An Efficient Method of Shared Key Generation Based on Truncated Polynomials

P.Prapoorna Roja,[†] P.S.Avadhani[††].and E.V.Prasad[†††],

[†]G.V.P. College of Engineering, Visakhapatnam, A.U. College of Engineering Visakhapatnam, J.N.T.U College of Engineering, Kakinada

Abstract

Many of the public key cryptosystems deal with two-party communication keeping confidentiality and authentication as primary goals. However there are many applications like banking that require multi-party communication. Though systems like RSA provided multi-party communication using shared key approach, overheads of RSA seem to be more because it has to choose n pairs of numbers such that the summation of these numbers is a large prime number. This needs to be done without revealing the shares of the numbers [1, 2]. The public key cryptosystem NTRU, based on polynomials, can be used for multi-party communication. This cryptosystem does not require these overheads. This paper proposes an algorithm for shared key authentication based on NTRU for multi-party communication, by giving algorithms for encryption, decryption and key generation.

1.0 Introduction

Data security and secure communication have become necessary at the advent of recent developments in Information Technology. Many cryptographic algorithms proposed in literature [3] provide safe and secure transit of documents over the widely used Internet. The basic goal of any cryptosystem is to allow two or more people to communicate in such a way that the content of their message will remain confidential and authenticated, even if there is a possibility of the messages being hacked.

The Public Key Crypto System (PKCS) provides a methodology for transmitting documents and allows two or more people to communicate while maintaining confidentiality and assuring authentication. The RSA cryptosystem [4] depends on the difficulty of factoring large numbers. The strength of the RSA cryptosystem relies heavily on the large prime numbers and the difficulty of factoring their product. Another public key cryptosystem called NTRU [5] stems out from the truncated polynomials and modulo operations on these polynomials. This NTRU cryptosystem reduces the amount of overheads unlike RSA.

All these cryptographic algorithms including NTRU base their algorithms upon a two-party communication. However, many applications require more than two parties to communicate among themselves simultaneously. This multi-party communication necessitated the need for the present day demand of applications that need a security enforcement technique requiring a multi-party communication.

The current area of research including this paper is focusing on the needy situation of multi-party communication, which has many applications like financial transactions. These applications require more than a single recipient to receive the message. In such cases, the private key needs to be shared among different receivers of the message in such a way that until both the receivers insert their keys, the message cannot be decrypted. These concepts of shared key generation and the algorithms for implementation are addressed in this paper. This paper uses the concepts of NTRU public key cryptosystem for their implementation of shared key approach.

2.0 Mathematical Preliminaries

2.1 Modular Arithmetic

If m is a positive integer, and a, b are two integers then, a is said to be congruent to b modulo m, if a – b is divisible

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.

by m denoted by $a \equiv b$ modulo m or simply $a \equiv b \pmod m$. The properties of modular arithmetic are

(i) If $a \equiv b \pmod m$ and $c \equiv d \pmod m$ then $ax + cy = bx + dy \pmod m$ for all integers x and y.

(ii) If $a \equiv b \pmod m$ then $a^n \equiv b^n \pmod m$ for any positive integer n

(iii) If $a \equiv b \pmod m$, then $F(a) \equiv F(b) \pmod m$, for any polynomial F (x) with integer coefficients.

2.2 Truncated Polynomials

NTRU public key cryptosystems lay their foundations on polynomials R (x) of degree n-1 having integer coefficients of polynomials in R (x)

Addition is done in the usual way, while multiplication is done using truncated polynomials

If $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1}x^{n-1}$ and

$g(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{n-1}x^{n-1}$

$h(x) = f(x) * g(x) = c_0 + c_1x + c_2 x^2 + c_3x^3 + \ldots + c_{n-1} x^{n-1}$

Where $c_i = a_0 b_i + a_1 b_{n-i} + \ldots + a_i b_o + a_{i+1}b_{n-1} + \ldots + a_{n-1} b_{i+1}$

2.3 Modulo Operations on Truncated Polynomials

We say that f (x) is congruent to k modulo p, if p divides every coefficient of f (x) except the constant term f (0) and p divides f (0) –k. This is denoted by $f(x) \equiv k \pmod p$.

