CHAPTER – IV

NTRU Key Distribution and Implementation – Peer to Peer (Group)
CHAPTER 4

NTRU KEY DISTRIBUTION AND IMPLEMENTATION - PEER TO PEER/GROUP

NTRU is an abbreviation for Nth degree truncated polynomial ring. The main characteristic is that during the encryption and decryption the polynomial multiplication is the most complex operation. The mathematicians were considered on speeding up the process.

This Chapter introduces significance of Fair Exchange protocols and presents the proposed protocols for invoking communication among different parties such as Customer, Trader, Multi Trader and Banker using our developed cryptosystem using NTRU scheme. In section 4.1 deals with Importance of Fair Exchange protocol using NTRU cryptosystem. Section 4.2 introduces Registration, Key-Generation phase which include generation of Public keys, Private Key for Customer\Users and Private Keys for Trader, Bank etc., using proposed protocol NTRU crypto scheme. Section 4.3 presents symmetric key generated used to share the public key in section 4.2 Section. Section 4.4 provides the communication between mobile users’ without involvement of Sever. Section 4.5 provides the basic structure of certification generated by the NTRU.

4.1. FAIR-EXCHANGE

Fair-exchange is an important property that must be ensured in all electronic\ mobile commerce environments where the merchants and the customers are reluctant to trust each other. This property guarantees that none of the transacting parties suffer because of the fraudulent behavior of the other party in the transaction. In this chapter, we proposed developing for registration protocol phase of Customer, Trader and Bank address the problem of fair-exchange for digital products. We show how to use NTRU[39] software components to develop such a protocol. Applications such as e-commerce\m-commerce payment protocols, electronic contract signing, and certified e-mail\SMS (Short Message Service) delivery require that fair exchange be assured. A fair-exchange protocol allows two parties to exchange items in a fair way so that
either each party gets the other's item, or neither party does. We describe a novel method of constructing very efficient fair-exchange protocols by distributing the computation of NTRU. Today, the vast majority of fair-exchange protocols require the use of zero-knowledge proofs, which is the most computationally intensive part of the exchange protocol. Using the intrinsic features of our dual-signature model, we construct protocols that require no zero-knowledge proofs in the exchange protocol.

An exchange or payment protocol is considered fair if neither of the two parties exchanging items or payment at any time during the protocol has a significant advantage over the other entity. Fairness is an important property for electronic commerce. In our protocol is detailed based on NTRU undeniable signatures which are more efficient than other similar fair payment schemes.

There are some protocols that do not ensure fairness but provide a weaker form of protection: the gathering of evidence during execution so that if one party obtains the other's item without sending his, the dishonest party can be prosecuted using the evidence. These protocols are efficient, but, as mentioned above, do not guarantee fairness. In gradual exchange protocols the probability of fair exchange is gradually increased over several rounds of message exchanges. These protocols are impractical because extensive amounts of communication are needed. Furthermore, proofs of their security rely on both parties having equal computational power, which is unrealistic for most applications.

In fair-exchange protocols with an on-line TTP, a TTP is directly involved in every exchange, and must be available for the entire duration of the exchange. The protocol itself is relatively simple md5 computationally efficient. However, maintaining a TTP that needs to be on-line constantly can be expensive. Moreover, the TTP can become a bottleneck, and pose scalability problems. In this type of a protocol we use the NTRU effectively without having the TTP involved directly instead of TTP is involved in the protocol only if one of the parties behaves unfairly or aborts the protocol prematurely; otherwise the TTP is never involved in the protocol. To clarify fairness primitive is used in an exchange protocol, we present an example of a basic optimistic fair-exchange protocol. The protocol framework is essentially. Let Alice (Trader-Receiver) and Bob (Customer-Sender) be two players trying to exchange their
respective digital signatures is and as on a message M (known a priori to both parties) using their registered NTRU Server or TTP. In the first step of the protocol, one of the items that Trader (Peer) / ML-Trader (Group)/Bank(Group) sends to is her "commitment" to the transaction, which we denote as CA.

4.2 REGISTRATION PHASE

The registration protocol needs basic information of different parties such customer, Trader and Bank. The registration phase to be performed only once, after which it can support any number of exchanges. We assume that the registration protocol is performed via confidential and authenticated channels which can be generated on different parties are submitting their details to ZTTP and the ZTTP will verifies and generates the Symmetric key in our proposed protocol we are using Diffie Hellman key Exchange algorithm using this symmetric key both parties are exchanging the information using private key algorithm (RC4). The key is used to encrypt the message to deter eavesdropping.

