文章编号:1001 - 9081(2006)09 - 2131 - 03

椭圆曲线点乘 IP 核的设计与实现

邹候文[1],王 峰[2],唐 屹[1]
(1. 广州大学 数学与信息科学学院,广东 广州 510006; 2. 福建工程学院 数理系,福建 福州 350014)
(zouhw@163.com)

摘 要:基于 NIST 推荐的 GF(2^163)上的 Koblitz 曲线,根据 López 改进 Montgomery 点乘算法,提出一种有限状态机控制的 ECC 点乘实现方案,设计了 ECC 点乘 IP 核。用 Quartus Ⅱ 5.0 在 EP2S90F1508C3 器件中综合仿真,整个 IP 核消耗逻辑资源 14 502 个 ALUTs,最高主频 166MHz,点乘运算速度可达 12 835 次/s。

关键词:椭圆曲线; 点乘; IP 核
中图分类号: TP309.7 **文献标识码**:A

Implementation of ECC point multiplication with IP core

ZOU Hou-wen[1], WANG Feng[2], TANG Yi[1]
(1. *School of Mathematics and Information Science, Guangzhou University, Guangzhou Guangdong* 510006, *China;*
2. *Department of Mathematics and Physics, Fujian University of Technology, Fuzhou Fujian* 350014, *China*)

Abstract: Based on Koblitz curve over GF (2^163) recommended by NIST and in the opinion of improved Montgomery algorithm introduced by López, an operation scheme control by Finite State Machines was presented and an ECC(Elliptic Curves Cryptography) point multiplication with IP core was designed. Comprehensive simulation by Quartus Ⅱ 5.0 was made in EP2S90F1508C3. A total of 14 502 ALUTs were consumed, the maximal dominant frequency was 166MHz and the speed of point multiplication was 12 835 times per second.

Key words: ECC(Elliptic Curves Cryptography); point multiplication; IP core

0 引言

NIST 推荐的 ECC 标准[1]分别基于素数域和二元域,而二元域又分别基于多项式基和正规基。在普通计算机中,素数域和二元域正规基的 ECC 运算速度基本一致,较多项式基快[2]。在硬件实现上二元域较素数域更有利于硬件实现,并且正规基优于多项式基[3]。在 ECC 中最重要的运算是点乘($Q = kP$),其他辅助运算耗时在理论分析上常常可以忽略。ECDSA 方案[4]中签名一次需要一次点乘运算,签名验证需要两次点乘运算,因此人们研究 ECC 的实现时主要集中在点乘算法上,常用的方法有 Double-Addiction、NAF、NAF-Windows 和 Montgomery 等,在坐标策略上分别有仿射、射影和混合坐标等[2]。由 López 提出的基于射影坐标改进 Montgomery 点乘算法中[5],$GF(2^m)$ 上的 ECC 点乘需要作 $6(m-1)+10$ 次域上的乘法和 1 次求逆,是目前时间复杂度最优的点乘算法。恰当选择椭圆曲线的参数,在安全度不降低的情况下乘法次数可进一步减为 $5(m-1)+10$ 次。正规基表示的域元平方和多次连续的平方都可以通过硬件循环移位一次实现,针对这一特性,结合费马小定理和 Itoh 等人提出的方法[6],对于 m 为 163 的求逆运算我们可以用 9 次域上乘法实现。

本文选择 NIST 推荐的 $GF(2^{163})$ 上的 Koblitz 曲线,采用 López 提出的基于射映坐标改进 Montgomery 点乘算法,利用硬件实现乘方开销非常小的特点把求逆通过 9 次域上乘法实现,提出一种有限状态机控制的 ECC 点乘实现方案。FPGA 综合仿真结果表明,本文设计的 ECC 点乘 IP 核需要做 $829(5*162+10+9)$ 次域上乘法完成一次点乘运算,适当选择乘法器参数(字宽为 16),在 EC2S90F1508C3 器件下消耗 14 502 个 ALUTs,最高主频达到 166MHz,每秒可以完成 12 835 次点乘运算。

