CHAT ENCRYPTION SYSTEM USING NTRU ALGORITHM

PROJECT REPORT

For

Security of E-Based Systems (BCI3004)

Submitted By

Name	**Registration No.**
Aditya K Iyer	20BCI0084
Divyansh Singh	20BCI0081
Tarang Gupta	20BCI0310
Utkarsh Singh	20BCI0284
Akshat Pattiwar	20BCI0258

Under the guidance of

Prof. Gopichand G

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

Abstract

Due to high development in field of quantum computing, in future widely used public key cryptosystems such as RSA, ECC and Diffie Hellman Key Exchange would be easily breakable. One such famous algorithm to achieve this task is Shor's algorithm. It finds all the prime factors of an integer and achieves so in complexity class BQP. Hence, there would be a rising need for an algorithm which is quantum-resistant. The aim of this review paper is to implement an advanced version of one such algorithm, known as NTRU encryption. In this review paper, we will be implementing NTRU encryption, a quantum resistant open-source public-key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data. This cryptosystem is resistant to attacks executed through Shor's algorithm and other quantum computer-based attacks. NTRUEncrypt will be implemented in this review paper as a chat encryption algorithm. It was found that NTRUEncrypt has successfully encrypted the message being sent and decrypted it on receiver's side. Message was first converted in form of a polynomial which was encrypted through said encryption scheme. This algorithm was found to be much faster compared to other cryptosystems such as RSA and ECC, by around five and three orders of magnitude respectively. NTRUEncrypt is a better alternative as compared to other public key cryptosystems such as RSA, due to the fact that the development in field of quantum computing is occurring at a rapid pace, coupled with the fact that NTRUEncrypt is much faster than other cryptosystems. Preparation for the preconditions for quantum computers can be ensured in a direction to an abundant precaution against hacking and to comply with security goals through this encryption scheme.

Keywords: NTRUEncrypt, quantum computing, Shor’s algorithm, lattice-based cryptography, truncated polynomial ring

1. Introduction

Public key cryptosystems such as RSA have been used for about 40 years till now. Safety of RSA and Diffie Hellman key exchange lies in highly difficult task of solving discrete d^{th} root problem and discrete logarithm problem respectively. No classical computer can solve these problems in polynomial time. However, in case of quantum computers things are a bit different.

If a quantum computer with a sufficient number of qubits could operate such that there is no capitulation to quantum noise and other quantum-decoherence phenomena, then it was found that Shor's algorithm, a polynomial-time quantum computer algorithm for integer factorization can break public-key cryptography schemes, such as: -

- RSA
- The Finite Field Diffie-Hellman key exchange
- The Elliptic Curve Diffie-Hellman key exchange

And do so in polynomial time (it takes quantum gates of $O(\log N)^2(\log\log N)(\log\log\log N)$ to solve integer factorization problem using fast multiplication). With certain modifications to this algorithm, it can be implemented to solve discrete logarithm problem in polynomial time too. Hence after the onset of affordable quantum computers in the future, vulnerability of cryptosystems based on abovementioned problems as the backbone of their security would be increased to a point where their usage would be meaningless. A good example would be the widely used chat application WhatsApp, which uses Signal Protocol for end-to-end encryption. Signal Protocol provided end-to-end encryption using ECDH Curve25519 key exchanges, but this is not quantum-safe and can cause a major security issue in the future.

Hence, there would be a high need of quantum resistant encryption algorithms, an encryption algorithm which cannot be bested by a quantum computer. The aim of this paper is to design a simple but effective chat application which is safe to attacks made by a quantum computer. This also implies that the algorithm to be implemented for chat security must follow already existing basic cryptographic standards.

This can be achieved by implementing NTRUEncrypt (stands for N^{th} Degree Truncated Polynomial. Ring Units), a quantum resistant algorithm which depends on the surmised difficulty of splitting some polynomials in a truncated polynomial ring into the quotients of two polynomials with very small coefficients and carefully selected parameters to thwart published attacks. Objective of this review paper would be to implement NTRUEncrypt scheme to a chat application, so as to tackle security issues generated from both classical cryptographic and quantum threats collectively.

2. Objective

Our aim in this project is to learn and use the NTRU algorithm method to create a chat system that will provide security of data to the users. We look forward to successfully building this system so that no input restrictions are applied, and we have made the system more efficient by reducing the time involved in creating and validating results.

3. Literature Survey

Title	Author, Year	Journal	Findings
Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers	Edward Gerjuoy, June 2005	American Association of Physics Teachers	The RSA encryption protocol is explained in this paper, as well as the various quantum computer manipulations that make up the Shor algorithm, how the Shor algorithm performs factoring, and the precise sense in which a quantum computer using Shor's algorithm can be said to factor very large numbers with less computational effort than a classical computer. It is clear that factoring N necessitates a large number of iterations of the procedure. The results show that the likelihood of attaining a successful factorization on a single run is roughly twice as high as previously thought.
Quantum Resistant Public Key Cryptography: A Survey	Ray A. Perlner, David A. Cooper, April 2009	National Institute of Standards and Technology	They provided a survey of some of the public key cryptographic algorithms that have been created that, while not widely used, are believed to be resistant to quantum computing-based attacks, and discussed some of the issues that protocol designers may need to take into account if these algorithms are needed in the future.

Quantum Algorithms for Computing Short Discrete Logarithms and Factoring RSA Integers	Martin Eker and Johan H, June 2017	Springer International	They explain how key problems like factoring RSA integers and determining group order under side information may be recast as small discrete logarithm problems in this study. This results in a less difficult procedure for factoring RSA integers than Shor's general factoring algorithm, in the sense that it places fewer demands on the quantum computer. The primary challenge in both their and Shor's algorithms is computing a modular exponentiation in superposition. When factoring an n-bit integer, Shor's approach uses an exponent of length 2n bits, but their algorithm uses slightly more than n/2 bits.
Countermeasures against Power Analysis Attacks for the NTRU Public Key Cryptosystem	Mun-Kyu LEE, Jeong Eun SONG, Dooho CHOI and Dong-Guk HAN, January 2010	The Institute of Electronics, Information and Communication Engineers	They demonstrated that a common software implementation of NTRU is vulnerable to simple power analysis and correlation power analysis, which includes a second-order power attack, in this study. They also provided unique countermeasures to prevent these attacks and conducted experiments to assess their countermeasures' performance overheads. According to their findings, the required memory and execution time overheads are only 8.17 percent and 9.56 percent, respectively, as compared to a Tmote Sky with an MSP430 CPU.

NTRUEncrypt – A Quantum Proof Replacement to RSA Cryptosystem	Kunal Meher , Divya Midhunchakkaravarthy, September 2020	International Journal of Advanced Trends in Computer Science and Engineering	The goal of encryption is to offer a secure communication environment and to protect information from unauthorized access. Asymmetric cryptosystems and symmetric cryptosystems are the two types of cryptosystems. Public Key Encryption is another name for an asymmetric cryptosystem (PKE). They looked at two asymmetric cryptosystems, RSA and NTRU, in this article. NTRU has keys that are relatively short and easy to construct, as well as high speed and low memory needs. Because NTRU is a relatively new cryptosystem, they spent more time discussing it in the paper.

Speed Records for NTRU	Jens Hermans, Frederik Vercauteren, and Bart Preneel, March 2010	Springer	For the first time, NTRUEncrypt is implemented on a GPU utilizing the CUDA platform in this paper. This operation lends itself excellently to parallelization, as illustrated, and performs exceedingly well when compared to similar security levels for ECC and RSA, with speedups of three to five orders of magnitude. The goal is to achieve a high throughput, which is achieved by conducting a large number of encryptions and decryptions in parallel. At a security level of 256 bits, a current GTX280 GPU can achieve a throughput of up to 200 000 encryptions per second. This results in a theoretical data throughput of 47.8 megabytes per second. When compared to a symmetric cipher (a comparison that isn't often made), this is only about 20 times slower than a contemporary AES implementation on a GPU.
The Post-Quantum Signal Protocol Secure Chat in a Quantum World	Ines Duits, February, 2019	University of Twente	-For messaging apps like WhatsApp, Skype, Facebook private Messenger, Google Allo, and Signal, the Signal Protocol enables end-to-end encryption, forward secrecy, backward secrecy, authentication, and deniability. The ECDH Curve25519 key exchanges and SHA-512 key derivation are used in the Signal Protocol to accomplish this. The ECDH key exchange, on the other hand, is not quantum-safe; if opponents had a quantum computer, they could gain the key and read along. They put ten various post-quantum key exchange mechanisms (KEMs) to the test,