4.2.1 Key Generation Phase (KGP)

In Key Generation Phase our proposed protocol using NTRU algorithm, will start once, the symmetric key distribution is over. In our proposed protocol, we are generating the public key, private key for users, Traders etc., to gain the authentication. For user should create public key as below:

Select a secret key $f$ (a random secret polynomial $f \in R$ with coefficients reduced to modulo $p$ ), select a random polynomial $g \in R$ with coefficients reduced to modulo $p$ and compute the inverse polynomial $f_q$ of the secret key $f$ in modulo $q$ after completing above process the public key $h$ is compute as below:

$$h = p f_q * g \ (mod \ q)$$

4.2.2. Encryption Phase: Encrypting the given message $m$ is given as follows

$$e \equiv pr * h + m \ (mod \ q)$$

Whereas the coefficients of the message $m \in R$ and the random polynomial $r \in R$, are reduced to modulop.
4.2.3. Decryption Phase: The Decryption process requires the following procedure.

\[ a = f \cdot e \pmod{q} \]
\[ a = f \cdot (r \cdot h + m) \pmod{q} \]
\[ a = f \cdot (r \cdot p \cdot f_q \cdot g + m) \pmod{q} \]
\[ a = pr \cdot g + f \cdot m \pmod{q} \]
\[ b = pr \cdot g + f \cdot m \pmod{p} \]
\[ b = f \cdot m \pmod{p} \]
\[ c = f_p \cdot b \pmod{p} \]
\[ c = m \pmod{p} \]

The final decryption step requires the user to compute the inverse polynomial \( F_p \) of the secret key \( f \) in modulo \( p \), the decryption process outlined above subsequently recovers the original message.

4.2.4. Keys Exchange session using Diffie-Hellman

We proposed to implement public key cryptography on the mobile phone by using NTRU algorithm. Which is a NTRU public key cryptography, mobile phones will be able to achieve all cryptographic operations such as key generation, encryption /decryption and signing /verifying without relying on the third party’s server. The users will also gain the confidentiality, authentication, integrity and non-repudiation security services for their mobile phone communication. However, the problem of how the communicating parties authentic each other appears. Although, the problem has been faced in first contact only; meaning that when they did not have each other’s public keys. To solve this problem, we proposed to use key exchange session to exchange the public key between them. The key exchange session is used in the proposed system. Diffie-Hellman[34] algorithm is used to make agreement on a temporary key that is used with encrypt the public key and exchange it with the partners.
Figure 4.0 illustrates the key exchange session steps.

From the above figure 4.0 Mobile User X (Peer) can start the key exchange sessions immediately after he/she generates his/her public keys and add Mobile User Y(Peer) contact information to his/her contacts list. He/she can start the key exchange session by calculating the value of $A$, depending on the secretly generated value $a$, and the shared secret parameters $g$ and $p$ (that is, $g$ and $p$ are fixed by the mobile application).

$$A = g^a \mod p$$

He/she then sends the value of $A$, with request to start key exchange session. Mobile User Y can reject the session if he/she is not ready to go through the key exchange session steps. He/she can also accept the request; if so, he/she must calculate the value of $B$ depending on the secretly generated value $b$ and the shared secret parameters $g$ and $p$ and send it back to Mobile User X with accept message.

$$B = g^b \mod p$$
Mobile User X will be able to calculate the value of K once he/she receives the value of B from Mobile User Y. Mobile User Y also will be able to calculate same key K, depending on the value of A, which has already been received from Mobile User X in the request message. Thus, Mobile Users X and Y will obtain the same secret key and then, they can use it for one time only to encrypt their public key and exchange it. For next key exchange session, users can use NTRU public keys to encrypt and sign the new cryptographic keys before exchanging them.