1 椭圆曲线点乘

考虑方程 $Q = kP$,k 是一个数,给定 k 和椭圆曲线上的点 P 可以由方程很容易计算出曲线上的另一点 Q,但是给定 Q 和 P 确定 k 就非常困难,这就是椭圆曲线构造一个密码系统的数学“难题”,称之为椭圆曲线离散对数问题(ECDLP)。ECC 中最重要的运算是点乘($Q = kP$),其中签名需要一次点乘运算,签名验证需要二次点乘。由 López 提出的基于射映坐标改进的 Montgomery 点乘算法中,一次点乘需要作 $6\log_2 k + 10$ 次域上的乘法和 1 次求逆[5]。

对于椭圆曲线:$y^2 + xy = x^3 + ax^2 + b$,求椭圆曲线点乘 $Q = kP = (x_2, y_2)$,其中 $P = (x_1, y_1)$,$k = (k_{n-1}\cdots k_1 k_0)_2$,则算法描述如下:

1 $X_1 = x_1$, $Z_1 = 1$;
2 $X_2 = x_1^4 + b$, $Z_2 = x_1^2$;
3 for $i = n - 2$ downto 0
3.1 if $k_i = 1$
3.1.1 $Z_3 = (X_1 Z_2 + X_2 Z_1)^2$, $X_3 = x_1 Z_3 + (X_1 Z_2)(X_2 Z_1)$,
$X_1 = X_3$, $Z_1 = Z_3$;

收稿日期:2006 - 03 - 21; **修订日期**:2006 - 06 - 05
基金项目:广州市属高校科技计划资助项目(2003); 广东省科技计划资助项目(2005B10101024)
作者简介:邹候文(1973 -),男,广东增城人,实验师,硕士,主要研究方向:嵌入式系统、信息安全; 王峰(1978 -),助教,硕士,主要研究方向:信息安全; 唐屹(1968 -),教授,博士,主要研究方向:信息安全.

3.1.2 $X_3 = X_2^4 + bZ_2^4$, $Z_3 = Z_2^2 X_2^2$, $X_2 = X_3$, $Z_2 = Z_3$;

3.2 else

3.2.1 $Z_3 = (X_1Z_2 + X_2Z_1)^2$, $X_3 = x_1Z_3 + (X_1Z_2)(X_2Z_1)$, $X_2 = X_3$, $Z_2 = Z_3$;

3.2.2 $X_3 = X_1^4 + bZ_1^4$, $Z_3 = Z_1^2 X_1^2$, $X_1 = X_3$, $Z_1 = Z_3$;

4 $x_2 = X_1/Z_1$

5 $y_2 = (X_1 + x_1)\{[(x_1Z_1 + X_1)(x_1Z_2 + X_2) + (x_1^2 + y)Z_1Z_2]/x_1Z_1Z_2\} + y_1$;

6 输出:$Q = (x_2, y_2)$;

NIST 推荐的 Koblitz 曲线的 a 和 b 都为 1,因此上述算法中的$bZ_2^4 = Z_2^4$,从而使整个算法减少了 $n-2$ 次域上的乘法。由于域元采用正规基表示,平方运算比较容易且多次连续的平方也是一次循环移位,在硬件实现中其资源和时间开销可以忽略。根据费马小定理和文献[6]所述可作如下推导:

$$a^{-1} = a^{2^{163}-2} = a^{2(2^{162}-1)} = \left(a^{2^{162}-1}\right)^2$$

$$a^{(2^{162}-1)} = a^{(2^{81}-1)(2^{81}+1)} = a^{(2^{81}-1)} * a^{(2^{81}-1)2^{81}}$$

$$a^{(2^{81}-1)} = a * a^{(2^{81}-2)} = a * a^{2(2^{80}-1)} = a * a^{(2^{80}-1)^2}$$

$$a^{(2^{80}-1)} = a^{(2^{40}-1)(2^{40}+1)} = a^{(2^{40}-1)} * a^{(2^{40}-1)2^{40}}$$

$$a^{(2^{40}-1)} = a^{(2^{20}-1)(2^{20}+1)} = a^{(2^{20}-1)} * a^{(2^{20}-1)2^{20}}$$

$$a^{(2^{20}-1)} = a^{(2^{10}-1)(2^{10}+1)} = a^{(2^{10}-1)} * a^{(2^{10}-1)2^{10}}$$