			as well as the Diffie-Hellman algorithm based on post-quantum supersingular isogeny (SIDH). They also looked at various variants of a partially post-quantum Signal Protocol.
Insight into the Operation of NTRU and a Comparative Study of NTRU, RSA and ECC Public Key Cryptosystems	Juliet N. Gaithuru, Majid Bakhtiari,July 2014	Malaysian Software Engineering Conference	The functioning of the NTRU public key cryptosystem, the restrictions for selecting its parameters, and its security are all described in detail in this paper, which is followed by a practical demonstration of its operation. This paper also discusses NTRU's benefits and weaknesses, as well as a comparison of NTRU, RSA, and ECC's performance. Finally, they provide a conclusion of possible topics for further investigation in order to guarantee assurance of NTRU security.
SMS Encryption using NTRU Algorithm	Er. Amanpreet Kaur, Er. Navpreet Singh, April 2015	International Journal of Advanced Research in Computer Science & Technology	On chat applications, NTRU, a rapid encryption method, has yet to be applied. The key benefit of this technique is that it encrypts and decrypts data quickly. Because chat applications must maintain a high level of speed, it is also vital to consider security measures as well as dependability issues. All of the results are compatible with the NTRU. The proposed system has a shorter execution time. The time it takes to decrypt data is cut in half.
Performance Analysis of Public key Cryptographic Systems RSA and NTRU	Narasimham Challa and Jayaram Pradhan, August 2007	IJCSNS	By developing the calculation algorithms and comparing their experimental running times, the performance characteristics of NTRU and RSA are observed. The speed of encryption and decryption is O (n log (n))

			operations. The RSA complexity is O (n^3) operations, but the NTRU complexity is O (n log(n) operations. They proposed and presented these two well-known Public Key Cryptographic Systems in this article, and they were implemented to test their performance on a variety of text files of various sizes.
Modifying Shor's algorithm to compute short discrete logarithms	Martin Ekera, December 2016	KTH Royal Institute of Technology	They review Shor's approach for computing discrete logarithms in F* p on a quantum computer in this study and tweak it to compute logarithms d in prime order q groups in the exceptional case where d<< q. They initially developed a modified approach for computing logarithms on the general interval.

4. NTRUEncrypt and its Working

The NTRU Encrypt Public Key Cryptography System, also known as the NTRU Encryption Algorithm, is a NTRU-lattice-based alternative to RSA and Elliptic Curve Cryptography (ECC), which is based on the smallest vector problem in the lattice which is considered to be non-breakable through usage of quantum computers. It relies on the presumed difficulty of factoring certain polynomials through a truncated polynomial ring into a quotient of two polynomials having very small coefficients. Another major advantage of for usage of NTRUEncrypt lies in its high execution speed. This is because operations related to encryption and decryption are much faster than other asymmetric cryptographic schemes such as RSA, ElGamal and Elliptic Curve Cryptography as both encryption and decryption use only simple polynomial multiplication.

Specifically, NTRU operations are supported on objects through a truncated polynomial ring: -

$\mathbf{R = Z[X]/(X^N\text{-}1)}$

with convolution multiplication such that all polynomials in the ring have integer coefficients and degree at most *N*-1. The general form of such polynomial would be: -

$$\mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1}$$

NTRU cryptosystem is specified by three integer parameters (N, p, q) which represent the maximal degree for all polynomials within the truncated ring R, a little modulus and outsized modulus, respectively, with the assumption that N is prime, q is always larger than p, and p and q are coprime; and 4 sets of polynomials L_f, L_g, L_m and L_r (a polynomial part of the private key, a polynomial for generation of the general public key, the message and a blinding value, respectively), all of degree at most N-1.

Working

The algorithm can be classified broadly in three major steps namely: -

i. Key Generation
ii. Encryption
iii. Decryption

I. Key Generation

Suppose we need to send a message from **A** to **B**. This requires the generation of both public and private key. Public key is known to both A and B but private key is known only to B (receiver). Let, the three abovementioned integer parameters be **(N, p, q)**. For key generation, two polynomial pairs **f** and **g** are required each of whose degree is at most **N-1** and whose coefficients are in the set {-1,0,1}, such that number of 1s as coefficients of **g** should be equal to number of –1s but they should not be equal in case of **f**. The polynomial **f** must satisfy the additional requirement that the inverses modulo q (f_q) and modulo p (f_p) (this can be calculated using Euclidean algorithm and Bézout's identity) exist, which means that: -

$$f.f_p = 1 \pmod p$$

And,

$$f.f_q = 1 \pmod q$$

Hence, the **f** chosen should be invertible.

The polynomials f, g and f_p are B's private key. The public key **h** of B is calculated by: -

$$h = pf_q.g \pmod q$$

II. Encryption

A, the sender will send the message in the form of of a polynomial m whose coefficients will lie in the range **[-p/2,p/2]**. Message polynomial can be represented in binary format. Now, A is required to choose a random polynomial **r** such that it has small coeffcients, though these coefficients are not restricted to the set {-1,0,1}. This polynomial is also termed as 'blinding value', due to its property/usage for obscuring the message polynomial. This blinding value must be kept as a secret on A's side. Using public key of B, r and m, the encrypted message **e** is obtained as: -

$$e = r.h + m \pmod q$$

III. Decryption

B, the receiver will decrypt the message using his private key by following steps: -

Step 1: B will multiply **f** from his private key with the the obtained encrypted message **e** as shown below: -

$$a = f.e \pmod q$$

This can be simplified as: -

$$a = f.(r.h+m) \pmod q$$

$$a = f.(r.pf_q.g+m) \pmod q$$

$$a = pr.g + f.m \pmod q$$

Now, we choose the coefficients of **a** in the interval [-*q*/2, *q*/2] to due to the fact that the original message might not be properly recovered since Alice chooses the coordinates of her message **m** in the interval [-*p*/2, *p*/2]. This means all coefficients of **pr.g + f.m** already lie within the interval [-*q*/2, *q*/2] because the polynomials **r**, **g**, **f** and **m** and prime *p* all have coefficients that are small compared to *q*. This means that all coefficients are left unchanged during reducing modulo *q* and hence the original message may be obtained completety.

Step 2: Now, a modulo p is calculated: -

$$b = a \pmod p$$

This is simplified as: -

$$b = f.m \pmod p$$ (as **pr.g** is a multiple of p)

Step 3: Now, b is multiplied with B’s private key $\mathbf{f_p}$ such that: -

$$c = f_p.b$$

$$c = f_p.f.m \pmod p$$

Now, as $f_p.f = 1$: -

$$c = m \pmod p$$

Hence, we get the original message as the result of decryption.

5. Proposed Architecture for Chat Implementation

Now, for better understanding in implementation part, representation of abovementioned three steps are shown below through flowcharts: -

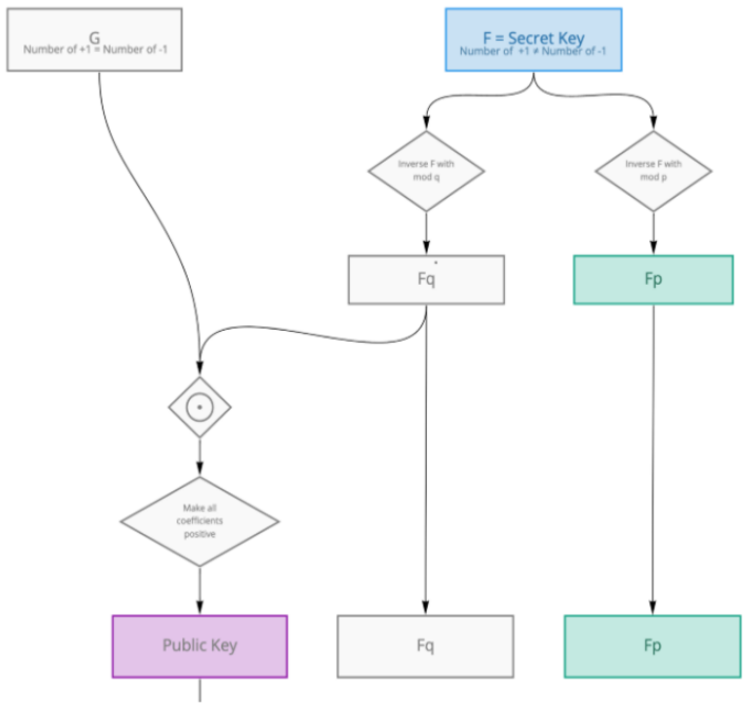

G
Number of +1 = Number of -1
F = Secret Key
Number of +1 ≠ Number of -1
Inverse F with mod q
Inverse F with mod p
Fq
Fp
⊙
Make all coefficients positive
Public Key
Fq
Fp