To analysis this case, we will focus on the keys exchange between two mobile users, mobile user X and mobile user Y, and the potential risks of intercepting the messages by the attacker mobile user Z. The first message is the request message for the keys exchange. The first message sent from Mobile user X to mobile user Y holds the value of A. Assuming that the attacker mobile user Z manages to capture this message, he/she will be unable to obtain the value of key K by only depending on the value of Mobile User A. The second message is the reply message which is sent from Mobile User Y to Mobile User X. this message is to accept the exchange of keys and it contains the value of B. Assuming that the attacker Mobile User Z manages to capture this message, he/she will not be able to obtain the value of key K because, the value a is kept secret; this value is only known by user Mobile User X, as well as the value of b which is kept secret by user Mobile User - Y. In addition, the lack of knowledge of Diffie-Hellman algorithm parameters g and p will make the calculation of the key K value impossible. The third and fourth messages are the encrypted messages using the NTRU keys algorithm, which hold the users’ public keys. Even if the attacker could capture the messages, he/she will fail to decrypt them and will not know the users’ NTRU public keys because of lack of knowledge of the value of key K. Moreover, the attacker Z will fail to start a key exchange session because of lack of knowledge of the Diffie-Hellman parameters and port number; also the solution will reject his request because his contact number is not in the contacts list. Therefore, the user will be confident after the completion of the keys exchange session that the process has been made with the right person. In addition, even if the attacker Z successfully impersonates one of the parties, he/she will fail to complete a successful keys exchange with the other user due to lack of knowledge of the Diffie-Hellman parameters that are needed to complete the process of making agreement on a shred secret key with the other user.
The key exchange stage is only a temporary stage needed only in the first contact between the users. Once the key exchange has been successfully accomplished, the next stage will start, which is, exchanging encrypted messages. This stage is permanent and fixed. At this stage, users will be able to send and receive the encrypted and signed messages. They will also be able to exchange new updates for the current keys in encrypted and signed messages. As a result, they will be able to verify the identity of the sender of any message and they can ignore any spurious message. Since the attacker fails to benefit from any of the captured messages during the keys exchange session, in the process of violation of the privacy of any party to the communication, he will not be able to decrypt the captured encrypted messages later. Thus, we can say that the proposed non-server security scheme for mobile communications is capable of providing a high level of security for users. It guarantees provision of the confidentiality, integrity, authentication and nonrepudiation security services.

4.3 DESIGN AND DEVELOPMENT OF PUBLICATION OF PUBLIC KEY OF GROUP NTRU USING DH KEY.

Using the above the protocol users can generate the public and private keys using NTRU (Section 4.2.4) after they can exchange the Public key using the DH key exchange algorithm which as be explained in the above Figure 4.0. The protocol as follows below steps.

4.3.1. User’s Public Key Generation Protocol.

Using the below protocol Mobile users can generate their public and private keys using NTRU (Section 4.2.4) from the TTSM after they can exchange the Public key using the DH key exchange algorithm which as be explained in the below protocol steps. (code is in appendix-I,II,III)

1. Mobile User Request for Connection establishment to TTMS (TTMS-Trust Third Messaging Service) Server
2. TTMS Server Creates the Session for Mobile user for future Communication
3. TTMS Server responds to Mobile User by establishing the Connection
4. Mobile User Request for Symmetric key and sends his information
5. TTMS Generates the Prime number set = Pi
Figure 4.1 Peer’s Public Key Generation Protocol.

- **Requesting for Connection**
- **Response from TTMS**
- **Requesting for Symmetric key & sends user information**
- **Generate Prime key Set**
- **Generate Prime Number P, and Primitive Number Y**
- **Generate Random number g**
- **Calculate G = Y^g mod P**
- **Response to client and sends the prime and primitive number**
- **Calculate H = Y^h mod P**
- **Generate Random Number h**
- **Generates Random Number h**
- **Calculates H = Y^h mod P**
- **Stores the P, and Y values**
- **Generates Random Number h**
- **Generates Sykey = G^h mod P**
- **Request for Key & sends the H value to Server**
- **Sends the Ack to Server & Requesting for Keys**
- **Decryptes Key by using Sykye using RC4**
- **Stores keys**
- **Sends the Ack to Server Received the Keys**
- **Publishes the Public key in Directory**
- **Close the client Session**
- **Encryptes Key by using Sykye using RC4**
- **Response Exchange along with keys**
- **Generates Sykey using prime Number TTMS generates public and private keys**
- **Using prime Number TTMS generates public and private keys**
- **Encrypted by SyKye using public**
- **Response Exchange along with keys**
6. TTMS Select the Largest Prime(Pi) and finds its Primitive root(Y) and sends to Mobile User
7. Mobile User Stores the Pi and Y
8. Mobile User generates the random number h, calculates the H=Yh mod Pi and sends H to TTMS
9. TTMS generates the random number g and calculates the G=Yg mod Pi
10. Sends G value to the Mobile User
11. Mobile User Generates the SyKey = Hg mod Pi
12. Mobile User sends ack and requests for public and private keys
13. TTMS Server Generates the SyKey = Gh mod Pi
14. Server generates the public and private keys using NTRU
15. Server encrypts the public and private keys using RC4 Symmetric algorithm and key as symmetric key (SyKey)
16. TTMS server sends encrypted message to Mobile User
17. Mobile User decrypts the public and private keys using RC4 Symmetric algorithm and key as Symmetric key (SyKey) and Stores it.
18. TTMS server publishes the Mobile User’s Public Key
19. Server closes the Session of Mobile User’s.

