第28卷 第6期 计 算 机 工 程 2002年6月
Vol.28 № 6 Computer Engineering June 2002
· 安全技术 · 文章编号：1000—3428(2002)06 —0111—03 文献标识码：A 中图分类号： TP309.7

NTRU公开密钥体制算法分析与实现

步山岳
(淮阴工学院计算机系，淮安 223001)

摘 要：介绍一种新的公开密钥体制NTRU。NTRU算法的安全性取决于从一个非常大的维数格中寻找很短向量的困难性，NTRU公开密钥体制算法主要计算对象是对多项式进行加、减、乘、模等运算。用NTRU产生的密钥方法比较容易，加密、解密的速度比RSA等著名算法快得多。从安全性和有效性方面分析，NTRU密码体制有着广阔的应用前景。

关键词：NTRU；乘逆；加密；解密

Analysis and Implementation on NTRU Public Key Cryptosystem

BU Shanyue
(Computer Department of Huaiyin Institute of Technology, Huai'an 223001)

【Abstract】 This paper introduces a new public key cryptosystem which is called NTRU algorithm. The security of the NTRU is based on the hard mathematical problem of finding very short vectors in lattices of very high dimension. The principal objects used by the NTRU PKCS are polynomial of degree N-1 having integer coefficient which are operated by addition,subtraction and multiplication etc. Encryption and decryption with NTRU are extremely fast and key creation is fast and easy. Because of their security and efficiency, the NTRU will be used widely in the future.

【Key words】 NTRU；Inversion；Encrytion；Decryption

NTRU (Number Theory Research Unit)公开密钥体制是由3位美国数学家发明的，经过几年的迅速发展与完善，该算法在密码学领域中受到了高度的重视并在实际应用中取得了很好的效果。NTRU算法的安全性是基于数论中在一个非常大的维数格中寻找一个很短向量的数学难题。就目前来说，NTRU的安全性和目前最有影响的RSA算法、椭圆曲线加密体制ECC等算法是一样安全的。在相同安全级的前提下，NTRU算法的速度要比其它公开密钥体制的算法快得多，用Tumbler 软件工具包执行NTRU时的速度比RSA快100多倍。用NTRU算法产生密钥的速度也很快，其密钥的bit数也较小。NTRU算法的优点意味着可以降低对带宽、处理器、存储器的性能要求，这也扩大了NTRU公开密钥体制的应用范围。

NTRU算法的数论基础是：

设有整数环 Z、整数 N≥2，用 R 表示多项式截断环(truncated polynomial rings)时，R可写成：

$$R = Z[X] / (X^N - 1)$$

对于任意正整数 q，令 R_q 代表模q的多项式截断环时，R_q可以写成：

$$R_q = (Z/qZ)[X] / (X^N - 1)$$

可证明当q是素数时,R_q具有可逆性。即对F,有F*满足：

$$F \cdot F^* \equiv 1 \bmod q \quad F, F^*$$

1 NTRU算法的描述

(1) NTRU算法参数

NTRU算法需要3个整数参数(N, p, q)和f、g、r、m具有的N 1 阶多项式环。在实际应用中为增加安全性，NTRU算法要求gcd (p，q)=1,并且p要远大于q。表1是NTRU技术根据不同的安全级推荐的参数。

其中d_f表示f的系数为1的个数，而-1的个数等于d_f-1，其余系数等于0。d_g、d_r分别表示g、r的系数为1和-1的个数都等于d_g、d_r，其余的系数等于0。

表1 NTRU算法参数

	N	p	q	d_f	d_g	d_r
一般安全	107	64	3	15	12	5
高安全	167	128	3	61	20	18
最高安全	503	256	3	216	72	55

(2)NTRU密钥的产生

根据NTRU参数N，首先随机选择两个多项式 f、g, 为使f对模p、模q的乘逆存在,f 首先应满足gcd(f,pq)=1 条件,我们分别用F_p、F_q代表f对模p、模q 的乘逆，即：

$$f \cdot F_p \equiv 1 \bmod p \quad \text{and} \quad f \cdot F_q \equiv 1 \bmod q \tag{1}$$

