

# 基于双基表示的并列点乘算法

鲍皖苏, 陈 辉

(解放军信息工程大学电子技术学院, 河南郑州 450004)

**摘 要:** 双基表示是一种整数表示法, 它将任意整数表示成 2 和 3 的混合幂次的和或差的形式, 并列点乘是一种快速的点乘算法, 应用于一些椭圆曲线密码体制中. 本文在现有的双基表示算法以及并列点乘算法的基础上, 提出了新的双基表示算法以及基于该双基表示算法的并列点乘算法, 该算法利用了一些特殊点的快速计算公式, 从而有效地提高了并列点乘算法的执行效率. 实验表明, 在密钥长度为 160 比特,  $[S]/[I] = 0.8$  时, 当  $[I]/[M] = 30$ , 新算法的效率比基于 JSF 表示的并列点乘算法提高了 22%; 当  $[I]/[M] = 10$ , 新算法比 JSF 表示提高了 6%; 当  $[I]/[M] = 8$ , 新算法比 JSF 表示提高了 3%.

**关键词:** 椭圆曲线; 并列点乘; DBNS; 标量乘法

**中图分类号:** TN918.1 **文献标识码:** A **文章编号:** 0372-2112 (2009) 04-0873-04

## A New Simultaneous Scalar Multiplication Based on Double Base Number System

BAO Wansu, CHEN Hui

(Institute of Electronic Technology, the PLA Information Engineering University, Zhengzhou, Henan 450004, China)

**Abstract:** Double base number system is a representation of the integers as the sum of mixed powers of two and three. Simultaneous scalar multiplication is a fast point multiplication algorithm, and applied to some elliptic curve cryptography systems. Based on the original double base system and simultaneous scalar multiplication, a new double base system algorithm and simultaneous scalar multiplication algorithm based on this double base system algorithm is presented in this paper. Due to the fast underlying fields arithmetic, the new simultaneous scalar multiplication algorithm enhances simultaneous scalar multiplication efficiently. Experiments prove that, with the 160 bits secret key and  $[S]/[I] = 0.8$ , our method performed 22% improvement better than JSF when  $[I]/[M] = 30$ , 6% when  $[I]/[M] = 10$ , 3% when  $[I]/[M] = 8$ .

**Key words:** elliptic curve; simultaneous scalar multiplication; double base number system; scalar multiplication

## 1 引言

1985 年 N. Koblitz 和 V. Miller 分别提出椭圆曲线密码体制(ECC), 此后椭圆曲线密码体制(ECC)就以其短密钥, 高安全性得到了人们的重视, 并广泛应用于信息安全领域. 标量乘法(或点乘运算)是椭圆曲线公钥体制实现过程中最基本的也是关键的环节之一, 其运算效率直接影响着整个密码系统的实现效率. 传统的点的标量乘法运算由反复的点加和倍点运算构成, 2005 年 V. Dimitrov 和 G. A. Jullien 提出了基于双基链的标量乘法<sup>[1]</sup>, 将  $2P \pm Q, 3P, 3P \pm Q, 4P$  和  $4P \pm Q$  的特殊计算公式与双基表示法(DBNS)融合, 有效地减少了基域上的运算量, 加快了标量乘法的实现效率.

在椭圆曲线公钥密码体制应用中, 如 ECDSA, 常常需要计算两个点乘的和, 即并列点乘  $k_1P + k_2Q$ , 因此, 提高并列点乘实现效率对椭圆曲线密码体制应用十分重要. 现有的实现并列点乘的算法有: 基于二进制的并

列点乘, 基于 NAF 的并列点乘和基于联合稀疏型(JSF)的并列点乘, 它们的点加次数取决于  $k_1, k_2$  的平均联合汉明重量. 本文提出了一种新的双基表示算法, 并基于该双基表示算法, 给出了一种新的并列点乘算法. 实验证明基于双基表示的并列点乘算法能够有效地提高并列点乘运算效率.

## 2 双基表示及双基表示点乘算法

**定义 1<sup>[1]</sup>** (s-integer)  $s$ -整数是指最大素数因子不超过第  $s$  个的素数的整数.

**定义 2<sup>[1]</sup>** (double base number system) DBNS 是一个代表系, 任意整数  $k$  都可以表示为 2-整数的和或差的形式, 即  $k = \sum_{i=1}^m s_i 2^{b_i} 3^{t_i}$ , 其中  $s_i \in \{-1, 1\}$ ,  $b_i, t_i \geq 0$ .