Peer to Peer Communication Server TTMS Screen

![Peer Server Protocol Sample Screen](image)

Figure 4.2. Peer Server Protocol Sample Screen
4.3.2. Trader’s Public Key Generation Protocol.

Using the below protocol Traders can generate their public and private keys using NTRU from the TTSM after they can exchange the Public key using the DH key exchange algorithm which as be explained in the below protocol steps.

1. Trader’s Request for Connection establishment to TTMS (TTMS-Trust Third Messaging Service) Server
2. TTMS Server Creates the Session for Trader’s for future Communication
3. TTMS Server responds to Trader’s by establishing the Connection
4. Trader’s Request for Symmetric key and sends his information
5. TTMS Generates the Prime number set = Pi
6. TTMS Select the Largest Prime(Pi) and finds its Primitive root(Y) and sends to Trader’s.
7. Trader’s Stores the Pi and Y
8. Trader’s generates the random number h, calculates the H=Yh mod Pi and sends H to TTMS
9. TTMS generates the random number g and calculates the G= Yg mod Pi
10. Sends G value to the Trader’s
11. Trader’s Generates the SyKey = Hg mod Pi
Figure 4.4. Peer’s Trader Public Key Generation Protocol

12. Trader’s sends ack and requests for public and private keys
13. TTMS Server Generates the SyKey = Gh mod Pi
14. Server generates the public and private keys using NTRU
15. Server encrypts the public and private keys using RC4 Symmetric algorithm and key as symmetric key (SyKey)
16. TTMS server sends encrypted message to Trader’s
17. Trader’s decrypts the public and private keys using RC4 Symmetric algorithm and key as Symmetric key (SyKey) and Stores it.
18. TTMS server publishes the Trader’s Public Key
19. Server closes the Session of Trader’s.

Peer to Peer Communication Server TTMS Screen

Figure 4.5. Peer Trader - TTMS Protocol Sample Screen

Peer to Peer Communication Trader Screen
4.3.3. Banker’s (Group) Public Key Generation Protocol.

Suppose that Banker’s wishes to sign a message $m$ with a ring signature for the ring $[41,]$, of $n$ individuals $B_1, B_2, \ldots, B_n$, where Banks is $BS$ for some $s$, $1 \leq s \leq n$. Given the message $m$ to be signed, $BS$’s private key $S_s=(f, g, f^{p-1}, f^{q-1}, g^{p-1}, g^{q-1})$, and the sequence of NTRU public keys $P_1, P_2, \ldots, P_n$ of all the ring members, $BS$ computes a ring signature as follows:
Figure 4.7. Peer’s Public Key Generation Protocol.

Group Communication Server TTMS Screen
4.4. COMMUNICATION BETWEEN MOBILE USERS’ WITHOUT INVOLVEMENT OF SEVER (TTMS)

Mobile users initiates the protocol with TTMS while registration phase later onwards using the TTMS keys mobile users can share\exchange the message. We assume that mobile user’s gone through a negotiation process to agree on the symmetric key generation, which might contain key generation phase unique identity prior to the start of the DHKEA protocol. The key is used to encrypt the message to deter eavesdropping.

4.4.1 Communication between user’s and Trader
User initiates the protocol with Trader. We assume that User and Trader have gone through TTMS a negotiation process to agree on the Premium information M (which might contain User’s unique identity, Trader's unique identity) prior to the start of the transaction protocol. This process may be as simple as User choosing a transaction to the Traders website/App. Note that User’s digital signature on M acts as digital check. In addition, User and Trader/Bank/TTMS agree on using the key-agreement protocol i.e NTRU key agreement.

4.4.2 Communication between Bank (Group) and User/Trade (peer)

Bank initiates the protocol with User/Trader with help TTMS. We assume that Bank and User/Trader have gone through a negotiation process to agree on the transaction M (which might contain Bank's unique identity, User/Trader unique identity, account number, type of transaction, amount of transaction and date of transaction) prior to the start of the transaction protocol. This process may be simple for Bank transactions from user/Trader website/App. Note that Bank's digital signature on M (which is Dual-signature) acts as digital check. In addition, Bank and User/Trader agree on using the key-agreement protocol i.e NTRU key agreement.