Fig i-Key Generation

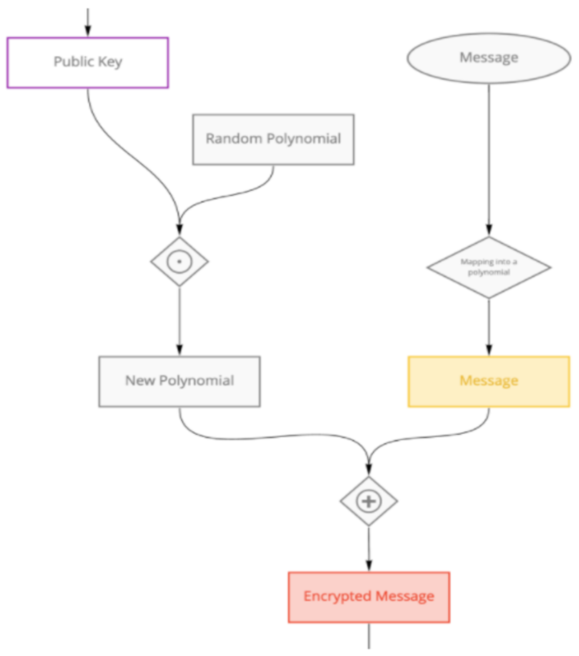


Fig ii- Encryption

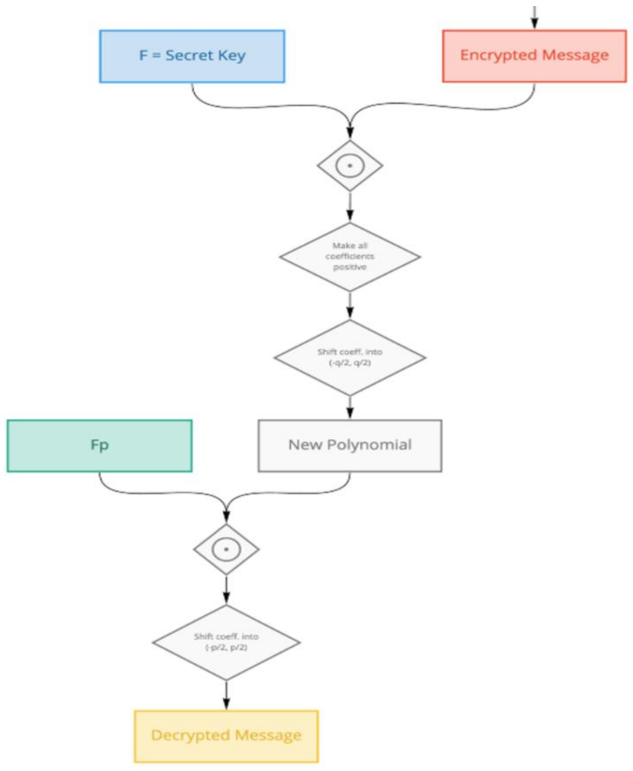

F = Secret Key
Encrypted Message
Make all coefficients positive
Shift coeff. into (-q/2, q/2)
Fp
New Polynomial
Shift coeff. into (-p/2, p/2)
Decrypted Message

Fig iii-Decryption

In this application, encrypting of texts will occur on client's (sender) side and decrypting on receiver's side, the message will be passed through a server which send the message to respective users through provided local addresses. A small framework of this application through a diagramtic representation is shown below: -

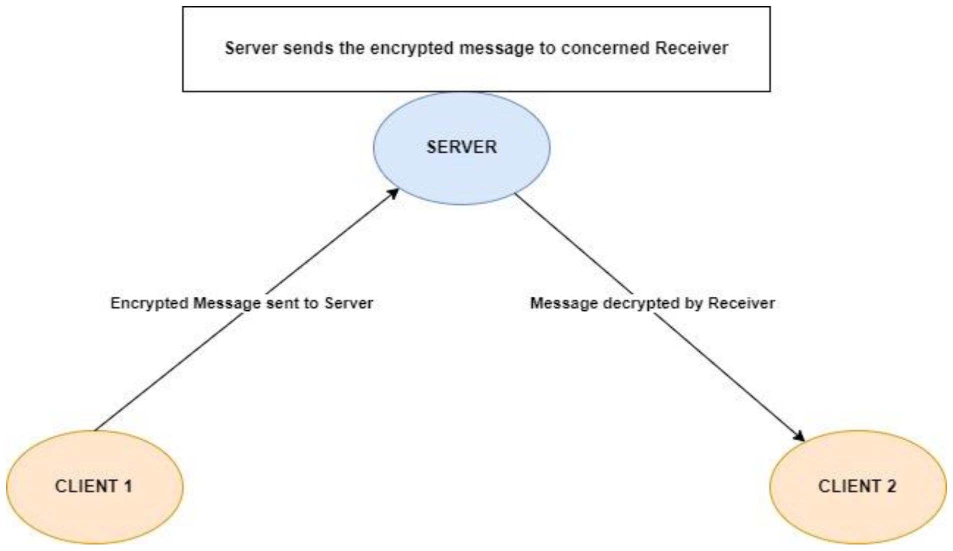

Server sends the encrypted message to concerned Receiver
SERVER
Encrypted Message sent to Server
Message decrypted by Receiver
CLIENT 1
CLIENT 2

Fig iv-Server and client

6. Standalone Application Implementation

First, helper functions are generated so as to calculate Fraction Modulo, modular inverse, gcd etc.

```
def egcd(a, b):
    x, y, u, v = 0, 1, 1, 0
    while a != 0:
        q, r = b // a, b % a
        m, n = x - u * q, y - v * q
        b, a, x, y, u, v = a, r, u, v, m, n
    gcdVal = b
    return gcdVal, x, y


# Extended GCD algorithm to find modular inverses:
def modinv(a, m):
    gcdVal, x, y = egcd(a, m)
    if gcdVal != 1:
        return None  # modular inverse does not exist
    else:
        return x % m


# Fraction Modulo
def fracMod(f, m):
    [tmp, t1, t2] = egcd(f.denominator, m)
    if tmp != 1:
        print("ERROR GCD of denominator and m is not 1")
        1 / 0
    else:
        out = modinv(f.denominator, m) * f.numerator % m
        return out
```

- For, implementing encryption and decryption, poly module is imported for performing polynomial operations. Basic pseudocodes for key generation, encryption and decryption are shown below: -

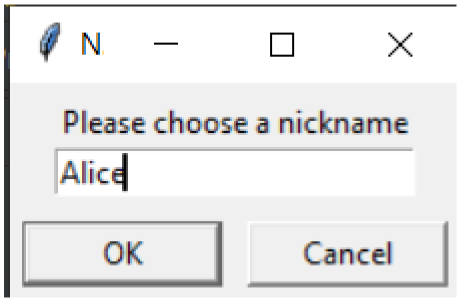
```
def key_generation()

  [gcd_f, s_f, t_f] = poly.extEuclidPoly(self.f, D) #Extended Euclidian algorithm implemented

   self.f_p = poly.modPoly(s_f, self.p)
   self.f_q = poly.modPoly(s_f, self.q)

x = poly.multPoly(self.f_q, self.g)
   self.h = poly.reModulo(x, D, self.q)

   # return h, self.f, f_p

def get_pubkey(self):
   return self.h

def get_privkeys(self):
   return self.f, self.f_p
```

```
def NTRUencrypt(message, p, q, D, h):

for i in range(len(msg1)):
        msg1[i] = ord(msg1[i])
    print(msg1)

    randPol = [-1, -1, 1, 1]
    encmsg = []
    l = []

    for i in msg1:

                  msg = poly.DecimalToBinary(i)

                     l.append(len(msg))

     e_tilda = poly.addPoly(poly.multPoly(poly.multPoly([p], randPol),h), msg)
        e = poly.reModulo(e_tilda, D, q)
        encmsg.append(e)

return encmsg, l
```