**定理 1<sup>[2]</sup>** 任意整数  $k$  都可以表示成至多  $O\left(\frac{\log k}{\log \log k}\right)$  个 2-整数之和.

任意一个整数都有多种双基表示形式,例如

$$127 = 2^2 3^3 + 2^1 3^3 + 2^0 3^0 = 2^2 3^3 + 2^1 3^0 + 2^0 3^1 \\ = 2^5 3^1 + 2^0 3^3 + 2^3 3^0$$

一般不同的双基表示形式由不同双基表示算法得到,基于贪婪算法的双基表示算法(算法1)具有所得到的双基表示的2-整数个数较少,易于实现等特点,成为目前使用最多的一种双基表示算法。

基于贪婪算法的双基表示算法[1]

Input *irbit* 正整数  $k$ ;  $b_{\max}$ ,  $t_{\max} > 0$ , 分别为 2 和 3 指数可能取得的最大值

Output 序列  $(s_i, b_i, t_i)_{i \geq 0}$  使得  $k = \sum_{i=1}^m s_i 2^{b_i} 3^{t_i}$  成立, 且  $b_1 \geq \dots \geq b_m \geq 0$ ,  $t_1 \geq \dots \geq t_m \geq 0$

```

1.  $s \leftarrow 1$ 
2. while  $k > 0$  do
3.   define  $z = 2^{b_3^t}$ , the best approximation of  $k$  with  $0 \leq b \leq b_{\max}$  and  $0 \leq t \leq t_{\max}$ 
4.   print  $(s, b, t)$ 
5.    $b_{\max} \leftarrow b$ 
6.    $t_{\max} \leftarrow t$ 
7.   if  $k < z$  then
8.      $s \leftarrow -s$ 
9.    $k \leftarrow |k - z|$ 
```

双基表示作为一种整数表示法,与  $2P$ 、 $2P \pm Q$ 、 $3P$ 、 $3P \pm Q$ 、 $4P$  和  $4P \pm Q$  等的特殊计算公式(分别记为  $D$ ,  $DA$ ,  $T$ ,  $TA$ ,  $Q$ ,  $QA$ )结合,得到基于双基数链的点乘算法[1],大大提高了标量乘法运算效率。那么能否将双基表示系统应用于并列点乘呢?本文对此进行了研究,提出了基于双基表示算法的并列点乘算法。

### 3 新的双基表示算法以及基于该算法的并列点乘算法

#### 3.1 新的双基表示算法

实验证明通过算法1得到的双基表示形式结合特殊计算公式用于并列点乘运算不能使其运算效率优于现有算法。在密钥长度为 160 比特,  $[S]/[I] = 0.8$  时,当  $[I]/[M] = 30$  该算法的基域运算量比基于 JSF 表示的并列点乘算法多出了 58%;当  $[I]/[M] = 10$ , 该算法比 JSF 表示多出了 39%;当  $[I]/[M] = 8$ , 该算法比 JSF 表示多出了 24%。因为通过算法1将  $k_1 P + k_2 Q$  中  $k_1$ ,  $k_2$  表示成双基表示形式后,无法利用预计算有效提高并列点乘运算效率。本节提出了一种新的双基表示算法(算法2),通过该算法求得  $k_1$ ,  $k_2$  的双基表示形式在进行并列点乘时适合进行预计算。

算法通过如下方法得到的双基表示形式:(1)若  $k \geq 2^{t_{\max}} 3^{t_{\max}}$  成立,则取  $z = 2^{b_3^t} 3^{t_{\max}} \leq k$ ,  $t_{\max} \leq b \leq b_{\max}$ ,使得对于任意  $z' = 2^{b'} 3^{t_{\max}}$ ,都有  $k - z \leq k - z'$  成立;并令  $k = k - z$ ;(2)若  $k < 2^{t_{\max}} 3^{t_{\max}}$  成立,则必然存在  $t$ ,  $0 \leq t < t_{\max}$ ,使得  $2^{t_3^t} \leq k < 2^{t+1} 3^{t+1}$  成立,令  $z_0 = 2^t 3^t$ ,根据  $k$  所在区间进行赋值:(i)若  $3^* z_0 \leq k < 2^* 3^* z_0$ ,则令  $z = 3z_0$ ,  $k = k - z$ ;(ii)若  $2^* z_0 \leq k < 3^* z_0$ ,则令  $z = 3z_0$ ,  $k = k - z$ ;(iii)若  $z_0 \leq k < 2^* z_0$ ,则令  $z = z_0$ ,  $k = k - z$ 。新的双基表示算法(算法2)