2.4 Inverse in Truncated Polynomials

The inverse mod p of a polynomial f (x) is another polynomial F (x), if it satisfies the property $f * F = 1 \pmod p$. It is observed, that not every polynomial has an inverse modulo p but is easy to determine if f has an inverse and compute the inverse if it exists.

3.0 Outline of the NTRU Algorithm

This section gives an idea of the working of the NTRU algorithm given by Silverman and Hoffstien [5].

The NTRU PKCS uses a ring that consists of truncated polynomials of degree n-1 denoted by $Z[x] / x^n-1$.

Select a large modulus q and a small modulus p, so that gcd (p,q) =1 . The coefficients of the truncated polynomial in $Z[x] / (x^n-1)$ will be reduced (mod q). In the final step of decryption, the coefficients of polynomial are reduced modulo p.

First select two small polynomials f and g (A small polynomial is one in which all the coefficients are either 0, -1, or 1). Select $d_f < n-1/2$ and make d_f coefficients of f to 1 and d_f -1 coefficients of f to –1 and the rest to 0. Similarly select d_g such that $d_g < n-1/2$ and make d_g coefficients of g to 1 and d_g coefficients of g to –1 and the rest to 0. f and g must be private.

3.1 Key Creation

1. Compute inverse of f mod q = f_q ,and inverse of f mod p = f_p with the property that $f. f_p = 1 \pmod p$ and $f. f_q = 1 \pmod q$.
2. Compute the public key $h = p*.f_q * g \pmod q$

Public key polynomial = h

Private key polynomial = {f, f_p }

3.2 Encryption

1. Select a message m and put it in the form of a polynomial m with the coefficient between –p/2 and p/2.
2. Pick a random small polynomial r with coefficient +1, -1, or 0 such that r(1) = 0
3. Encrypt the message m as $e = r*h + m \pmod q$

3.3 Decryption

1. Upon receiving the ciphered text e, compute $a = f*e \pmod q$
2. Express the coefficients of a in the range -q/2 to q/2
3. Compute $b = a *f_p \pmod p$
4. Original message m= b

4.0 The Proposed Efficient Method of Shared Key Generation Using NTRU

This paper proposes a shared key generation basing on NTRU, such that everybody knows the public key h, and the private key {f, f_p} is shared between the partners. This paper assumes there is a trusted third party, and given shares of the polynomials it will compute the inverse and distribute shares of the inverse to the participants. One more requirement is that previous to the communication, the shared recipients A and B have already generated public key, private key pair and a modulus M each, based on any commonly agreed upon PKCS. Let these values be pub_1, pr_1, and m_1 for participant A and pub_2, pr_2 and m_2 of participant B. Let X be a party wishing to send a document to A and B. Further A and B also need to agree upon a common degree N and the values of p and q.

4.1 Key Creation

This section deals with the algorithm for key creation for shared key using polynomials.

(See figure 1)

1. Let A generate two polynomials f_1 and g_1 such that
   $f_1(x) = a_0 + a_1 x + a_2 x^2 + a_3x^3 + \ldots + a_{n-1} x^{n-1}$
   $g_1(x) = c_0 + c_1x + c_2 x^2 + c_3x^3 + \ldots + c_{n-1} x^{n-1}$
2. Similarly let B generate two polynomials f_2 and g_2 such that
   $f_2(x) = b_0 + b_1x + b_2 x^2 + b_3x^3 + \ldots + b_{n-1} x^{n-1}$
   $g_2(x) = d_0 + d_1 x + d_2 x^2 + d_3x^3 + \ldots + d_{n-1} x^{n-1}$
3. Let both A and B send their f_1, f_2 to the Trusted Third Party (TTP) by encrypting them with the public key of the TTP.
4. The TTP will calculate $f = f_1 * f_2$ and find the inverse of f if it exists else the same is informed to A and B, in which case they have to choose fresh values of f_1 and f_2. The process is repeated until an inverse of f exists. Let the inverse of f be f_p.
5. The TTP will factorize f_p into two parts f_{p1} and f_{p2}. The TTP will also calculate f_q and then send the shares of f_p (f_{p1} and f_{p2}) and f_q to A and B by encrypting them with the public keys of the A and B respectively.
6. The polynomial g in NTRU algorithm needs to be constructed by using g_1 and g_2. A has to formulate the small polynomial g_1 and B needs to formulate g_2. For exchanging the values of g_1 and g_2, A will encrypt the coefficients and degree of g_1 with the public key of B, and B will encrypt the coefficients g_2 and degree of g_2 with the public key of A. Then A and B will need to decrypt the messages received and compute $g = g_1 * g_2$.
7. Let each party now calculate the public key $h = p * f_q * g \bmod q$.
8. The private key of A is (f_{p1}, f_1) and that of B is (f_{p2}, f_2)

