一种改进的固定基点标量乘快速算法

王玉玺 张串绒 张柄虹

(空军工程大学信息与导航学院 西安 710077)

摘 要 对于固定基点的标量乘法,LLECC算法具有很高的计算效率,但是预计算量大、存储空间要求高限制了算法的应用。采用基于窗口的非相邻编码方法对标量k编码并按照新的排列方式得到系数矩阵后,利用编码方法的稀疏特性便可降低算法的存储量;为解决新的编码方式下增加的倍点计算,利用二进制有限域上计算效率较高的半点计算代替一般的倍点运算,从而提高改进算法的计算效率。对比分析显示,在标量长度为160bit、编码窗口宽度为4bit等相同条件下,改进算法与原算法相比计算效率提高了12.4%,存储量降低了53.3%。

关键词 椭圆曲线,标量乘,半点运算,基于窗口的非相邻编码

中图法分类号 TN918.1 **文献标识码** A

Improved Fast Algorithm of Scalar Multiplication for Fix Base Point

WANG Yu-xi ZHANG Chuan-rong ZHANG Bing-hong

(College of Information and Navigation, Air Force Engineering University, Xi'an 710077, China)

Abstract For the scalar multiplication of a fix base point, the algorithm of LLECC has a good performance in efficiency, however the huge amount of precomputation and storage restrict its application. Through a new arrangement of the coefficient matrix in which the scalar k is recoded in the non-adjacent form based on the window, the storage can be reduced with the sparse property of the encoding method. When the length of the scalar is 160bit and the window's width is 4bit, the improved algorithm can reduce 12.4% and 53.3% in the aspect of the computational complexity and storage cost, compared with the original LLECC algorithm.

Keywords Elliptic curve, Scalar multiplication, Point halving, NAF-w

1 引言

椭圆曲线公钥密码(ECC)由Neal Koblitz[1]和Victor Miller[2]于1985年提出,算法的安全性基于椭圆曲线离散对数问题的难解性,是利用有限域上椭圆曲线有限群代替基于离散对数问题密码体制中的有限循环群所得到的一类密码体制。椭圆曲线公钥密码因为其较高的单位比特安全性和较高的计算效率,特别适合于手机、智能卡等计算能力及存储空间有限的终端设备。ECC密码虽然具有较好的计算效率,但是仍不能满足许多实际应用的需要,围绕ECC算法提高其计算效率成为目前研究的重点。

由椭圆曲线加密机制(ECIES)可以看出,影响椭圆曲线公钥密码计算效率的主要运算为椭圆曲线上点的标量乘运算,即给定曲线上一点P及整数k,计算$Q=kP$。如何降低标量乘算法计算量、提高计算速度,成为改善整个算法计算效率的关键。Lim和Lee在文献[3]中提出了一种灵活的基于预计算的快速标量乘算法(LLECC),该算法在存储空间足够大的情况下具有非常高的计算效率,但是当存储空间有限时,算法效率受到严重制约。在2002年,Yang在文献[4]中将LLECC算法与非相邻标量编码方法(NAF)及对齐算法[5]相结合,提出了新的改进算法,该算法与原有算法相比计算效率得到进一步提高,但是需要存储更多的预计算量,新算法仍不适于具有较小存储能力的无线移动设备。

本文围绕提高LLECC算法效率、降低算法存储量,提出了利用基于窗口的非相邻编码方法(NAF-ω)对标量重新编码,并通过新的排列方式得到系数矩阵,充分利用编码方法的稀疏特性减少算法所需要存储的预计算点;同时利用半点运算代替传统的倍点运算,提高改进算法的计算效率。通过分析对比发现,新算法计算效率提高了12.4%,存储空间节省了53.3%。

2 基础知识

定义在有限域F_{2^m}上的椭圆曲线$E: y^2+xy=x^3+ax+b$,其中$a,b\in F_{2^m}$,$b\neq 0$。令点$P=(x_1,y_1)\in E(F_{2^m})$,$P\neq -P$,则仿射坐标下倍点计算公式为$2P=(x_3,y_3)$,其中

$$\begin{cases} x_3=\lambda^2+\lambda+a=x_1^2+\dfrac{b}{x_1^2} \\ y_3=x_1^2+\lambda x_3+x_3 \\ \lambda=x_1+\dfrac{y_1}{x_1} \end{cases} \tag{1}$$