$$a^{(2^{10}-1)} = a^{(2^5-1)(2^5+1)} = a^{(2^5-1)} * a^{(2^5-1)2^5}$$

$$a^{(2^5-1)} = a * a^{(2^5-2)} = a * a^{(2^4-1)} = a * a^{(2^4-1)^2}$$

$$a^{(2^4-1)} = a^{(2^2-1)(2^2+1)} = a^{(2^2-1)} * a^{(2^2-1)2^2}$$

$$a^{(2^2-1)} = a^3 = a^2 * a$$

因此可以通过 9 次域上的乘法实现 $GF(2^{163})$ 上的求逆:$Inverse(a) = a^{-1} = a^{2^{163}-2}$。

$u_1 = aa^2$ $u_2 = u_1 u_1^{2^2}$

$u_1 = a u_2^2$ $u_2 = u_1 u_1^{2^5}$

$u_1 = u_2 u_2^{2^{10}}$ $u_2 = u_1 u_1^{2^{20}}$

$u_1 = u_2 u_2^{2^{40}}$ $u_2 = a u_1^2$

$u_1 = u_2 u_2^{2^{81}}$ $a^{-1} = u_1^2$

2 椭圆曲线点乘 IP 核

我们采用状态机控制方式实现椭圆曲线点乘 IP 核,最重要的运算部件是域上的乘法器,整个状态机的设计思想是把整个点乘运算中的域上乘法按顺序由乘法器运算得到结果,并尽量使其他运算(域上的加法、乘方等)在乘法器运行过程中穿插运算。

2.1 点乘运算有限状态机

从点乘算法中可以看到整个运算过程中的乘法数据具有相关性,因此采用乘法流水线或多个乘法器并行对加快运算速度的意义不大,因此所有乘法运算都由一个乘法器完成,其存储分配和状态转换描述如下:

存储器调度

R0:k R1:x R2:y R3:$X1$ R4:$Z1$ R5:$X2$;

R6:$Z2$ R7:$T1$ R8:$T2$ R9:$T3$ R10:$T4$;

s1:锁存 k 和 $P(x,y)$;

s2:初始化 R3:$X1 = x$, R4:$Z1 = 1$, R5:$X2 = x\hat{}4 + b$, R6:$Z2 = x\hat{}2$, $n = 162$;

s3:IF $R0(n)$ = '1' THEN GOTO s5; ELSE GOTO s4;

s4:IF $n = 0$ THEN 输出无穷远点并结束; ELSE $n = n - 1$,GOTO $s3$;

s5:IF $n = 0$ THEN GOTO sm6; ELSE $n = n - 1$;

sM1l:$T1 = X1 * Z2$;

sM2l:$T2 = X2 * Z1$;

sM3l:$T3 = T1 * T2$;

s3al:$T4 = T1 + T2$;

s3bl:$T4 = T4\hat{}2$;

sM4l:$T1 = x * T4$;

s4wal:$T2 = T1 + T3$;

s4putl:IF $KI = 1$ THEN $X1 = T2, Z1 = T4, T1 = X2\hat{}2, T3 = Z2\hat{}2$; ELSE $X2 = T2, Z2 = T4, T1 = X1\hat{}2, T3 = Z1\hat{}2$;

sM5l:$T2 = T1 * T3$;

sm5al:$T1 = T1\hat{}2, T3 = T3\hat{}2$;

sm5bl:$T4 = T1 + T3$;

sm5putl:IF $R0(n) = 1$ THEN $Z2 = T2, X2 = T4$; ELSE $Z1 = T2, X1 = T4$;

IF $n = 0$ THEN GOTO sm6; ELSE $n = n - 1$, GOTO sm1;

sm6:$T1 = x$, $T2 = y$, $T3 = Z1 * Z2$;

sm7:$Z1 = Z1 * T1$;

sm8:$Z1 = Z1 + X1$, $Z2 = Z2 * T1$;

sm9:$X1 = Z2 * X1$, $Z2 = Z2 + X2$;

sm10:$Z2 = Z2 * Z1$, $T4 = T1\hat{}2$, $T4 = T4 + T2$;

sm11:$T4 = T4 * T3$;

sm12:$T4 = T4 + Z2$, $T3 = T3 * T1$;