4.2 ENCRYPTION

This section deals with the encryption process involved. The process of encryption is explained in figure 2 .When X wants to send a document to be viewed by both A and B , X will initially have to express the message m as a polynomial and choose a random polynomial r where r is small similar to f and g. Then X needs to encrypt the message m using the formulae

$$e = r * h + m$$

4.3 DECRYPTION

The method used to decrypt a given ciphered document by the method of sharing the private key is dealt in this section. Figure 3 gives a diagrammatic explanation of the process.

The ciphered text e is sent to both A and B.

1. Let each participant (i.e A and B) choose random small polynomials r_1 and r_2.
1. Let A calculate $z_{a1} = f_1*(e + r_1)$, let B calculate $z_{b1} = f_2* (e + r_2)$
2. The values of z_{a1} and z_{b1} are to be exchanged using the previously agreed upon public keys pub_1 and pub_2. After decryption z_{a1} will be known by B and z_{b1} by A
3. Now A will calculate $z_{a2} = f_1 * z_{b1} \bmod q$ and B will calculate $z_{b2} = f_2 * z_{a1} \bmod q$. A will then need to adjust the coefficients of z_{a2} in the range of [q/2, –q/2]. B will also need to repeat the same with z_{b2}. A will then calculate $z_{a3} = z_{a2} \bmod p$ and B will calculate $z_{b3} = z_{a2} \bmod p$.
4. A will calculate $z_{a4} = z_{a3} * f_{p1}$ and B will calculate $z_{b4} = z_{b3}* f_{p2}$. Then these values of z_{a4} and z_{b4} will need to be exchanged between A and B
5. A will retrieve the message m by computing $m = f_{p1} *z_{b4} – r_1$ and B will retrieve the message m by computing $f_{p2} *z_{a4} – r_2$.

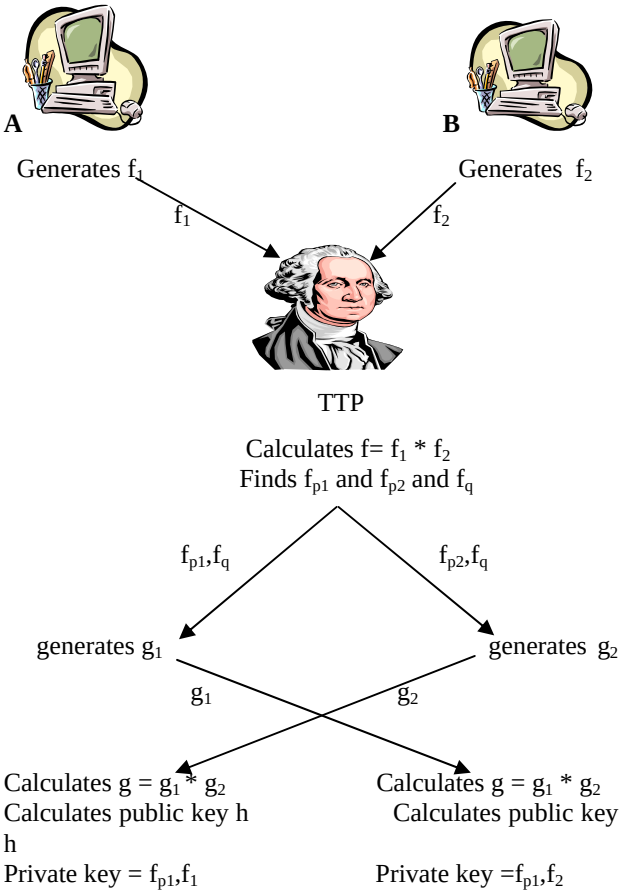
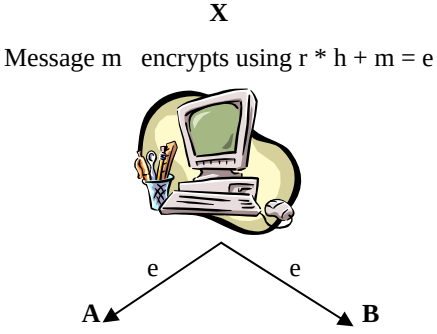