令曲线上不同两点$P=(x_1,y_1)\in E(F_{2^m})$和$Q=(x_2,y_2)$

到稿日期:2013-01-08 返修日期:2013-04-28 本文受国家自然科学基金(61272486)资助。

王玉玺(1989-),男,硕士生,主要研究方向为密码学与网络安全,E-mail:WangYX10013@163.com;张串绒(1964-),女,博士后,教授,主要研究方向为密码学与信息安全。

$\in E(F_{2^m})$，$P\neq\pm Q$，则点加公式为 $P+Q=(x_3,y_3)$，其中：

$$\begin{cases} x_3=\lambda^2+\lambda+x_1+x_2+a \\ y_3=\lambda(x_1+x_3)+x_3+y_1 \\ \lambda=\dfrac{y_1+y_2}{x_1+x_2} \end{cases} \tag{2}$$

设 I 表示有限域上逆运算，M 表示有限域上乘法运算，S 表示有限域上乘方运算，由上述计算公式可知点加运算(A)的计算量为 $I+2M+S$，倍点运算(D)的计算量为 $I+2M+2S$。

2.1 LLECC 算法

对于基点固定的标量乘运算，当存储空间允许时，可以将算法中可能用到的部分中间结果进行预计算并存储，实现牺牲空间提高算法效率的目的。基于该思想，在 1994 年 Lim 和 Lee 提出了一种适用于无线网络环境中的灵活的预计算方法，该方法在存储空间足够大的条件下具有较高的计算效率。

将二进制标量 k 写成 $h\times a$ 的矩阵形式，记矩阵每一行为 k_i，$0\leqslant i\leqslant h-1$，每行比特数为 $a=\left\lceil\dfrac{l}{h}\right\rceil$。将 k_i 进一步分为 v 块，每一块记为 $k_{i,j}$，其比特数为 $b=\left\lceil\dfrac{a}{v}\right\rceil$，$0\leqslant j\leqslant v-1$。标量 k 的矩阵表示如表 1 所列。

表 1 标量 k 的矩阵表示形式

k_0	$k_{0,v-1}$	…	$k_{0,j}$	…	$k_{0,0}$
…	…	…	…	…	…
k_i	$k_{i,v-1}$	…	$k_{i,j}$	…	$k_{i,0}$
…	…	…	…	…	…
k_{h-1}	$k_{h-1,v-1}$	…	$k_{h-1,j}$	…	$k_{h-1,0}$

由表 1 可得：

$$k=k_{h-1}k_{h-2}\cdots k_1k_0=\sum_{i=0}^{h-1}k_i2^a \tag{3}$$

式中，$k_i=k_{i,v-1}k_{i,v-2}\cdots k_{i,1}k_{i,0}=\sum_{j=0}^{v-1}k_{i,j}2^{jb}$。

设 $P_0=P$，$P_i=2^aP_{i-1}=2^{ia}P$，对于矩阵中每一行二进制序列 $k_i=(e_{i,a-1}e_{i,a-2}\cdots e_{i,1}e_{i,0})_2$，有 $k_{i,j}=(e_{i,jb+b-1}e_{i,jb+b-2}\cdots e_{i,jb+1}e_{i,jb})_2$，则：

$$kP=\sum_{i=0}^{h-1}k_iP_i=\sum_{r=0}^{b-1}2^r\sum_{j=0}^{v-1}\sum_{i=0}^{h-1}e_{i,jb+r}2^{jb}P \tag{4}$$

为提高标量乘算法的计算效率，需要进行一定的预计算，即对所有 $0\leqslant i<2^h$，$0\leqslant j\leqslant v-1$，计算：

$$\begin{cases} G[0][i]=e_{h-1}P_{h-1}+e_{h-2}P_{h-2}+\cdots+e_0P_0 \\ G[j][i]=2^b(G[j-1][i])=2^{jb}G[0][i] \end{cases} \tag{5}$$

式中，$P_i=2^iP$，利用预计算得到标量乘计算公式为：

$$kP=\sum_{r=0}^{b-1}2^r\sum_{j=0}^{v-1}G[j][I_{j,r}] \tag{6}$$

式中，$I_{j,r}=e_{h-1,bj+r}e_{h-2,bj+r}\cdots e_{1,bj+r}e_{0,bj+r}$，$0\leqslant j<b$。