新的双基表示算法

Input: 整数  $k$ ;  $b_{\max}$ ,  $t_{\max}$  分别是 2 和 3 的指数允许的最大值,且都大于 0

Output: 序列  $(s_i, b_i, t_i)_{i \geq 0}$  使得  $k = \sum_{i=1}^m s_i 2^{b_i} 3^{t_i}$  且  $b_0 \geq b_1 \geq \dots \geq b_m \geq 0$

$t_0 \geq t_1 \geq \dots \geq t_m \geq 0$

1.  $s \leftarrow 1$ ;

2. while ( $k > 0$ ) do

3. if ( $k > 2^{t_{\max}} 3^{t_{\max}}$ ) then

4. define  $z = 2^{b_3^t} 3^{t_{\max}}$ , the largest integer less than or equal to  $k$  with  $t_{\max} \leq b \leq b_{\max}$ ;

5. print  $(s, b, t)$ ;

6.  $k \leftarrow k - z$ ;

7. else

8. define  $z_0 = 2^t 3^t$ , the largest integer less than or equal to  $k$  with  $0 \leq t < t_{\max}$ ;

9. if ( $3^* z_0 \leq k < 2^* 3^* z_0$ ) then

10. define  $z = 3z_0$ ;

11. print  $(s, t, t + 1)$ ;

12.  $k \leftarrow k - z$

13. if ( $2^* z_0 \leq k < 3^* z_0$ ) then

14. define  $z = 3z_0$ ,  $s = -s$ ;

15. print  $(s, t, t + 1)$ ;

16.  $k \leftarrow k - z$ ;

17. if ( $z_0 \leq k < 2^* z_0$ ) then

18. define  $z = z_0$ ;

19. print  $(s, t, t)$ ;

20.  $k \leftarrow k - z$ ;

下面证明算法2的表示方法是合理的:

(I) 若  $k \in [2^{t_{\max}} 3^{t_{\max}}, 2^{b_{\max}} 3^{t_{\max}})$ , 则  $\exists t_{\max} \leq i \leq b_{\max}$  使得  $2^{i_3^t} 3^{t_{\max}} \leq k < 2^{i+1} 3^{t_{\max}}$  成立, 此时  $z = 2^i 3^{t_{\max}}$ , 由  $k < 2^{i+1} 3^{t_{\max}}$  可得:  $k - 2^i 3^{t_{\max}} < 2^{i_3^t} 3^{t_{\max}}$ ,  $t_{\max} \leq i \leq b_{\max}$ ;

(II) 若  $k \in [2^{t_{\max}} 3^{t_{\max}}, 2^{b_{\max}} 3^{t_{\max}})$ , 则  $\exists i < t_{\max}$  使得  $2^{i-1} 3^{i-1} \leq k < 2^i 3^i$  成立, 此时

$$z = \begin{cases} 2^{i-1} 3^i & s = s \quad \text{当 } 2^{i-1} 3^i \leq k < 2^i 3^i \\ 2^{i-1} 3^i & s = -s \quad \text{当 } 2^i 3^{i-1} \leq k < 2^{i-1} 3^i \\ 2^{i-1} 3^{i-1} & s = s \quad \text{当 } 2^{i-1} 3^{i-1} \leq k < 2^i 3^{i-1} \end{cases}$$

若  $2^{i-1} 3^i \leq k < 2^i 3^i$ , 同上由于  $k < 2^i 3^i$ , 所以  $k - 2^{i-1} 3^i < 2^{i-1} 3^i$ ; 同理可证若  $2^{i-1} 3^{i-1} \leq k < 2^i 3^{i-1}$ ,  $k - 2^{i-1} 3^{i-1} < 2^{i-1} 3^{i-1}$ ; 若  $2^i 3^{i-1} \leq k < 2^{i-1} 3^i$ , 因为  $k > 2^{i-1} \cdot 3^{i-1}$ , 所以  $2^{i-1} 3^i - k < 2^i 3^{i-1}$ . 故算法2的表示方法是合理的。

例1:  $314159 = 2^{73} 3^7 + 2^{53} 3^6 + 2^{53} 3^5 + 2^{43} 3^5 - 2^{33} 3^4 - 2^{23} 3^3 - 2^{13} 3^2 + 2^{03} 3^1 - 2^{03} 3^0$ .