Fig. 1 Key Creation: All communication between parties to take place in encrypted form



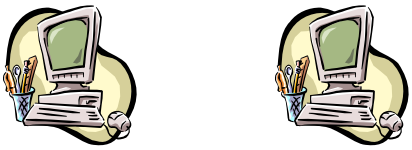


Fig 2. Encryption

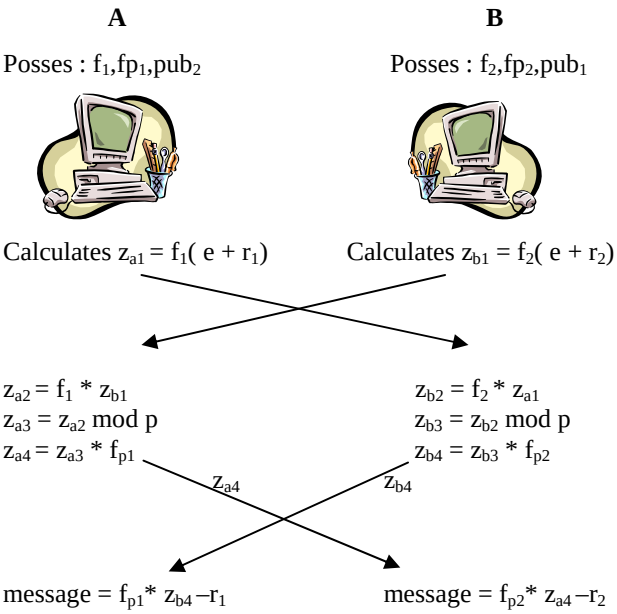


Fig 3. Decryption**:** All communication between parties to take place in encrypted form

5.0 Analysis of The Shared Key Using NTRU Algorithm

The strength of the NTRU algorithm lies in keeping f and g secret [5]. Though f is to be maintained secret it is f_p that is commonly used hence this paper has concentrated in keeping both f , f_p confidential by dividing f into shares and each party involved in the communication have to keep their shares of f_i secret . The f value is calculated by the TTP after receiving the shares of f_i from each participant as Πf_i for i=1,2,... n. The TTP after calculating f and then its f_p w.r.t p, will factorize it into n parts and distribute it to all the n parties. $f_p = \Pi f_{pi}$ it will not be possible for any individual party to decrypt all by themselves as they have to obtain the shares of f_p from the remaining parties. While obtaining the values, the algorithm suggested in this paper has taken care that, the

value of the share of f_p of any party is not revealed to the others.

During decoding initially, each user will need to compute

$z_{a1} = f_1(e + r_1)$
$= f_1(r*h + m + r_1)$
$= f_1(r(p*f_q* g) + m + r_1)$
$= f_1(r* p *f_q *g + m + r_1)$

Similarly $z_{b1} = f_2(r * p * f_q * g + m + r_2)$

These values, are exchanged by the two parties

Hence A knows both z_{a1} and z_{b1}

When A calculates $z_{a2} = f_1* z_{b1} \bmod q$
$= f_1 * f_2(r * p * f_q * g + m + r_2) \bmod q$
$= f * (r * p * f_q * g + m + r_2) \bmod q$
$= p * r * g + f (m + r_2) \bmod q$