算法 1 LLECC 算法

Input: scalar k, point P

Output: kP

1. Set R=O
2. For r=b−1 to 0 by−1 do
   R=2R
   For j=v−1 to 0 by−1 do
   R=R+G[j][$I_{j,r}$]
3. Return R

通过分析可知，LLECC 算法需要计算并存储 $(2^h-1)v$ 个预计算点，算法平均计算量为 $[(1-\frac{1}{2^h})a-1]A+(b-1)D$。

2.2 半点运算

由二进制域 F_{2^m} 上椭圆曲线倍点计算式(1)可知，若已知点 $P=(x,y)$，则 $Q=2P=(u,v)$ 计算如下：

$$\lambda=x+\frac{y}{x} \tag{7}$$

$$u=\lambda^2+\lambda+a \tag{8}$$

$$v=x^2+u(\lambda+1) \tag{9}$$

半点运算为倍点运算的逆运算，即指给定 $Q=(u,v)$，计算 $P=(x,y)$。基本思路为由式(8)求解 λ，进而由式(9)解得 x，最后由式(7)解得 y。由文献[6]可知，当 $\{G\}$ 为椭圆曲线上一个 n 阶(n 为奇数)子群时，倍点和半点是子群 $\{G\}$ 上的自同构，因此对于任意一点 $Q\in\{G\}$，都有 $P\in\{G\}$ 使得 $Q=2P$。半点运算具体算法为：

算法 2 半点计算

Input: Q=(u,v)

Output: $P=\frac{1}{2}Q=(x,\lambda_p)$

1. Find a solution $\hat{\lambda}$ of $\hat{\lambda}^2+\hat{\lambda}=u+a$
2. Compute $t=v+u\hat{\lambda}$
3. If $Tr(t)=0$, $\lambda_p=\hat{\lambda}$, $x=\sqrt{t+u}$
   Else $\lambda_p=\hat{\lambda}+1$, $x=\sqrt{t}$
4. Return(x,λ_p)

由文献[6]可知，在输入点为仿射坐标表示形式时，算法 2 的计算复杂度为 $2M$。

2.3 NAF-ω 算法

标量乘运算中基本计算单元为点加和倍点运算，对标量 k 合理编码可以有效降低点加计算次数，从而提高标量乘计算效率，目前应用较多的标量编码方式为非相邻编码方式(NAF)[7]。由于利用 NAF 的标量乘算法一次只能处理 2 位，制约了算法效率的进一步提高，在存储空间允许的情况下提出了一次处理 w 位的基于窗口的 NAF 算法，通过预计算牺牲空间换取时间进一步提高计算效率。

算法 3 NAF-ω 算法

Input: window width w, scalar k

Output: $NAF_w(k)$

1. Set i=0
2. While k≥1 do
   2.1 If k%2!=0
   $k_i=k\bmod 2^w$, $k=k-k_i$
   Else $k_i=0$
   2.2 k=k/2, i=i+1
3. Return $NAF_w(k)=(k_{i-1},k_{i-2},\cdots,k_1,k_0)$

在长度为 l 的 $NAF_w(k)$ 中，平均非零数字密度为 $\dfrac{l}{w+1}$，具有更好的稀疏性，序列中每一个非零元素 k_i 的取值范围为 $[-2^{w-1}+1,2^{w-1}-1]$。

3 LLECC 算法改进

通过对原有 LLECC 算法分析可知，影响算法应用的主

要因素为预计算量大、对设备存储能力要求较高。本文通过利用 NAF-ω 编码方法对标量 k 的矩阵表示形式重新编码，并充分利用 NAF-ω 编码的稀疏特性降低了算法存储量，同时利用计算效率较高的半点运算代替算法中原有的倍点运算，使算法效率进一步提高。

设椭圆曲线上一点 $P\in\{G\}$，令 $t=\lfloor \log_2 n \rfloor+1$，其中 n 为子群$\{G\}$的阶，利用算法 3 计算 $NAF_w(2^t k \bmod n)$，则有：

$$k\equiv\sum_{i=0}^{t}\frac{k_{t-i}'}{2^i}(\bmod n)$$
$$\equiv\frac{k_0'}{2^t}+\frac{k_1'}{2^{t-1}}+\ldots+\frac{k_{t-1}'}{2}+k_t'(\bmod n) \quad (10)$$