#### 3.2 基于算法2的并列点乘算法

本节提出了一种基于算法2的并列点乘算法(算法3),该算法结合特殊计算公式与少量预计算,提高了并列点乘的执行效率。实验证明,新的并列点乘算法优于现有并列点乘算法。

定义3 (标量数) 标量乘法  $kP$  中的  $k$ , 称为标量数。

定义4 (合并排序运算) 设  $k_1$  和  $k_2$  为两个正整数,通过算法2得  $k_1$  和  $k_2$  的双基表示形式:

$k_1P = \sum_{i=0}^n s_i 2^{b_i} 3^{t_i} P$ ,  $k_2Q = \sum_{j=0}^m s_j 2^{x_j} 3^{y_j} Q$ , 记:  $R_1 = P$ ,  $R_2 = Q$ ,  $R_3 = 2P$ ,  $R_4 = 2Q$ ;  $R_5 = 2P + Q$ ,  $R_6 = 2Q + P$ ,  $R_7 = P - 2Q$ ,  $R_8 = 2P - Q$ ,  $R_9 = 2P + 2Q$ ,  $R_{10} = P + Q$ ,  $R_{11} = P - Q$ ,  $R_{12} = 2P - 2Q$ .

定义合并排序运算由以下两步构成:

(1) 合并: 将  $k_1P + k_2Q$  以如下形式表示:

$$k_1P + k_2Q = \sum_{i=0}^n s_i 2^{b_i} 3^{t_i} P + \sum_{j=0}^m s_j 2^{x_j} 3^{y_j} Q$$

合并标量数绝对值相同的项, 得到

$$k_1P + k_2Q = \sum_{i=0}^{n'} s_i 2^{b_i} 3^{t_i} R_{w'_i} + \sum_{j=0}^{m'} s_j 2^{x_j} 3^{y_j} R_{w'_j}$$

$w'_i, w'_j \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ .

(2) 重新排序: 将  $k_1P + k_2Q = \sum_{i=0}^{n'} s_i 2^{b_i} 3^{t_i} R_{w'_i} +$

$\sum_{j=0}^{m'} s_j 2^{x_j} 3^{y_j} R_{w'_j}$  的各项按 2 的指数从大到小进行重新排列, 2 的指数相同的项则按照 3 的指数从大到小排列. 因此  $k_1P + k_2Q$  可表示成如下形式:

$$k_1P + k_2Q = \sum_{i=0}^{n'+m'} s_i 2^{b'_i} 3^{t'_i} R_{v_i} = \sum_{i=0}^{n'+m'} s_i 2^{b'_i} 3^{t'_i} R_{v_i},$$

$v_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ .

算法 3 基于算法 2 的并列点乘:

Input: 整数  $k_1 = \sum_{i=0}^n s_i 2^{b_i} 3^{t_i}$  和  $k_2 = \sum_{j=0}^m s_j 2^{x_j} 3^{y_j}$ , 其中  $k_1$  和  $k_2$  的表示形式都是通过算法 2 得到的;

$b_0 \geq b_1 \geq \dots \geq b_n \geq 0$ ,  $t_0 \geq t_1 \geq \dots \geq t_n \geq 0$ ;  $x_0 \geq x_1 \geq \dots \geq x_m \geq 0$ ,  $y_0 \geq y_1 \geq \dots \geq y_m \geq 0$ ;

并设  $b_0 \geq x_0$

Output:  $k_1P + k_2Q$

1. 预计算  $2P, 2Q, P + Q, P - Q, 2P - Q, P - 2Q, P + 2Q, 2P + Q, 2P + 2Q$ ;

令  $R_1 = P, R_2 = Q, R_3 = 2P, R_4 = 2Q, R_5 = 2P + Q, R_6 = 2Q + P, R_7 = P - 2Q, R_8 = 2P - Q, R_9 = 2P + 2Q, R_{10} = P + Q, R_{11} = P - Q, R_{12} = 2P - 2Q$ ;