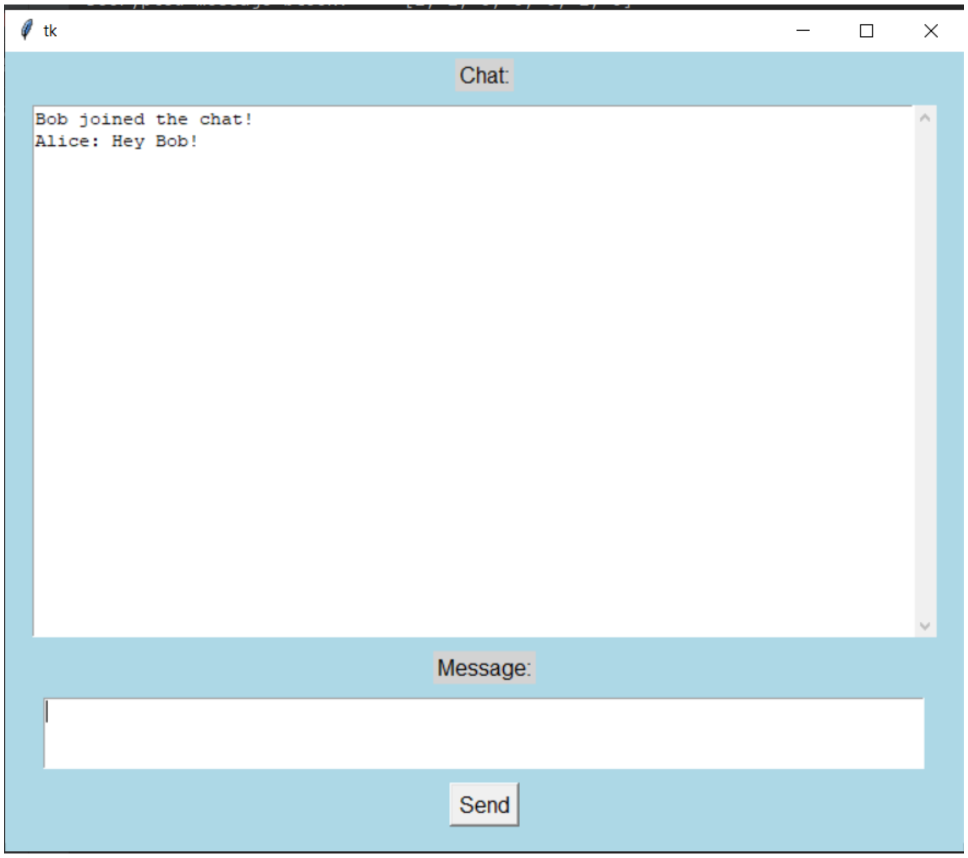
```
def decrypt(encmsg, f, f_p, p, q, D, l):
    decmsg = []
    j = 0
    for e in encmsg:
        tmp = poly.reModulo(poly.multPoly(f, e), D, q)
        centered = poly.cenPoly(tmp, q)
        m1 = poly.multPoly(f_p, centered)
        tmp = poly.reModulo(m1, D, p)
decmsg.append(tmp[:l[j]])
        j = j + 1
fm = []

    for i in decmsg:
        decmsgpart = poly.binaryToDecimal(int(''.join(list(map(str, i)))))
        fm.append(decmsgpart)

  message = ''.join(list(map(chr, fm)))

    return message
```

6.1 Front End

After the initialization of server, the application on the client's side is run. Client is asked to enter a nickname for identification in chats: -

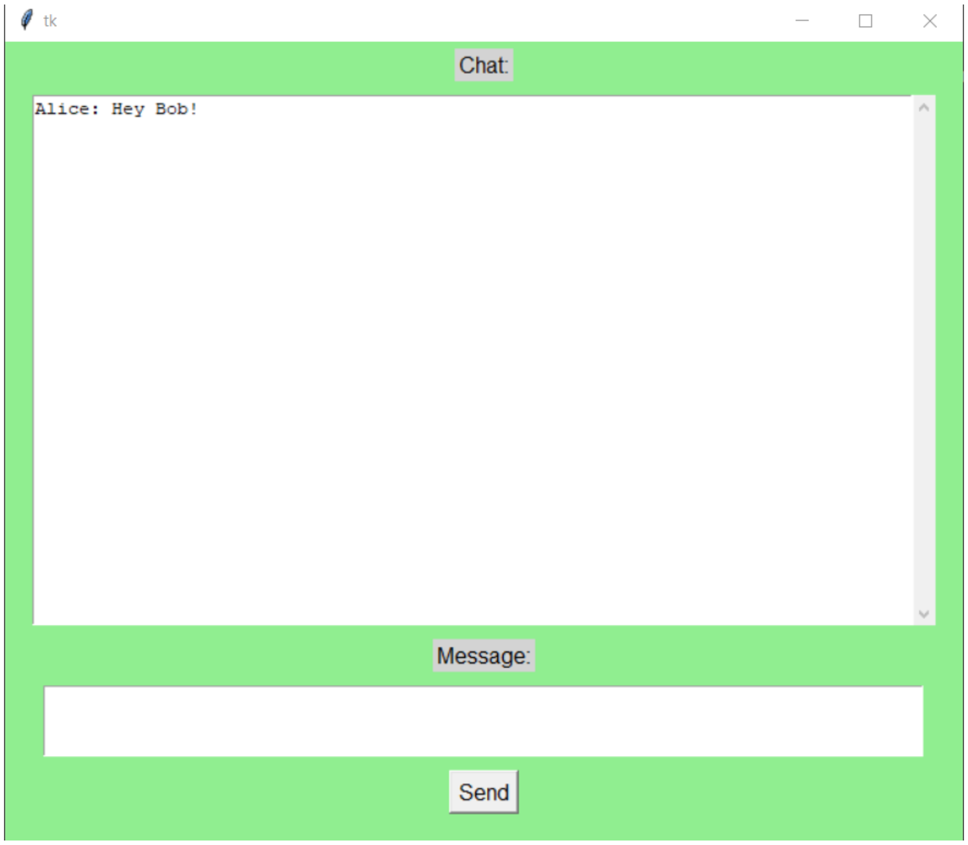


Sender's Side: -



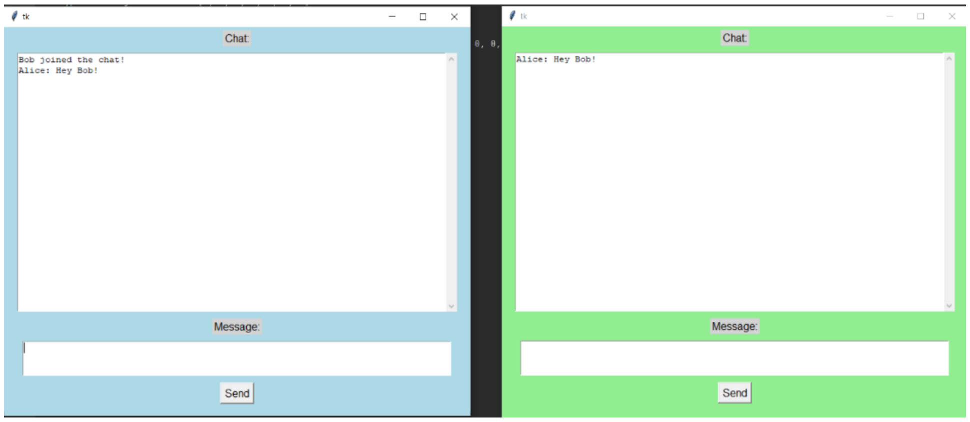
```
Bob joined the chat!
Alice: Hey Bob!
```

Receiver's Side: -



```
Alice: Hey Bob!
```

Combined Interface: -

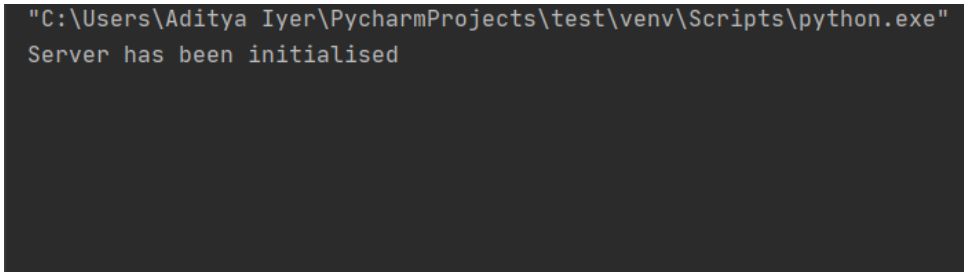


```
Bob joined the chat!
Alice: Hey Bob!
```

```
Alice: Hey Bob!
```