-- FOR $inverse(T3)$ BEGIN

smi13:$R1 = T3 * T3\hat{}2$;

smi14:$R2 = (R1 * R1\hat{}2)\hat{}2$;

smi15:$R1 = T3 * R2\hat{}2$;

smi16:$R2 = R1 * (R1\hat{}2)\hat{}5$;

smi17:$R1 = R2 * (R2\hat{}2)\hat{}10$;

smi18:$R2 = R1 * (R1\hat{}2)\hat{}20$;

smi19:$R1 = R2 * (R2\hat{}2)\hat{}40$

smi20:$R2 = T3 * R1\hat{}2$;

smi21:$R1 = R2 * (R2\hat{}2)\hat{}81$;

smi21wa:$T3 = R1\hat{}2$;

-- FOR $inverse(T3)$ END

sm22:$T4 = T3 * T4$;

sm23:$X2 = X1 * T3$;

sm23wa:$Z2 = X2 + T1$;

sm24:$Z2 = Z2 * T4$;

smend:$Z2 = Z2 + T2$;

smout:输出 $R5$ 为 $x2$, $R6$ 为 $y2$;

从算法中可以看到,整个点乘的运算时间由 k 的二进制位数确定,若 k 的比特数为 n,则其运算时间为 $5 + (n-1) * (5M + 6) + 19M + 4$ 个时钟周期(M 为乘法器的时钟周期数)。

2.2 正规基乘法器的设计

如果域 $GF(2^m)$ 上在 $GF(2)$ 的基 N 形如 $|\beta, \beta^{2^1}, \cdots, \beta^{2^{m-2}}, \beta^{2^{m-1}}|$,则称 N 为 $GF(2^m)$ 上的正规基。用正规基表示的

域元的乘法运算较为简单、有效。设 $a,b,c \in GF(2^m)$,c 是两个域元的乘积,$\alpha = (^{2^{m-1}}\cdots\alpha^{2^2}\alpha^2\alpha)$ 是正规基的基底。

$$a = (a_{m-1}\cdots a_1\alpha_0) = a_{m-1}\alpha^{2^{m-1}} + \cdots + a_1\alpha^2 + a_0\alpha = a\alpha$$

$$b = (b_{m-1}\cdots b_1b_0) = b_{m-1}\alpha^{2^{m-1}} + \cdots + b_1\alpha^2 + b_0\alpha = b\alpha$$

$$c = (c_{m-1}\cdots c_1c_0) = a\alpha(b\alpha)^T = a(\alpha\alpha^T)b^T = aMb^T$$

其中,T 表示向量的转置。

根据文献[7],高斯正规基乘法规则描述如下:

输入:整数 $m > 1$ 和 T,T 是 $GF(2^m)$ 具有 T 类高斯正规基 B 的类型(我们采用的曲线的 $T = 4$)。

输出:$F(a, b)$。

1　$p = Tm + 1$

2　计算 u,其中 $u^T \equiv 1 \bmod p$

3　按照如下方式计算序列 $F(1), F(2), \cdots, F(p-1)$:

3.1　$w = 1$

3.2　For j from 0 to $T-1$ do

　　$n = w$

　　For i from 0 to $m-1$ do

　　$F(n) = i$

　　$n = 2n \bmod p$

　　$w = uw \bmod p$

4　如果 T 是偶数,则 $J = 0$,

否则 $J := \sum_{k=1}^{m/2}(a_{k-1}\ b_{m/2+k-1} + a_{m/2+k-1}\ b_{k-1})$

5　输出公式:$c_0 = J + \sum_{k=1}^{p-2} a_{F(k+1)}\ b_{F(p-k)}$

根据文献[8]有

$$L_j(A,B) = \begin{cases} F_{m-1-jw}^{2^{w-1}} + F_{m-2-jw}^{2^{w-2}} + \cdots + F_{m-w-jw}, & 0 \leqslant j < k-1 \\ F_{w-1}^{2^{w-1}} + \cdots + F_0^{2^r}, & j = k-1 \end{cases}$$