2. 将  $k_1P + k_2Q$  写成  $\sum_{i=0}^n s_i 2^{b_i} 3^{t_i} P + \sum_{j=0}^m s_j 2^{x_j} 3^{y_j} Q$  形式, 进行合并排序运算得到:

$$k_1P + k_2Q = \sum_{i=0}^{n+m} s_i 2^{b_i} 3^{t_i} R_{v_i}, v_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

3.  $Z \leftarrow s_0 P$ ;

4. for  $i = 0 \dots l$  do

5.  $u \leftarrow b'_i - b'_{i+1}$ ;

6.  $v \leftarrow t'_i - t'_{i+1}$ ;

7. if ( $u = 0$ ) then

8.  $Z \leftarrow 3(3^{v-1} Z) + s_{i+1} P$ ;

9. else

10.  $Z \leftarrow 3^v Z$ ;

11.  $Z \leftarrow 4^{u-1/2} Z$ ;

12. if ( $u \equiv 0 \pmod{2}$ ) then

13.  $Z \leftarrow 4Z + s_{i+1} P$ ;

14. else

15.  $Z \leftarrow 2Z + s_{i+1} P$ ;

16. return  $Z$ .

算法 3 中的标量乘法  $2P, 2P \pm Q, 3P, 3P \pm Q$  和  $4P, 4P \pm Q$  都可以通过特殊计算公式<sup>[3]</sup>来提高运算效率. 需要说明一点<sup>[1]</sup>: 由于  $[I]/[M]$  在二元域和素数域上的值不同,  $4P$  的在这两种域上的特殊计算公式也不同. 若设  $[S]/[I] = 0.8$ , 容易验证在素数域上  $4P$  比  $2(2P)$  需要更少的基域运算; 而在二元域上,  $4P$  的实现方式则需要根据  $[I]/[M]$  的具体取值来判断; 当  $[I]/[M] < 4$  时,  $2(2P)$  比  $4P$  需要的基域运算更少. 因而, 在素数域上可以直接使用  $4P, 4P \pm Q$  来获得更快的速度, 而在二元域上则要根据  $[I]/[M]$  的具体值来确定实现方式. ( $[I], [S], [M]$  分别表示基域运算中的求逆, 平方, 乘法运算)

#### 4 算法效率分析

本节比较算法 3 与现有的并列点乘算法的实现效率. 首先分别在素数域和二元域上, 随机选取大量 160 比特的大整数  $k_1, k_2$ , 然后用算法 3 进行并列点乘运算, 最后计算它们所需的基域运算量平均值, 并同现有并列点乘算法进行比较. 下表列出了当密钥长度为 160 比特时, 由算法 3 进行并列点乘运算时的平均运算量.

表 1 素数域,  $|K| = 160$

$b_{\max}, t_{\max}$	$D$	$DA$	$T$	$TA$	$Q$	$QA$	$[I]$	$[S]$	$[M]$
76, 53	-	63	27	26	4	5	160	394	1081

表 2 二元域上,  $|K| = 160$

$b_{\max}, t_{\max}$	$D$	$DA$	$T$	$TA$	$[I]$	$[S]$	$[M]$
76, 53	7	68	27	26	154	336	1049

设  $k_1$  和  $k_2$  是两个  $l$  比特的随机整数, 基于二进制的并列点乘算法需要的平均运算量是:  $(l-1)D + \frac{3}{4}(l-1)A$ , 基于 NAF 的并列点乘算法需要的平均运算量是:  $(l-1)D + \frac{5}{9}(l-1)A$ , 基于 JSF 的并列点乘算法平均运算量是:  $(l-1)D + \frac{1}{2}(l-1)A$ .

对  $l = 160$  的情况进行分析可得: 若基点在素数域上选取, 且  $[I]/[M] = 30$ ,  $[S]/[I] = 0.8$  时, 新的算法效率比基于二进制表示的并列点乘提高 39%, 比基于 NAF 表示的并列点乘提高 33%, 比基于 JSF 表示的并列点乘提高 22%; 而  $[I]/[M] = 10$  时, 新算法比二进制表示提高 27%, 比 NAF 表示提高 19%, 比 JSF 表示提高 6%. 若基点在二元域上选取, 且  $[I]/[M] = 8$  时, 新算法比二进制表示提高 20%, 比 NAF 表示提高 17%, 比 JSF 表示提高 30%.