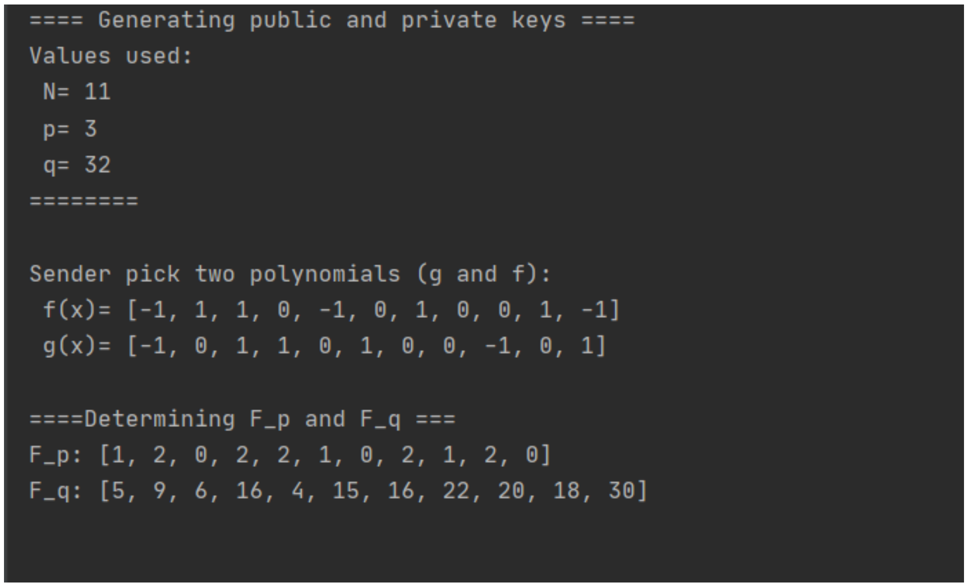
6.2 Back End

Initialisation of Server: -



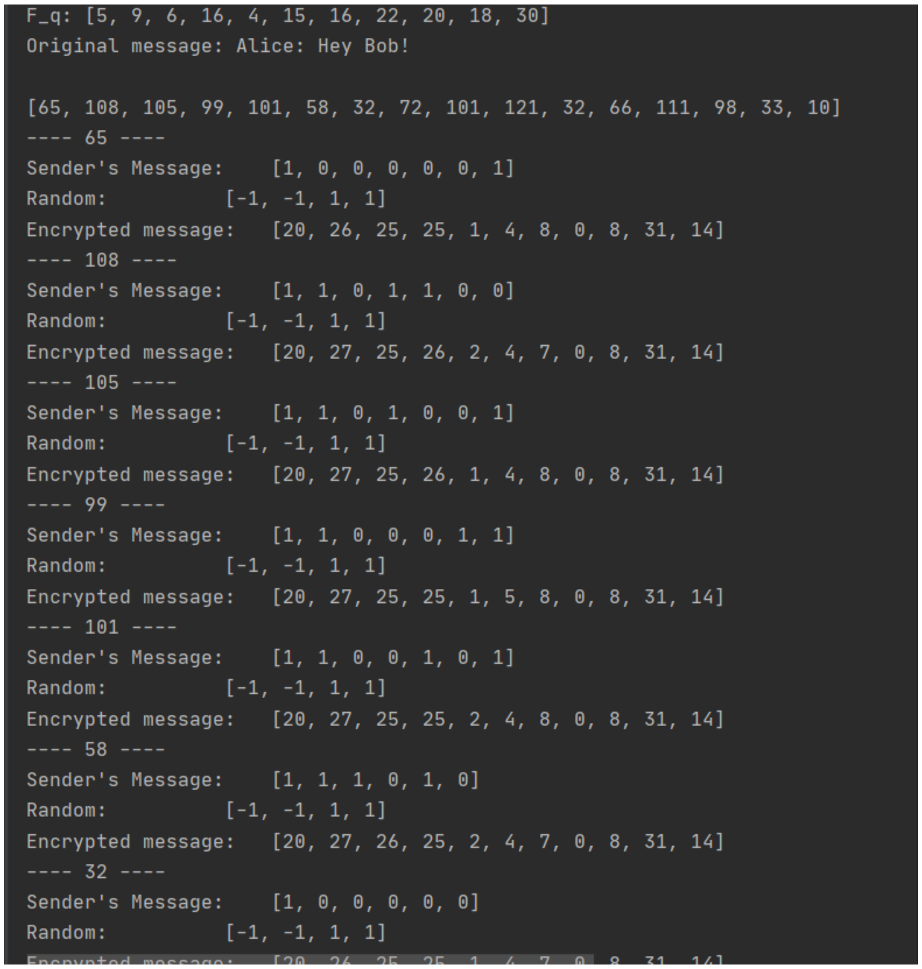
```
"C:\Users\Aditya Iyer\PycharmProjects\test\venv\Scripts\python.exe"
Server has been initialised
```

Generation of f_p and f_q: -



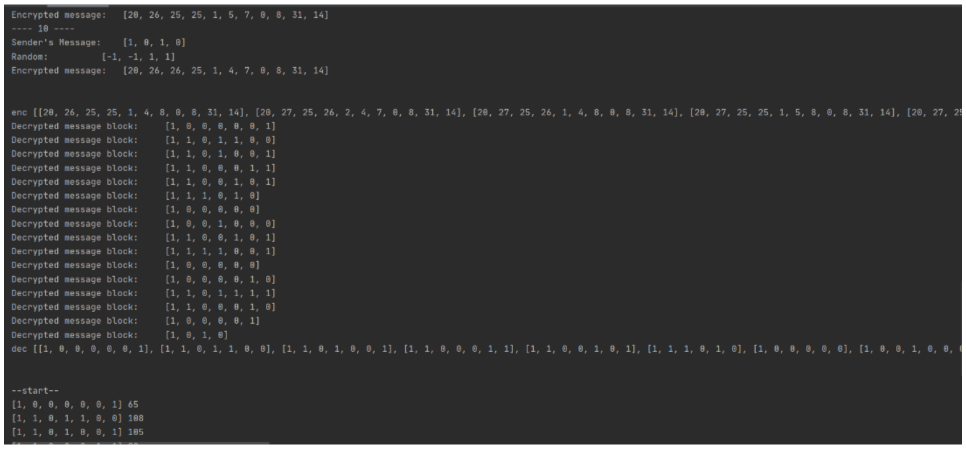
```
==== Generating public and private keys ====
Values used:
 N= 11
 p= 3
 q= 32
========

Sender pick two polynomials (g and f):
 f(x)= [-1, 1, 1, 0, -1, 0, 1, 0, 0, 1, -1]
 g(x)= [-1, 0, 1, 1, 0, 1, 0, 0, -1, 0, 1]

====Determining F_p and F_q ===
F_p: [1, 2, 0, 2, 2, 1, 0, 2, 1, 2, 0]
F_q: [5, 9, 6, 16, 4, 15, 16, 22, 20, 18, 30]
```

Encryption through NTRU scheme of each character’s UNICODE: - (here shown till 32 due to space constraints)