其中:w 是字长,$k = \left\lfloor \frac{m}{w} \right\rfloor$,$r$ 满足 $m = wk - r$ 且 $0 \leqslant r < w$。

$$C^{2^r} = ((L_0^{2^w} + L_1)^{2^w} + \cdots + L_{k-2})^{2^w} + L_{k-1}$$

$$L_0(A,B) = F_{m-1}^{2^{w-1}}(A,B) + F_{m-1}^{2^{w-2}}(A^2,B^2) + \cdots + F_{m-1}(A^{2^{w-1}},B^{2^{w-1}})$$

$$F_i(A,B) = F_{k+i}(A^{2^k},B^{2^k}), \quad 0 \leqslant k,i \leqslant m-1$$

正规基乘法可描述为:

输入:$A,B \in GF(2^m)$

$$A = \sum_{i=0}^{m-1} a_i\beta^{2^i} = (a_{m-1},a_{m-2},\cdots,a_0)$$

$$B = \sum_{i=0}^{m-1} b_i\beta^{2^i} = (b_{m-1},b_{m-2},\cdots,b_0)$$

输出:$C = AB$。

1　初始化:$X = (a_{m-1}\cdots a_1a_0)$,　$Y = (b_{m-1}\cdots b_1b_0)$,　$D = (d_{m-1}\cdots d_1d_0) = 0$;

2　for $i = 0$ to $k-1$ {

2.1　$D = D^{2^w} + (F_{m-1}^{2^{w-1}}(A,B) + F_{m-1}^{2^{w-2}}(A^2,B^2) + \cdots + F_{m-1}(A^{2^{w-1}},B^{2^{w-1}}))$;

2.2　$X = X^{2^w}, Y = Y^{2^w}$; }

3　$D = D^{2^w} + (F_{m-1}^{2^{w-1}}(A,B) + F_{m-1}^{2^{w-2}}(A^2,B^2) + \cdots + F_{m-1}^{2^w}A^{2^{w-r-1}},B^{2^{w-r-1}})$;

4　$C = D^{2^{m-r}}$;

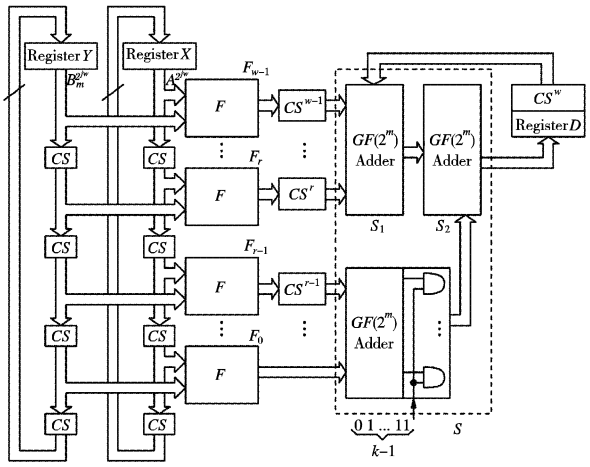
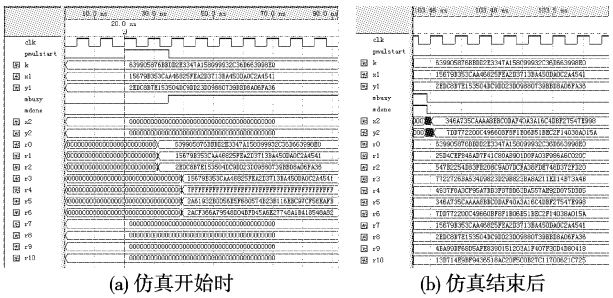


图 1　字长为 w 的正规基乘法器

2.3　实验结果

在设计过程中,先把点乘算法按适于硬件实现和调试的方式重新描述,然后按描述编写 C 语言程序,在 C 语言程序调试出正确结果后再用 VHDL 语言编码。考虑到资源消耗和关键路径延时等因素,取乘法器的字宽 $w = 16$,每 15 个时钟完成一次域上乘法。用 Quartus Ⅱ 5.0 综合,在 EP2S90F1508C3 器件下最高频率 166MHz,消耗 14 502 个 ALUTs,其仿真时序波形如图 2 所示。