4.5. DIGITAL CERTIFICATES PHASE

Digital certificates or public key certificates, serve as the equivalent of a passport for electronic identity. Digital certificates are electronic forms of identification that can be validated by a recognized authority. All users of PKI must have this form of registered identification.

4.5.1. Our TX.509 Version 3 Certificate

A certification authority (CA), an e-medical server, purpose is to issue digital certificates[26] and to confirm the identity of the person associated with a certificate. CA’s brings an added level of trust to PKI-based transactions. The role of the CA can be better understood by using an analogy to a real world passport office. A passport office issues a passport, a secure document that certifies the person holding the passport is who he or she claims to be. Any country that trusts the authority of the passport holder’s passport office will also trust that individual’s passport. The
passport office is a third party that vouches for the identity of the person in question. The function of a CA is the same as that of a passport office – a trusted third party (EMS) authority.

A digital certificate is a data structure digitally signed by a trusted entity, called Certification Authority (CA). It contains the public key that is being certified. The purpose of its use (e.g. encryption, signing, key exchange etc.), the subject (the owner of the public key), the issuer (the CA), date of issuing, validity period of the certificate, an identifier denoting the signature algorithm used to sign the certificate, and optional extension fields that can be used to customize certificates. The fields contained in a standard certificate are defined by the X.509 Certificate Specification and an X.509 v3 (version 3) certificate is presented to request a digital certificate, an entity generates a public key pair and presents the public key to the CA, together with some proof of identity. Optionally, the CA may generate a key pair for the requestor, and send the private key securely to him along with the signed public-key certificate. The CA performs the identity check, if satisfied, generates and signs a certificate for the requestor. To verify the origin of a public key, others check the CA’s signature contained in the certificate associated with the public key. In this way, the issued certificate securely binds the requestor’s identity to his public key, as the others trust the authority of the CA to perform the certification correctly.

4.5.2 The Certification Process

The certification process is as follows:

1. Subscriber (sender) applies to CA for digital certificate.
2. CA verifies subscriber’s identity and issues digital certificate.
3. CA publishes certificate to public, on-line repository.
4. Subscriber signs message with private key and sends message to second party.
5. Receiving party verifies digital signature with sender’s public key and requests verification of sender’s digital certificate from CA’s public repository.

6. Repository reports status of subscriber’s certificate.

There is a possibility that a private key associated with the public key contained in a certificate can become compromised prior to the expiration of the certificate. In such cases, the issuing CA should revoke the certificate as soon as possible. Most commonly used mechanism for certificate revocation is for the CA to publish it in a Certificate Revocation List (CRL). The CA issues and updates the CLR at regular intervals, and users should consult this list each time they perform signature validation.
verification. Thus, a certificate is said to be valid if and only if it has a valid CA’s signature, has not expired, and has not been listed in the CA’s most recent CRL.

A CA is not limited to issuing certificates to individuals and organizations it may also issue certificates to other CA’s. In this way, a hierarchy of CA’s is constructed. Starting from the top, the root CA issues certificates to CA’s at a level immediately below, which, in turn, issue certificates to CA’s at the next level, etc., and finally down to the leaves of the hierarchy where certificates are issued to users. This hierarchical model, involving a tree-structure of CA’s where each node certifies the nodes below itself, forms the basis of Public Key Infrastructure (PKI). PKI thus consists of a set of services that employ various CA’s to perform the issuance of public-key certificates and certificate management for the purpose of wide-scale application of public-key cryptosystems and digital signatures. Throughout this thesis, we assume that trusted CA’s issue all public keys used in the protocols, and that PKI is in place to provide the services of certificate application, issuance and revocation. In the following section, we briefly describe the SSL/TLS protocol, the network security protocol that utilizes all of the previously mentioned concepts and that our e-medical transaction protocols will be combined with.

4.6 CONCLUSION

In this chapter we have shown NTRU certificate verification system for generation Transaction of symmetric keys, public keys and private key between Peer to Peer or Peer to Group or Group-Group. We have presented a secure communication among different partiers in while key generation and sharing with very secure manner. We provide the sample result and code for generation of public key and private key using NTRU sharing with secure manner using Diffie Hellman key Exchange algorithm for Peer to Peer or Peer to Group or Group to Group.