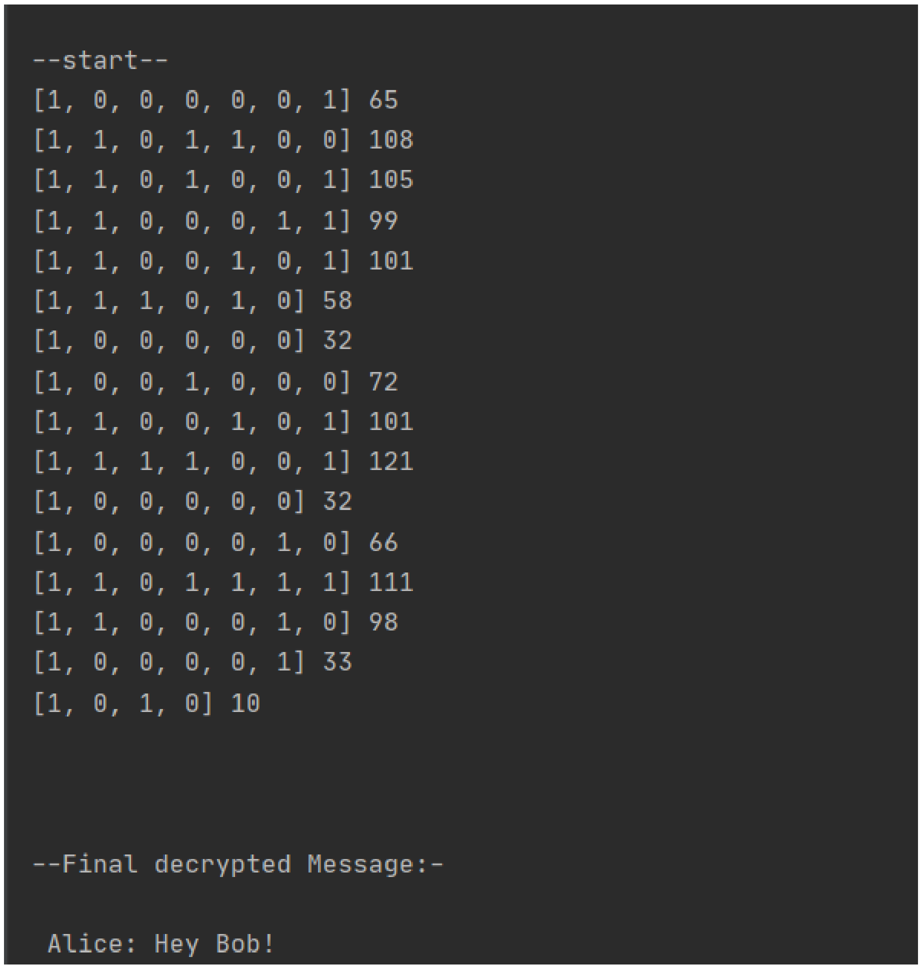
```
F_q: [5, 9, 6, 16, 4, 15, 16, 22, 20, 18, 30]
Original message: Alice: Hey Bob!

[65, 108, 105, 99, 101, 58, 32, 72, 101, 121, 32, 66, 111, 98, 33, 10]
---- 65 ----
Sender's Message:    [1, 0, 0, 0, 0, 0, 1]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 26, 25, 25, 1, 4, 8, 0, 8, 31, 14]
---- 108 ----
Sender's Message:    [1, 1, 0, 1, 1, 0, 0]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 27, 25, 26, 2, 4, 7, 0, 8, 31, 14]
---- 105 ----
Sender's Message:    [1, 1, 0, 1, 0, 0, 1]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 27, 25, 26, 1, 4, 8, 0, 8, 31, 14]
---- 99 ----
Sender's Message:    [1, 1, 0, 0, 0, 1, 1]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 27, 25, 25, 1, 5, 8, 0, 8, 31, 14]
---- 101 ----
Sender's Message:    [1, 1, 0, 0, 1, 0, 1]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 27, 25, 25, 2, 4, 8, 0, 8, 31, 14]
---- 58 ----
Sender's Message:    [1, 1, 1, 0, 1, 0]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 27, 26, 25, 2, 4, 7, 0, 8, 31, 14]
---- 32 ----
Sender's Message:    [1, 0, 0, 0, 0, 0]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 26, 25, 25, 1, 4, 7, 0, 8, 31, 14]
```

Decryption through NTRU scheme of each character’s UNICODE: -



```
Encrypted message:   [20, 26, 25, 25, 1, 5, 7, 0, 8, 31, 14]
---- 10 ----
Sender's Message:    [1, 0, 1, 0]
Random:          [-1, -1, 1, 1]
Encrypted message:   [20, 26, 26, 25, 1, 4, 7, 0, 8, 31, 14]


enc [[20, 26, 25, 25, 1, 4, 8, 0, 8, 31, 14], [20, 27, 25, 26, 2, 4, 7, 0, 8, 31, 14], [20, 27, 25, 26, 1, 4, 8, 0, 8, 31, 14], [20, 27, 25, 25, 1, 5, 8, 0, 8, 31, 14], [20, 27, 25
Decrypted message block:     [1, 0, 0, 0, 0, 0, 1]
Decrypted message block:     [1, 1, 0, 1, 1, 0, 0]
Decrypted message block:     [1, 1, 0, 1, 0, 0, 1]
Decrypted message block:     [1, 1, 0, 0, 0, 1, 1]
Decrypted message block:     [1, 1, 0, 0, 1, 0, 1]
Decrypted message block:     [1, 1, 1, 0, 1, 0]
Decrypted message block:     [1, 0, 0, 0, 0, 0]
Decrypted message block:     [1, 0, 0, 1, 0, 0, 0]
Decrypted message block:     [1, 1, 0, 0, 1, 0, 1]
Decrypted message block:     [1, 1, 1, 1, 0, 0, 1]
Decrypted message block:     [1, 0, 0, 0, 0, 0]
Decrypted message block:     [1, 0, 0, 0, 0, 1, 0]
Decrypted message block:     [1, 1, 0, 1, 1, 1, 1]
Decrypted message block:     [1, 1, 0, 0, 0, 1, 0]
Decrypted message block:     [1, 0, 0, 0, 0, 1]
Decrypted message block:     [1, 0, 1, 0]
dec [[1, 0, 0, 0, 0, 0, 1], [1, 1, 0, 1, 1, 0, 0], [1, 1, 0, 1, 0, 0, 1], [1, 1, 0, 0, 0, 1, 1], [1, 1, 0, 0, 1, 0, 1], [1, 1, 1, 0, 1, 0], [1, 0, 0, 0, 0, 0], [1, 0, 0, 1, 0, 0,


--start--
[1, 0, 0, 0, 0, 0, 1] 65
[1, 1, 0, 1, 1, 0, 0] 108
[1, 1, 0, 1, 0, 0, 1] 105
```

Recombination of decrypted UNICODES: -



```
--start--
[1, 0, 0, 0, 0, 0, 1] 65
[1, 1, 0, 1, 1, 0, 0] 108
[1, 1, 0, 1, 0, 0, 1] 105
[1, 1, 0, 0, 0, 1, 1] 99
[1, 1, 0, 0, 1, 0, 1] 101
[1, 1, 1, 0, 1, 0] 58
[1, 0, 0, 0, 0, 0] 32
[1, 0, 0, 1, 0, 0, 0] 72
[1, 1, 0, 0, 1, 0, 1] 101
[1, 1, 1, 1, 0, 0, 1] 121
[1, 0, 0, 0, 0, 0] 32
[1, 0, 0, 0, 0, 1, 0] 66
[1, 1, 0, 1, 1, 1, 1] 111
[1, 1, 0, 0, 0, 1, 0] 98
[1, 0, 0, 0, 0, 1] 33
[1, 0, 1, 0] 10



--Final decrypted Message:-

 Alice: Hey Bob!
```

7. Web Application Implementation

To make NTRU chat encryption accessible to a wider audience, a web application implementation can be developed. This allows users to easily encrypt and decrypt their messages through a user-friendly interface, without the need for any specialized software or hardware.

The web application implementation of secure chat application can be developed using **Django Stack** and **NTRU Cryptography**. These frameworks provide the necessary tools to build a secure and scalable web application that can handle multiple users and messages in real-time.

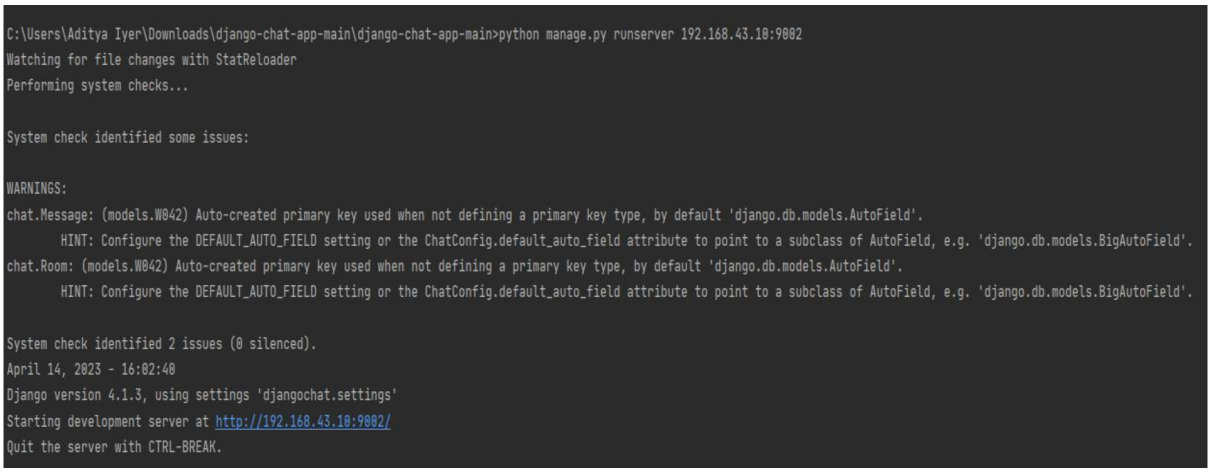
In the implementation of a web based NTRU chat encryption system, there are some key considerations that were made note of. These include:

1. <u>Security</u>: As with any encryption system, security is of paramount importance. The web application must ensure that all messages are encrypted and decrypted securely, and that the user's private keys are kept confidential.
2. <u>User Interface</u>: A user-friendly interface is essential to the success of the web application. Users should be able to easily encrypt and decrypt their messages, view their message history, and manage their keys.
3. <u>Scalability</u>: The web application should be designed to handle multiple users and messages in real-time. This requires careful consideration of the architecture and the use of efficient data structures and algorithms.
4. <u>Accessibility</u>: The web application should be accessible from any device with an internet connection. This can be achieved by making the application responsive and optimizing it for different screen sizes and resolutions.