式中，$k_i'\in[-2^{w-1}+1,-2^{w-1}-1]$。

由表 1 可知原有 LLECC 算法中标量 k 的矩阵表示顺序为从右到左、从上到下。为充分利用 NAF-ω 编码方式的稀疏特性，本文采用从上到下、从右到左的排列方式，在半点表示形式下有：

$$k=c_{a-1}c_{a-2}\cdots c_1c_0=\sum_{l=0}^{a-1}c_l\left(\frac{1}{2}\right)^{lh}$$

设 $P_0=P,P_j=\left(\frac{1}{2}\right)^{hb}P_{j-1}=\left(\frac{1}{2}\right)^{jhb}P$，$0<j<v$，则标量乘计算公式为：

$$kp=\sum_{l=0}^{a-1}c_l\left(\frac{1}{2}\right)^{lh}P=\sum_{r=0}^{b-1}\left(\frac{1}{2}\right)^{rh}\sum_{j=0}^{v-1}c_{jb+k}\left(\frac{1}{2}\right)^{jhb}P \quad (11)$$

式中，$c_{jb+k}=e_{h-1,bj+k}e_{h-2,bj+k}\cdots e_{1,bj+k}e_{0,bj+k}$。

由 NAF-ω 编码特性可知，对于长度为 h 的 NAF-ω 编码序列有：

$$-\left(\sum_{i=1}^{\lceil\frac{h}{w}\rceil}(2^{w-1}-1)\times 2^{w(i-1)}\right)\leqslant e_{h-1}\cdots e_0$$
$$\leqslant\left(\sum_{i=1}^{\lceil\frac{h}{w}\rceil}(2^{w-1}-1)\times 2^{w(i-1)}\right) \quad (12)$$

根据椭圆曲线上点逆运算计算复杂度可以忽略的特性，改进后的 LLECC 算法只需对所有 $1\leqslant i\leqslant\left(\sum_{i=1}^{\lceil\frac{h}{w}\rceil}(2^{w-1}-1)\times 2^{w(i-1)}\right)$和 $0\leqslant j\leqslant v$ 进行预计算：

$$\begin{cases}G[0][i]=e_{h-1}\frac{1}{2^{h-1}}P+e_{h-2}\frac{1}{2^{h-2}}P+\cdots+e_0P\\ G[j][i]=\frac{1}{2^{hb}}(G[j-1][i])=\frac{1}{2^{jhb}}G[0][i]\end{cases}$$

所以有：

$$kP=\sum_{r=0}^{b-1}\frac{1}{2^{hr}}\sum_{j=0}^{v-1}G[j][I_{j,r}] \quad (13)$$

$I_{j,r}=e_{h-1,bj+r}e_{h-2,bj+r}\cdots e_{1,bj+r}e_{0,bj+r}$。改进后 LLECC 算法为：

算法 4 LLECC 改进算法

Input: scalar k, point P

Output: kP

1. Compute $NAF_w(2^tk \bmod n)$ using Algorithm 3, then

$$k\equiv\sum_{i=0}^{t}\frac{k_{t-i}'}{2^i}(\bmod n)$$
$$\equiv\frac{k_0'}{2^t}+\frac{k_1'}{2^{t-1}}+\cdots+\frac{k_{t-1}'}{2}+k_t'(\bmod n)$$

2. Translate the sequence $(k_0'k_1'\cdots k_{t-1}'k_t')$ into a h×a matrix from up to down and then from right to left

3. Set R=0

4. For r=b−1 to 0 by −1 do

4.1 If h=1 then $R=\frac{1}{2}R$

Else Compute $R=\frac{1}{2^h}R$ using Algorithm 2

4.2 For j=v−1to 0 by −1 do

If $(e_{h-1,bj+r}e_{h-2,bj+r}\cdots e_{1,bj+r}e_{0,bj+r})_{NAF_w}>0$

then $R=R+G[j][I_{j,r}]$

Else if $(e_{h-1,bj+r}e_{h-2,bj+r}\cdots e_{0,bj+r})_{NAF_w}<0$

then

$I_{j,r}'=-(e_{h-1,bj+r}e_{h-2,bj+r}\cdots e_{1,bj+r}e_{0,bj+r})_{NAF_w}$ and

$R=R-G[j][I_{j,r}]$

5. Return R

改进后算法通过变换标量系数矩阵的排列方式，充分利用 NAF-ω 编码算法的稀疏特性，在满足 $h \bmod w\equiv 0$ 的情况下，减少了算法需要存储的预计算量。利用半点运算代替一般的倍点运算，在算法 4.1 步中减少计算量，改进后的 LLECC 算法平均计算量为$[[1-(\frac{w}{w+1})^h]a-1]A+h(b-1)H$，$H$ 代表半点计算，算法仅需保存$(\sum_{i=1}^{\lceil\frac{h}{w}\rceil}(2^{w-1}-1)\times 2^{w(i-1)})v$ 个预计算点。