计算：

$$h = q \cdot F_p \cdot g \bmod p \tag{2}$$

NTRU算法取私人密钥为一对多项式环(f，F_q)，公开密钥为多项式环h。

(3)NTRU算法的加密过程

设有明文多项式m，根据参数d_r随机选择多项式r，使用公开密钥h，按公式(3)对明文m加密，得到密文e。

$$e = (r \cdot h + m) \bmod p \tag{3}$$

(4)NTRU算法的解密过程

用一对私人密钥(f，F_q)解密密文e，计算：

$$a = f \cdot e \bmod p \tag{4}$$

$$b = a \bmod q \tag{5}$$

$$c = F_q \cdot b \bmod q \tag{6}$$

多项式c就是解密后的明文m。

2 NTRU算法的实现

NTRU算法的计算对象是多项式环之间的加、减、模、

作者简介：步山岳(1960～)，男，硕士，研究方向：数据加密与计算机密码术

收稿日期：2001-08-02

乘和求逆等运算。多项式环之间的加、减运算与普通多项式的加、减运算完全相同。

(1)多项式环的乘法规则

设a、b是N-1次多项式，即 deg(a)=deg(b)=N-1, a,b∈ R。

$a = a_0 + a_1X + a_2X^2 + ... + a_{N-1}X^{N-1}$,

$b = b_0 + b_1X + b_2X^2 + ... + b_{N-1}X^{N-1}$

为便于编程和书写方便，我们将a、b改写成向量形式：

$a = (a_0, a_1, a_2, \cdots, a_{N-1})$， $b = (b_0, b_1, b_2, \cdots, b_{N-1})$

当a,b相乘时与普通多项式相乘基本相同，但要将乘积后的X^N变成1，X^{N+1}变成X，X^{N+2}变成X^2，...。即：

$a * b = (c_0, c_1, ..., c_{N-1})$,

$c_K = a_0b_K + a_1b_{K-1} + \cdots + a_Kb_0 + a_{K+1}b_{N-1} + a_{K+1}b_{N-2} + \cdots + a_{N-1}b_{K+1} = \sum_{i+j=k \bmod N} a_i b_j$

a*b的算法描述如下：

算法 1：

```
Step 1:  for (i=0;i<=N-1;i++)
          { k=i+1;c[i]=0;
Step 2:    for (j=0;j<=N-1;j++)
         { k--; c[i] = c[i] + a[k]*b[j];
Step 3:   if (k==0)  k=n+1;}   }
```

例 1：$a = 2 + X + 3X^2$, $b = 1 + 2X + X^2$

$a*b = 2 + X + 3X^2 + 4X + 2X^2 + 6 + 2X^2 + 1 + 3X = 9 + 8X + 7X^2$

(2)a mod p计算方法

多项式环的求模运算与普通多项式的求模运算完全相同，但在研究过程中发现，在算法2增加等价变换Step3、Step4步骤将会使产生NTRU密钥更容易，这种变换并不影响模运算法则。

算法 2：

```
Step 1:  p1= p/2;
Step 2: for (int i=0;i<=N-1;i++)
            {a[i]=a[i] mod p;
Step 3:  if (a[i]>p1)  a[i]=a[i]-p;
Step 4:  if (a[i]<-p1) a[i]=a[i]+p;}
```

例 2：$a = 12 - 14X - 8X^2 + 8X^3 - X^4$，p = 3

$a \bmod p = -2X - 2X^2 + 2X^3 - X^4 = X + X^2 - X^3 - X^4$

(3)b mod(X^n-1)计算方法

令$a = X^n - 1$,根据带余除法定理总存在q,r∈ R使

b=a*q+r, deg(b)>=deg(a), deg(a)>deg(r)或 r = 0,

由于a是一个特殊多项式，根据带余除法可以将r=b mod a运算变换成加法运算即：

$r_i = b_i + \sum_{k=i+N}^{\deg(b)} b_k$ ， k = i + N，i<N。

这样将给编程和计算带来方便。

b mod (X^N-1)的算法描述如下：

算法 3：

```
Step 1: for (i=0;i<N;i++)
         { k=i;k=i+N;
Step 2:   while (k<=deg(b))
         { b[i]=b[i]+b[k]; k=k+N;}
}
```

例 3：$b = X^4 + 2X^3 - 3X^2 + 4X + 1$, $a = X^2 - 1$

b mod a = 6X 1

(4)a(X)・b(X) ≡ 1 mod p 的计算

已知a(x),任意素数p，a(X)、b(X)∈ R_p若a(X)・b(X) ≡ 1 mod p时，则称b(X)是a(X)的乘逆。资料2中给出对任意素数p, 求解b(X)的方法。