As the values of r, g, m and r_2 are small compared to q [5]

$z_{a2} = p * r * g + f(m + r_2)$

$z_{a3}= z_{a2} \bmod p$

$z_{a4} = z_{a3} * f_{p1}$
$= z_{a2} * f_{p1} \bmod p$
$= f_{p1} (p * r * g + f(m + r_2) \bmod p$
$= f * f_{p1} (m + r_2) \bmod p$

Similarly $z_{b4} = f * f_{p2} (m + r_1) \bmod p$

These values of z_{a4} and z_{b4} are exchanged. Hence A knows z_{b4}.

Message m is retrieved by the operation $f_{p1}* z_{b4} – r_1$
$= f_{p1} * f * f_{p2} (m + r_1) \bmod p – r_1$
$= m + r_1 – r_1$
$= m.$

6.0 Conclusions

This paper proposes algorithm for shared key generation, encryption, and decryption based on NTRU. The paper assumed that one party is communicating with two parties. However, this can be extended to n parties.

References

[1] C.Cocks, “Split Knowledge Generation Of RSA Parameters, Cryptography and Coding” 6 Th IMA Conference, Lecture Notes In Computer Science Style, Vol 1423, pp 237-251,Springer Verlag, New York, 1997.

[2] Boneh.D., Franklin M., “ Efficient Generation of shared RSA keys “ in proceedings of Crypto - 97, 1997, pp 425 – 439

[3] Bruce Schneier“ Applied Cryptography”, Wiely And Sons 1994, ISBN 0 – 471 – 597562-2.

[4] Rivest R.L., Shamir . A and Adleman, L. “ A Method for obtaining Digital Signature and Public Key Cryptosystem”, Comm. ACM , Vol. 21, No. 2 , 1978, pp 120-126.

[5] J. Hoffstein, D. Lieman, J. Silverman“ Polynomial Rings and Efficient Public Key Authentication”, Proceeding of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99), M. Blum and C.H. Lee, eds., City University of Hong Kong Press,1999.

[6] R.Padmavathy, P.Prapoorna Roja“ Enterprise Information Security” proceedings of the National Conference CSI-BIG 2003” seminar held at Visakhapatnam from jun 7th – 10th ,2003.

[7] P.S.Avadhani,, N.Chalamaih, P.Prapoorna Roja “ Secure Transit Of Confidential Documents Over Internet Using High Speed RSA” proceedings volume IV ,pg no 74-78, International Conference CCCT’04 held at Austin from August 14-17 ,2004.

[8] P.Prapoorna Roja, P.S.Avadhnai “ Shared key Authentication using NTRU”, proceedings of International conference on E-Security, held at Visakhapatnam, Feb 24th –26th,2006.

Mrs.P.Praapoorna Roja received her B.Tech degree from Andhra University college of Engineering in 1992. She received her M.Tech degree in Control Systems in 1995 and M.Tech in Computer Science & Technology in 2001from Andhra University. She has experience of 11 years in teaching and is presently working as a professor in Department. of Information Technology in G.V.P.College of Engineering. Visakhapatnam, India.

Prof. P.S.Avadhani did his Masters Degree and Ph.d from IIT kanpur. He is presently working as a Professor in Department on CSSE and is heading the same in Andhra university college of Engineering in Visakhapatnam. He has 32 papers published in various national/ inetrnatioanl journals and conferences. His research areas include Cryptography, Data Security, Algorithms, and Computer Graphics. He is member for Board of studies for A.U

Prof. E.V.Prasad received his Ph.d from IIT Roorke. He is presently the Vice-Principal and heading the Department of CSE, J.N.T.U College of Engineering, Kakinada . He has 26 papers published in various national/ international journals and conferences. His research areas include Network security, Data Mining, Computer Architecture, Parallel Algorithms. He was a review committee member for IEEE transactions on Reliability. He is the Chairman for Board of studies for J..N.T.U College of Engineering. He is also a member of various prestigious comities like AICTE,NBA.