(a) 仿真开始时　　(b) 仿真结束后

图 2　ECC 点乘 IP 核时序仿真波形图

图 2 的仿真时钟周期为 125MHz,pmulstart 为启动信号,输入数据为 k 和 $P(x1,y1)$。点乘器运算期间输出 mbusy 为高电平,完成后 mdone 给出一个时钟周期的高电平后把点乘的结果从 $x2,y2$ 输出。由图中可以看出,每秒可完成 9 665 次点乘,按最高时钟频率 166MHz 算,则每秒钟可完成 12 835 次点乘。而文献[9]在 XC2V8000-5 器件中消耗 18 079 个 Slices,乘法器字宽 w = 32,最高时钟频率 90.2MHz,每秒钟可完成 9 433 次点乘。

3　结语

本文从 ECC 硬件实现的角度,选择 NIST 推荐的 $GF(2^{163})$ 上的 Koblitz 曲线,采用 López 提出的基于射映坐标改进 Montgomery 点乘,并把求逆通过 9 次域上乘法实现,提出一种有限状态机控制的 ECC 点乘实现方案,较为详细地介绍了实现过程。实验结果表明本文设计的 IP 核具有较高的效率,与主处理器结合,可快速完成椭圆曲线密码的加密、解密、签名和验证,并可作为 IP 模块嵌入到其他 IP 或 IC 设计和应用中。

参考文献:

[1]　FIPS 186-2 of NIST. DIGITAL SIGNATURE STANDARD(DSS)[S],2000.

(下转第 2136 页)

地址上等待被调用? 在此,为了解决这个问题,程序使用了NT 系统的一个特性,即,kernel32. dll 和 user32. dll 总是被系统映射到相同的内存地址[3]。也就是说,这两个 DLL 中的函数在每个任务空间中的地址总是相同的。因此本文程序在这些地址范围中去寻找一个肯定会存在的语句 ret(DLL 中由许多函数组成,每个函数结束必定会有返回语句)。然后,将它的地址填入到调用门中。这样一来,不仅保证了在任务中每次 CALL 调用门时都能够找到所要执行的语句,而且,更重要的是,当执行 ret 语句时,由 CALL 语句所引起的已入栈的当前地址会被弹出,因此,从 CALL 语句后直到段间返回 retf 语句间的程序段就会运行于核心态中。

3.2 钩挂 Windows API 函数

进入核心态之后,除了修改 CR0,取消 COW 机制之外,主要的工作就是要对有关的系统 DLL 函数安装 Hook。在本例中主要是要监视 SetWindowsHookEx 和 CreateRemoteThread 两个系统 API 函数。下面以对 CreateRemoteThread 函数的钩挂为例说明此技术的实现。

一直以来,有两种钩挂 API 函数的方案,第一,找到 API 函数入口点并修改最前面的几个字节为跳转语句到新的代码上,然后视情况返回到原函数代码中; 第二,通过修改模块的引入或引出表来改变 API 函数的入口地址到新的代码段中。在抢先式多任务的 32 位 Windows 系统中,使有后一种方法比较安全[5]。

为了改变 API 函数入口地址,就必须找到其入口地址所在的位置。作为 API 函数的 CreateRemoteThread 存在于 kernel32. dll 中,而动态链接库是通过一组输出函数地址表向系统提供服务的。因此,搞清楚 DLL 的结构是实现钩挂函数的关键。

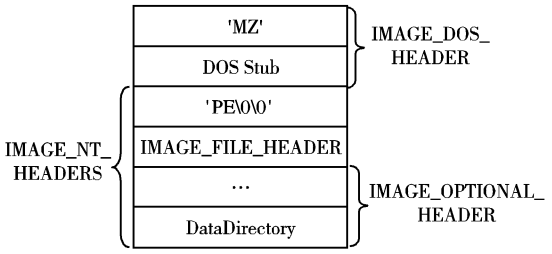


图 2 PE 文件头格式

WindowsNT 系统中,动态链接库文件是作为 PE 文件格式进行存储的。如图 2 所示,PE 文件的文件头中主要由图中所标注的三部分内容组成。其中 IMAGE_FILE_HEADER 结构的最后一个域说明了此模块是可执行文件还是动态链接库文件。图 2 中所示的 IMAGE_OPTIONAL_HEADER 结构的最后一个域 DataDirectory 是一个成员为 IMAGE_DATA_DIRECTORY 结构的结构数组,共有 16 个成员。而 MAGE_DATA_DIRECTORY 结构有两个成员组成,其一为所指示特定数据的 RVA(相对虚拟地址),其二为此特定数据的大小信息。在此结构数组中第一个成员指向的是此模块的函数引出表。