算法4：

```
Step 1: 初始化 k=0;b[X]=1;c[X]=0;f[X]=a[X];g[X]=X-1
Step 2:  while ((f[0]= =0)&(deg(f)!=0))
{f[X]= f[X]/X;c[X]=c[X]*X;k=k+1}
Step 3: if (deg(f)= =0)
        {b[X]=f[0]*b[x] mod p;k1=(N-k)mod p;
        if (k1<0) k1=N+k1;
        b[X]=X^k1 b[X] mod(X^N-1);return b[X]}
Step 4:  if (deg(f)<deg(g))
{ 交换f[X]、g[X]; 交换b[X]、c[X],}
Step 5：u=f[0]*g[0] * mod p;
Step 6: f[X] = f[X] - u*g[X]( mod p), b[X] = b[X] - u*c[X]( mod p)
Step 7: goto Step 2
```

算法中的f[0]*、g[0]*分别是f[0]、g[0]对p的乘逆，文献[5]中给出了求整数乘逆的算法。在Step 2中只要将f,c的向量元素分别向左和右移1位就可以实现f(X)/X、c(X)*X运算。在Step 3 中增加了对N-k进行约化处理，避免出现X的指数为负数的情况，保证计算X^{K1}b[X] mod(X^N-1)的结果正确性，也简化了编程。对N-k 进行约化处理的可行性是因为f(X)、g(X)∈ R_p。在算法执行过程中deg(f)、deg(g)、deg(b)、deg(c)的值不断变化，因此进行Step 6 计算时，要使deg(f)=max (deg(f),g(g))、deg(b)=max(deg(b),deg(c)),扩展的向量值为0。

例 4：N=7, p=5, a(X)=(-2,1,-3,0,2,0,3),

按算法(4)可以计算出b(X)=(3,3,2,1,2,1,4)

验算：a(X)*b(X)=(6,-5,-5,5,0,15,0)=1 mod 5

为产生NTRU密钥，有时需要计算对模p^r， a(X)的乘逆b(X)。这时可以先根据算法4，计算 a(X)・b(X) ≡ 1 mod p中的b(X),再根据算法5计算a(X)・b(X) ≡ 1 mod p^r中的b(X)。

算法5：

```
Step 1: q=p
Step 2: while(q<p^r)
Step 3:{  q=q*p；b[X]=b[X](2-a[X]b[X]) mod q}
```

已知 p^r =pr ,根据pr值，可以按照算法6求出p、 r值。

算法 6：

```
Step 1:  p=1;q= √pr  ;p1=0
Step 2:  while ((p1!=pr)&(p<=q))
          { r=1;p++;p1=p
Step 3:  while (p1<pr){ p1= p1*p,r++ }
        }
Step 4:  if (p1>pr) {p=pr;r=1;}
```

由于NTRU技术推荐参数p值一般为3或2^r，而2、3是一个特殊的小素数，资料2中给出针对素数2、3, 求解乘逆b(X)的算法7、算法8。

算法7：

```
Step 1: 初始化 k=0,b[X]=1;c[X]=0,f[X]=a[X];g[X]=X-1
Step 2: while ((f[0]==0)&(deg(f)!=0))
        {[X]= f[X]/X;c[X]=c[X]*X;k=k+1}
Step 3: if (deg(f)= =0)
```

{K1=(N-K)mod p;

if (k1<0) k1=N+k1;

return $X^{K-1}b[X] \bmod (X^N-1)$}

Step 4: if (deg(f)<deg(g))

{ **交换**f[X]、g[X];**交换**b[X]、c[X],}

Step 5: f[X] = f[X] + g[X](mod 2); b[X] = b[X] + c[X](mod 2)

Step 6: goto Step 2

算法 8：

Step 1:**初始化** k=0;b[X]=1;c[X]=0;f[X]=a[X];g[X]=X-1

Step 2: while ((f[0]==0)&(deg(f)!=0))

{f[X]= f[X]/X,c[X]=c[X]*X,k=k+1}

Step 3: if (deg(f)= =0)

{K1=(N-K)mod p;

if (k1<0) k1=N+k1;0

return $\pm X^{K1}b[X] \bmod (X^N-1)$}

Step 4: if (deg(f)<deg(g))

{ **交换**f[X]、g[X];**交换**b[X]、c[X],}

Step 5：if (f[0]==g[0])

{f[X] = f[X] - g[X](mod 3);b[X] = b[X] - c[X](mod 3)}

Step 6: else

{f[X] = f[X] + g[X](mod 3);b[X] = b[X] + c[X] (mod 3)}

Step 7: goto Step 2

从算法7, **算法**8 **中我们看到，当模为**2**和**3**时执行算法**7, **算法**8**的速度要比使用算法**4**快得多。**

3 NTRU算法举例

例5：**取参数**$N = 11, p = 32, q = 3, d_f = 4, d_g = 3, d_r = 3$。**根据**$d_f$、$d_g$**值随机选择两个多项式**f、g：

f = (1，1，1，0，-1，0，1，0，0，1，-1)

g = (-1，0，1，1，0，1，0，0，-1，0，-1)