4 改进算法性能分析

为分析改进后算法在计算效率及所需存储空间的性能优势，将改进后算法与原有算法及 Yang 在文献[4]中所提算法就算法平均计算量及算法存储的预计算点进行对比，得到表 2。

表 2 算法性能比较

算法名称	平均计算量	存储空间
LLECC 算法	$[(1-(1/2)^h)a-1]A+(b-1)D$	$(2^h-1)v$
Yang 算法	$[(1-0.71^h)a-1]A+(b-1)D$	$\frac{(3^h-1)}{2}v$
LLECC 改进算法	$[[1-(w/w+1)^h]a-1]A+h(b-1)H$	$(\sum_{i=1}^{\lceil\frac{h}{w}\rceil}(2^{w-1}-1)\times 2^{w(i-1)})v$

为了便于比较，设特征值为 2 的有限域内求逆运算与乘法运算及乘方运算与乘法运算的比例关系为 $I/M=10$，$S/M=0.8$[8]。设标量长度为 160bit，算法各项参数为：$h=8$，$a=20$，$v=5$，$b=4$，$w=4$，则原 LLECC 算法平均计算量为 283M，需存储 1275 个预计算点坐标；Yang 算法平均计算量为 267.5M，需存储 32800 个预计算点坐标；改进后 LLECC 算法平均计算量为 248.25M，需存储 595 个预计算点坐标。由上述分析对比可知，在标量长度为 160bit 时，本文提出的改进算法与 Yang 算法和原算法相比计算效率分别提高了 7.2%、12.4%，与原算法相比存储空间减少了 53.3%。

结束语 本文通过对二进制域上固定基点 P 的 LLECC 标量乘算法进行分析，围绕算法对用户设备存储能力有较高要求的缺陷，提出利用 NAF-ω 编码方法对标量系数矩阵进行重新编码，通过改变原有算法矩阵元素的排列方式，充分利用了 NAF-ω 编码的稀疏特性，使改进后算法存储量降低 53.3%；为提高计算效率，对改进算法中出现的 2^tP 运算采用具有较高计算效率的半点计算，从而使算法与具有较高计算效率的 Yang 算法相比再提高 7.2%。

参 考 文 献

[1] Koblitz N. Elliptic curve cryptosystems [J]. Mathematics of computation, 1987(48): 203-209

[2] Miller V S. Use of elliptic curves in cryptography[C]// Proceedings of CRYPTO'85. LNCS 218, Springer, 1986: 417-426

[3] Lim C H, Lee P J. More flexible exponentation with precomputation Advances in Cryptology[C]// Crypto'94. Springer-Verlag, Berlin, 1994: 95-107

[4] Yang W C, Lin K M, Laih C S. A precomputation method for elliptic curve point multiplication[J]. Journal of the Chinese Institute of Electrical Engineering, 2002, 9(4): 339-344

[5] Lo G W. The study and implementation on elliptic curve digital signature schemes[D]. Taiwan, NCKU, 2000

[6] Fong K, Hankerson D, Lopez J, et al. Field inversion and point halving revisited [J]. IEEE Transactions on Computers, 2004, 53(8): 1047-1059

[7] Hankerson D, Menezes A, Vanstone S. Guide to Elliptic Curve Cryptography[J]. Computer Reviews, 2005, 46(1): 13-23

[8] Digital Signature Standard(DSS), FIPS 186-2[S]. USA: Federal Information Processing Standards Publication 186-2, National Institute of Standards and Technology, 2000

[9] 白羽，范恒英. 一种改进的椭圆曲线标量乘的快速算法[J]. 西南科技大学学报，2011，09

[10] Pang Shi-chun, Tong Shou-yu, Cong Fu-zong, et al. An Efficient Elliptic Curve Scalar Multiplication Algorithm Against Side Channel Attacks[C]// Proceedings of the 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE2010). Springer-Verlag, Berlin: 2010: 361-364

