不失真就可以解决 Java 中文乱码问题。

参考文献:

- [1] Wang Yu. Multibyte-character processing in J2EE[EB/OL]. <http://www.javaworld.com>, 2004-04-19.
- [2] Fengsundy. Java 中文问题详解,底层编码解剖[EB/OL]. <http://www.pconline.com.cn/481175.html>. 2004-10-29.
- [3] Johnlhr. Java 处理中文化问题详解[EB/OL]. <http://www.globebbs.com/bbs/showthread.php>. 2005-01-03.
- [4] 周竞涛,王明微. 关于 Java 中文问题的几条分析原则[EB/OL]. <http://www.javafan.net/article/20050330155356600.html>. 2002.
- [5] 段明辉. Java 编程技术中汉字问题的分析及解决[EB/OL]. http://www-128.ibm.com/developerworks/cn/java/java_chinese/index.html. 2000.
- [6] Monty M, Widenius, Axmark D, et al. MySQL Reference Manual[M]. [s. l.]: O'Reilly & Associates, Inc, 2002.