函数引出表结构中,有三个域同最终找到具体的引出函数密切相关,它们分别是:

1) AddressOfNames 域,此域位于引出表相对偏移 20H 处,它指向一组 DWORD 类型的数组,其中每个 DWORD 代表了一个 ASCIIZ 字符串的 RVA,这个字符串就是此模块所提供函数的函数名,遍历此数组可得到此模块的引出的所有函数的函数名; 2) AddressOfNameOrdinals 域,此域位于引出相对偏移 24H 处,它也指向一组 DWORD 类型的数组,该数组的每项与引出表的 Base 域的值相减,就得到对应上个域的相同偏移位置的那个函数的函数地址表的索引值; 3) AddressOfFunctions 域,此域位于引出表相对偏移 1CH 处,它同样也指向一组 DWORD 类型的数组,它的每项对应的就是索引值所代表函数的引出地址[6]。

需要特别说明的是,上述很多域使用了相对虚拟地址(RVA),而在实际使用中模块的基地址可用 GetModuleHandleA 函数得到。

4 结语

通过对上述关键技术的使用,本文实例在 XP 系统下成功的钩挂到了 kernel32. dll 所提供的 CreateRemoteThread 函数,并对其进行了屏蔽处理。使进程无法直接利用此函数启动远程线程,实现代码注入。

参考文献:

[1] ROBERT KUSTER. Three ways to inject your code. into another process[EB/OL]. http://www. codeproject. com/threads/winspy. asp.

[2] SUNWEAR. MGF 病毒技术分析[EB/OL]. http://blog. csdn. net/sunwear.

[3] RICHTER J. Windows 核心编程[M]. 北京 机械工业出版社, 2000. 299 - 309.

[4] 杨季文. 80X86 汇编语言程序设计教程[M]. 北京 清华大学出版社, 2000. 361 - 421.

[5] 冉光志, 陈旭春, 练锴, 等. Visual C++ 应用技巧与常见问题[M]. 北京: 机械工业出版社, 2003. 182 - 189.

[6] LUEVELSMEYER PE 文件格式 [EB/OL]. http://www. pediy. com/tutorial/chap8/Chap8-1-6.

(上接第 2133 页)

[2] BROWN M, HANKERSON D, LOPEZ J, *et al*. Software Implementation of the NIST Elliptic Curves Over Prime Fields[A]. Proc. of CT-RSA'2001[C]. Springer-Verlag, LNCS 2020, 2001. 250 - 265.

[3] HASAN MA, WANG MZ, BHARGAVA VK. A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields[J]. IEEE Transactions on Computers, 1993, 42(11): 1278 - 1280.

[4] ANSI X9. 62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)[S], 1999.

[5] LÓPEZ J, DAHAB R. Fast multiplication on elliptic curves over GF(2m) without precomputation [A]. CHES'99 Workshop on Cryptographic Hardware and Embedded Systems[C]. Lecture Notes in Computer Science 1717. Springer-Verlag, 1999.

[6] ITOH T, TSUFII S. A fast algorithm for computing multiplicative inverse in GF(2m) using normal bases[J]. Information and Computation, 71998, 8(3): 171 - 177.

[7] IEEE P1363/D13, Standard Specifications for Public Key Cryptography[S], 1999. 93 - 94.

[8] REYHANI - MASOLEH A, HASAN MA. Low Complexity Word -Level Sequential Normal Basis Multipliers[J]. IEEE transactions on computers, 2005, 54(2): 98 - 110.

[9] JARVINEN K, TOMMISKA M, SKYTTA J. A scalable architecture for elliptic curve point multiplication[A]. 2004 IEEE International Conference on Field-Programmable Technology, ICFPT 2004[C], 2004. 303 - 306.