由(1)**式按照算法**8**求出**F_q= (1,0,1,0,1,0,0,1,0,0,1)

由(1)**式按顺序使用算法**6、7、5**求出**F_p

F_p = (6，12，8，30，32，2，10，31，12，22，5)

由(2)**式求出**h = (5,9,23,30,21,22,7,29,26,23,0)

设Bob**的公开密钥为**h，**一对私人密钥为**(f，F_q)。

Alice **要发送信息给**Bob**时，先根据**d_r**值随机选择一个多项式**r：

r=(1, 0, -1, 1, 0, 1, 0, -1, 0, 0, -1)

设Alice**要发送的明文多项式**m**为：**

m=(1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1)

Alice**根据**(3)**式使用**Bob**的公开密钥**h**对**m**进行加密得到密文**e：

e=(18,16,8,0,24,22,28,21,8,31,23)

Bob**收到**Alice**的密文**e **后，根据**(4),(5),(6)**式用自己的私人密钥**f，F_p**分别对**e **进行解密得：**

a=(6,-3,-4,5,-6,4,2,-15,4,7,-7)

b=(0，0，-1，-1，0，1，-1，0，1，1，-1)

c =(1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1)

c**就是**Alice**发送给**Bob**的明文**m。

4 结束语

按照公式(1),(2)**生成的密钥，有时并不能够正确实现加密和解密。这时可以通过编程方法自动修改**f**或**g**中的部分元素就可以了。**

本文给出了实现NTRU**公开密钥体制完整的方法，论文对文献**[2]**中对任意素数**p, **求解**a(X)**的乘逆**b(X)**的方法进行一些修改和补充说明，为进一步研究和应用** NTRU**算法提供参考依据。**

通过对NTRU**算法分析看到，**NTRU**算法设计非常巧妙，整个算法过程只包括小整数的加、乘、模运算，从而提高算法的执行速度。在国外对**NTRU**算法的研究、开发和应用进展异常迅速，**NTRU**算法被认为是公开密钥体制中最快的算法，也是比较容易实现的算法。同时国外有很多研究机构正在对**NTRU**算法安全性进行研究，但到目前为止还没有任何理由说明**NTRU**算法是不安全的，有理由相信**NTRU**算法完全有可能在公开密钥体制中占有主导地位。**

参考文献

1 Cohen H . A Course in Computational Algebraic Number Theory . Berlin:Graduate Texts in Math.,Vol.138 , Springer Verlag , 1993

2 NTRU Cryptosystems Technical Report #014. www.ntru.com

3 NTRU Cryptosystems Technical Report #009. www.ntru. com

4 Goldreich O , Goldwasser S , Halvei S . Public-key Cryptography from Lattice Reduction Problems.Proc. CRYPTO97, Lect.Notes in Computer Science 1294 , Springer-Verlag, 1997

5 **刘衍衍编著 . 计算机安全技术 . 长春：吉林科学技术出版社**,1997

☆☆

(**上接第**65**页**)

匹配，得到监控结果。如图3**为监控系统的结构图。**

图3**中，采集器用来收集网络通信信息，并向分析单元提供分析所需的数据，同时还能接收分析单元的指令，进一步采集分析单元所需的特定信息。分析单元采用人工神经元网络对网络采集信息和模式库进行模式匹配得出分析报告，同时定期进行自学习，更新模式库，及时跟踪并反映拒绝服务攻击模式的最新动态。关于本监控系统的设计与实现，我们在另文论述。**

参考文献

1 **陈 波**, **于 泠**.DoS**攻击原理与对策的进一步研究**. **计算机工程与应 用**,2001,37 (10)

2 **陈 波**,**于 泠**.SYN flooding**攻击对策研究计算机工程**,2001,27(7)

3 Lau F,Rubin S H,Smith M H,et al.Distributed Denial of Service Attacks.http://www.ensc.sfu.ca/people/faculty/ljilja/papers/smcoo edited.pdf,2000

4 Bellovin S.Security Problems in the TCP/IP Protocol Suite.Comput. Commun.Rev.1989,19(2):32-48

5 Bellovin S.Distributed Denial of Service Attacks.http://www. research. att.com/~smb/talks,2000-02

6 Dttrich D.The DoS Projects Trinoo Distributed Denial of Service Attack Tool,the Stacheldraht Distributed Denial of Service Attack Tool, the Tribe Flood Network Distributed Denial of Service Attack Tool. http:// www.Washington.edu/People/dad.,1999-10

7 Gregory P H.Solaris Security.Prentice Hall PTR,2000