选取 NIST 推荐的二进制域上椭圆曲线  $B-163$ :  $m = 163, f(z) = z^{163} + z^7 + z^6 + z^3 + 1, a = 1, h = 2, S = 85E25BFE5C86226CDB12016F7553F9D0E693A268$ .

b= 20A601907B8C953CA1481EB10512F78744A3205FD  
n= 4000000000000000000292FE77E70C12A4234C33  
x= 3F0EBA16286A2D57EA0991168D4994637E8343E36  
y= D51FBC6C71A0094FA2CDD545B11C5C0C797324F1  
x<sub>1</sub>= 696CB09CBBFD41DDA168FF6841207E5C135C72ACB  
y<sub>1</sub>= 57225C8265D574971086b77E8E4487166F0DFCBEE  
m 是二进制域  $F_{2^m}$  的扩张次数;  $f(z)$  是次数为  $m$  的约减多项式;  $S$  是随机产生椭圆曲线系数所选的种子;  $a, b$  椭圆曲线  $y^2+xy=x^3+ax^2+b$  的系数;  $n$  基点  $P$  的素数阶;  $h$  为余因子;  $x, y$  为  $P$  的  $x$  坐标和  $y$  坐标;  $x_1, y_1$  为  $Q$  的  $x$  坐标和  $y$  坐标. 在 CPU 为 Pentium4 2.5GHz, 内存为 256MB, 操作系统为 WindowsXP 的个人电脑上, 借助 MIRCAL library version 5.0 随机选取 10000 对 163 比特的大整数  $k_1, k_2$ , 预求出它们的二进制, NAF, 联合稀疏型及算法 5.2 的双基表示, 测得基于二进制的并列点乘算法, 基于 NAF 的并列点乘算法, 基于 JSF 的并列点乘算法和新的并列点乘算法平均运行时间列表如下:

表 3 B-163 实验结果

基于二进制的 并列点乘算法	基于 NAF 的并 列点乘算法	基于 JSF 的 并列点乘算法	新并列点乘算法
69.52(ms)	65.01(ms)	57.61( ms)	55.99( ms)

5 总结

本文提出了新的求解双基表示的算法和并列点乘算法, 使得新的求解双基表示的算法能够适用于新的并列点乘算法. 实验证明新的并列点乘算法优于现有的并列点乘算法. 下一步的工作将是优化求解双基表示的算法, 减少表达式中非零元的个数, 从而进一步减

少并列点乘中的基域运算次数.

参考文献:

[1] Dimitrov V S, Imbert L, Mishra P K. Fast elliptic curve point multiplication using double base chain[ OL] . <http://eprint.iacr.org/2005/069>. ps. gz.  
[2] Dimitrov V S, Jullien G A and Miller W C. An algorithm for modular exponentiation [ J]. Information Processing Letters, 1998, 66(3): 155– 159.  
[3] Kirsten Eisentrager, Kristin Lauter, Montgomery P L. Fast elliptic curve arithmetic and improved Weil pairing evaluation[ A]. Topics in Cryptology CT-RSA2003[ C]. University of California, Berkeley LNCS. Springer Publishing Vol 2612, 2003. 343– 354.  
[4] Yasuyuki Sakai, Kouichi Sakurai. Algorithms for efficient simultaneous elliptic scalar multiplication with reduced joint hamming weight representation of scalars[ A]. ISC 2002, International conference on information security[ C]. Sao Paulo, Bresil, LNCS. Springer Publishing Vol 2433, 30, 2002. 484– 499.  
[5] Solinas J A. Low weight binary representations for pairs of integers[ R]. Technical Report CORR 2001-41, Center for Applied Cryptographic Research, University of Waterloo, Canada.

作者简介:

鲍皖苏 男, 1966 年 9 月生于安徽天长, 解放军信息工程大学电子技术学院教授, 博士研究生导师. 主要研究领域: 公钥密码理论与技术、量子计算算法、量子密码. E-mail: 2004bws@sina.com  
陈 辉 男, 1983 年 10 月生于湖北武穴, 解放军信息工程大学电子技术学院硕士研究生, 研究方向为椭圆曲线公钥密码.