[11] 张建. GF(2^n)上椭圆曲线标量乘法快速算法的研究[D]. 呼和浩特：内蒙古大学，2012

[12] 周梦，周海波. 椭圆曲线快速点乘算法优化[J]. 计算机应用研究，2012，29(8)：3056-3058

[13] 李忠，彭代渊. 基于滑动窗口技术的快速标量乘法[J]. 计算机科学，2012，39(6A)：54-57

(上接第 121 页)

我们用 SEVE 产生若干个输入(实验设置为 5 个)后，马上将其运行以发现程序是否出错。表 1 的倒数两栏式符号执行性能评价，包括符号化字节数和最大路径约束个数。表 2 为使用 fuzzing 方法对上述两个漏洞的测试结果。RAZ 执行时间同样设定为 500 分钟。RAZ 分别产生了 3902 和 4493 个测试用例，其中针对第一个漏洞，第 2910－3002 号输入使程序崩溃；针对 mp3 漏洞，第 2141－2205 号输入使程序崩溃。

从表 1 及表 2 可以看出，SEVE 明显优于传统 fuzzing。首先，输入不依赖文件格式，即使对于 Word 等复杂格式也可分析，将 SEVE 稍作扩展，可用于网络型软件对协议漏洞进行分析，对于 SMB 等未公开协议尤其有效。其次，符号执行能够生成较有效的测试用例，在较早的时候发现软件 bug。

结束语 本文设计并实现了基于符号执行的二进制级漏洞发现系统，它有效克服了传统 fuzzing 测试的缺点，能有效生成测试输入，且无需分析输入格式。实验表明，我们的系统能在较短时间内产生测试输入并发现软件漏洞。后续工作主要在 3 个方面：1)对汇编指令进行精确语义分析并包含更多类型指令；2)求解器性能优化；3)执行树遍历算法研究。

参 考 文 献

[1] 王彤彤，韩文报，王航. 基于安全需求的软件漏洞分析模型[J]. 计算机科学，2007，34(9)：287-289

[2] 毛澄映，卢炎生，胡小华. 数据挖掘技术在软件工程中的应用综述[J]. 计算机科学，2009，36(5)：1-6

[3] Brumley D, Poosankam P, et al. Automatic patch-based exploit generation is possible: techniques and implications[C]// SP'08: Proceedings of the IEEE Security and Privacy Symposium. NJ: IEEE, 2008: 143-157

[4] 宋超臣，黄俊强，等. 计算机安全漏洞检测技术综述[J]. 技术研究，2012，01：77-79

[5] 吴志勇，王红川，等. Fuzzing 技术综述[J]. 计算机应用研究，2010，27(3)：829-832

[6] Miller B P, Fredriksenl. An empirical Study of the reliability of UNIX utilities [J]. Communications of the ACM, 1990, 33(12): 32-44

[7] Miller Barton P, Gergogy C, Fresrick M. An empirical study of the robustness of MacOS applications using random testing[C]// Proceedings of the 1st International Workshop on Random Testing. New York: ACM, 2006: 46-54

[8] 陈衍铃，赵静. 基于虚拟化技术的动态污点分析[J]. 计算机应用，2011，31(9)：2367-2372

[9] Cowbc, Pu C, et al. StackGuard : Automatic adaptive detection and prevention of buffer-overflow attacks[C]// Proceedings of the 7th conference on USENIX Security Symposium. Berkeley, 1998: 5-13

[10] King J C. Symbolic execution and program testing[C]// Communications of the ACM. 1976: 385-394

[11] 过辰楷，姬秀娟，许静. 基于分支混淆算法的符号执行技术 [J]. 计算机科学，2012，39(9)：115-119

[12] Cadar C, Ganesh V, et al. EXE: Automatically generating inputs of death[C]// CCS'06: Proceedings of the 13th ACM Conference on Computer and Communications Security. New York: ACM, 2006: 322-335

[13] Luk C-K, Robert C, et al. Pin: building customized program analysis tools with dynamic instrumentation[C]//PLDI05: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM, 2005: 190-200

[14] De Moura, Leonardo, Bjrner, et al. Z3: an efficient SMT Solver [C]//TACAS: Proceedings 14th International Conference. Berlin: Springer, 2008: 337-340

[15] 魏瑜豪，张玉清. 基于 fuzzing 的 MP3 播放软件漏洞发掘技术 [J]. 计算机工程，2007，33(24)：158-167