The web application implementation of NTRU chat encryption is an important step towards making secure online communication accessible to everyone. By considering the key considerations outlined above, we attempt to build a secure, user-friendly, and scalable web application that can protect users' messages from eavesdropping and tampering and resistant from quantum attacks.

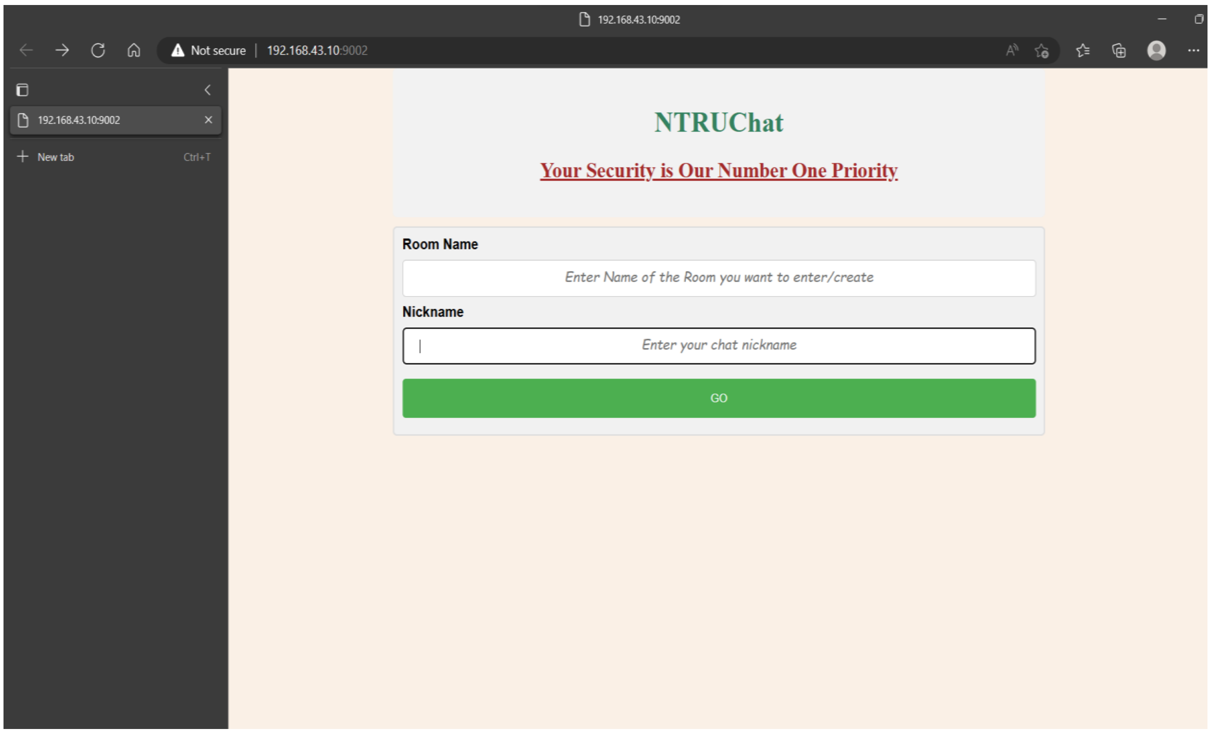
The working NTRUEncrypt in the backend of this web application is same as the working of tkinter standalone application; hence the focus is kept on the frontend. Following is the implementation of our web chat application named '**NTRUChat**':

1. Initialising the server, the IP could be specified in front of runserver command, by default it keeps the IP address as home IP and port as 8000. In this case it is connected to IP: 192.168.43.10 and Port: 9002.



```
C:\Users\Aditya Iyer\Downloads\django-chat-app-main\django-chat-app-main>python manage.py runserver 192.168.43.10:9002
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
chat.Message: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the ChatConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.
chat.Room: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the ChatConfig.default_auto_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 2 issues (0 silenced).
April 14, 2023 - 16:02:40
Django version 4.1.3, using settings 'djangochat.settings'
Starting development server at http://192.168.43.10:9002/
Quit the server with CTRL-BREAK.
```

2. Initial Interface which prompts the user to enter username and the chat room he would like to enter. If the chat room does not exist, a new chat room will be initialised. Both Alice and Bob's initial interface is shown here:



NTRUChat

Your Security is Our Number One Priority

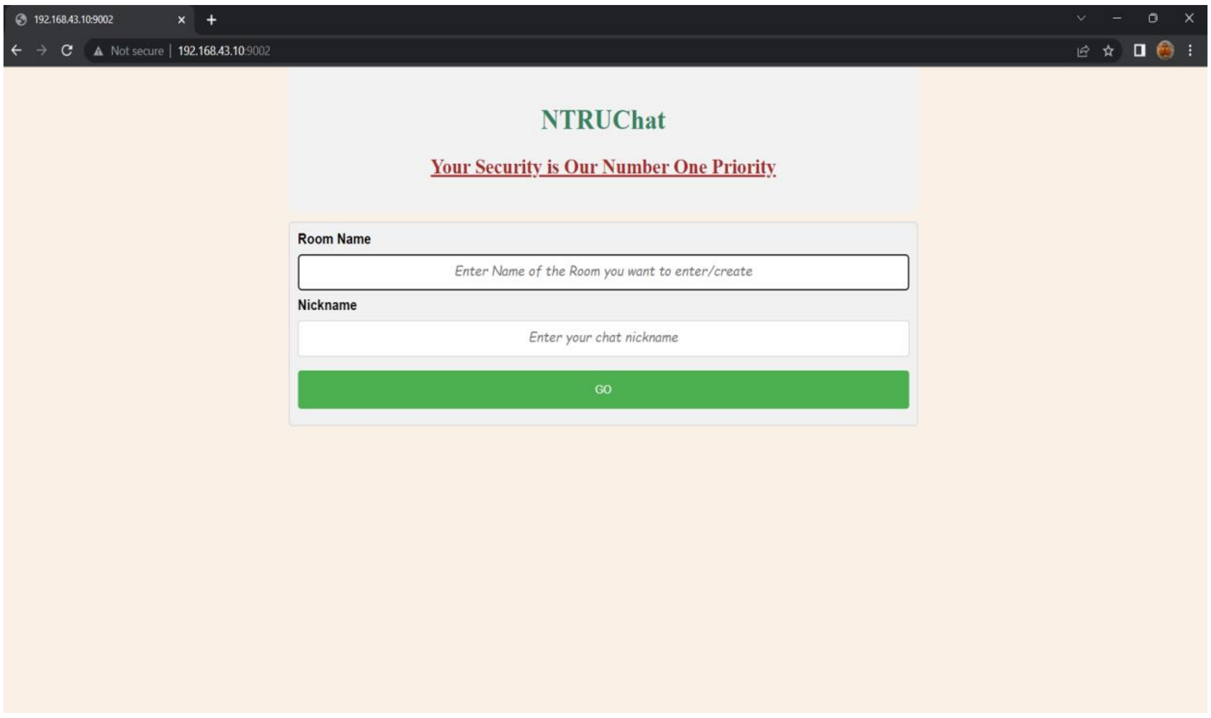
Room Name

Enter Name of the Room you want to enter/create

Nickname

Enter your chat nickname

GO


192.168.43.10:9002

Not secure | 192.168.43.10:9002

NTRUChat

Your Security is Our Number One Priority

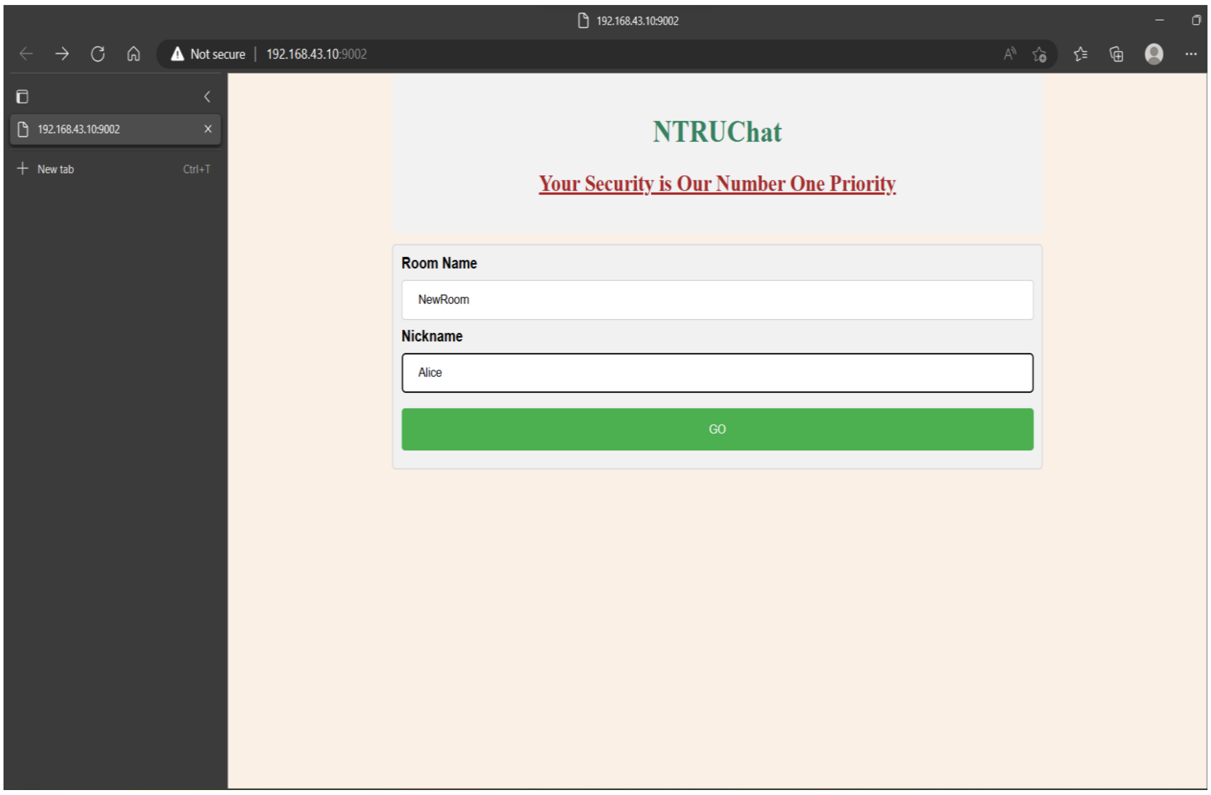
Room Name

Enter Name of the Room you want to enter/create

Nickname

Enter your chat nickname

3. Username and chatroom name entered:


192.168.43.10:9002

Not secure | 192.168.43.10:9002

NTRUChat

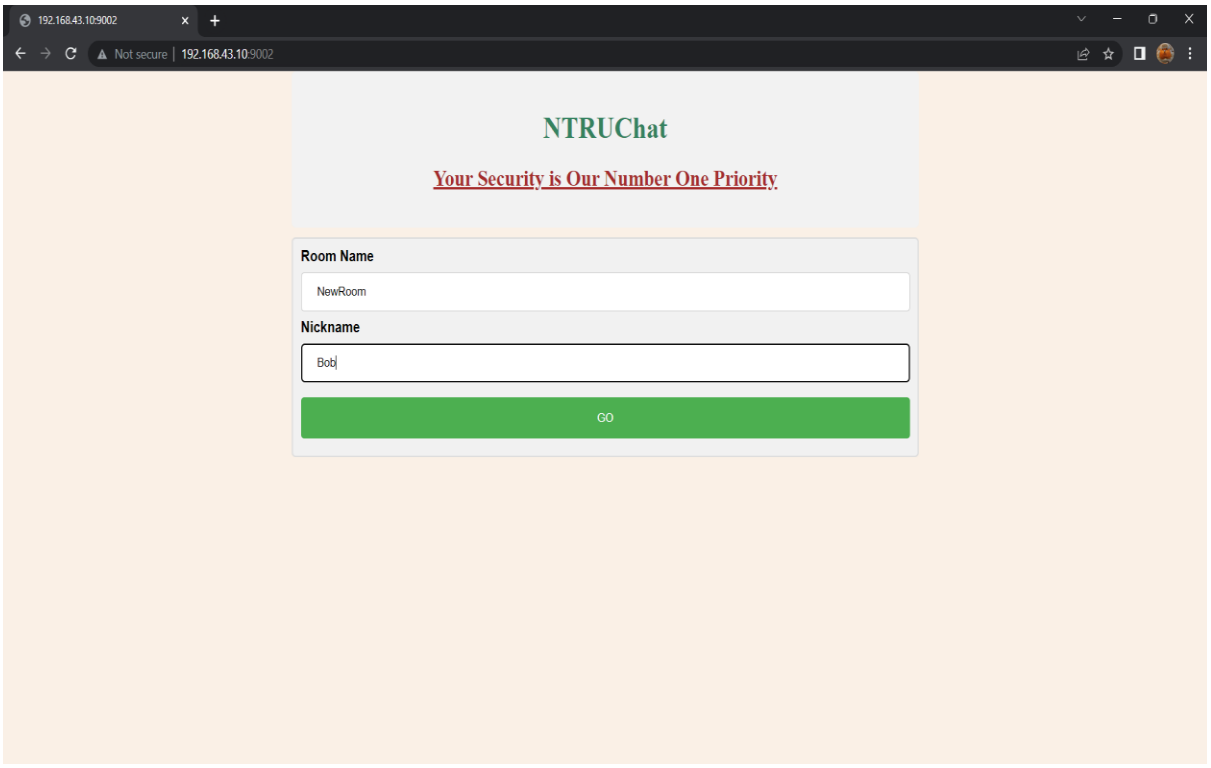
Your Security is Our Number One Priority

Room Name

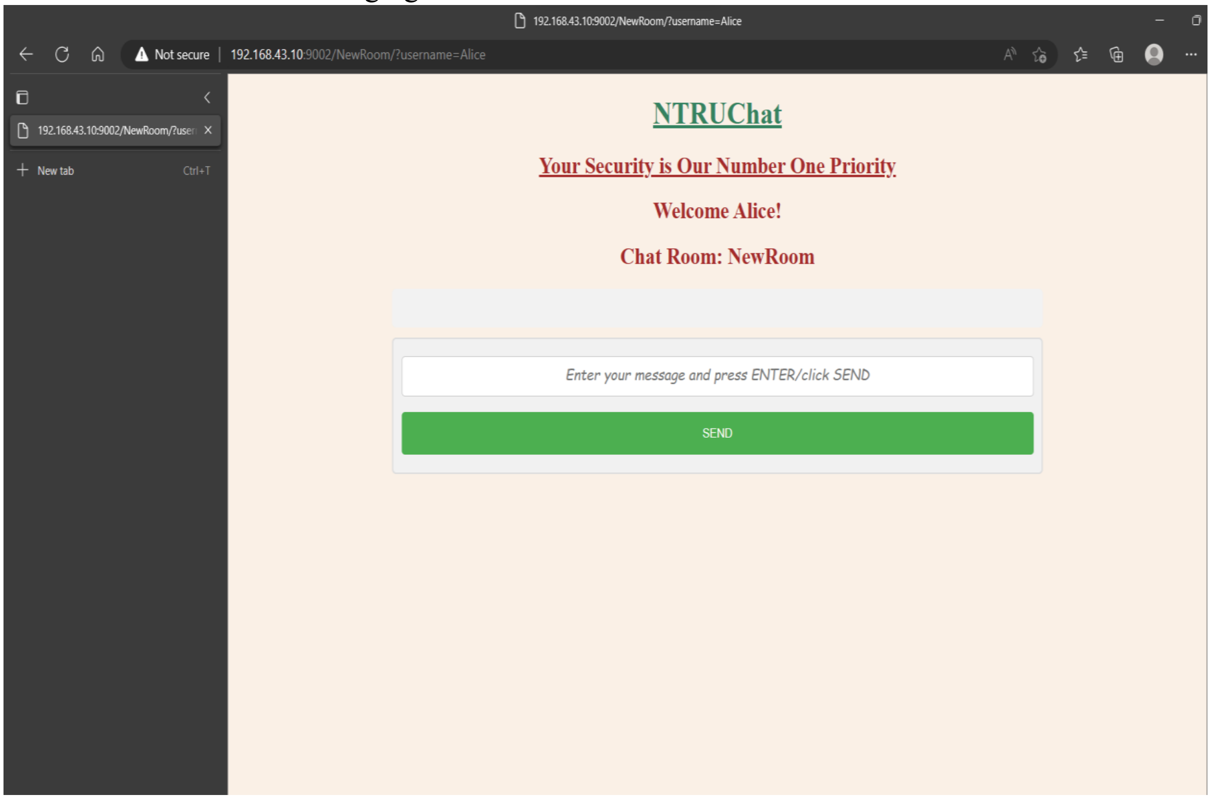
NewRoom

Nickname

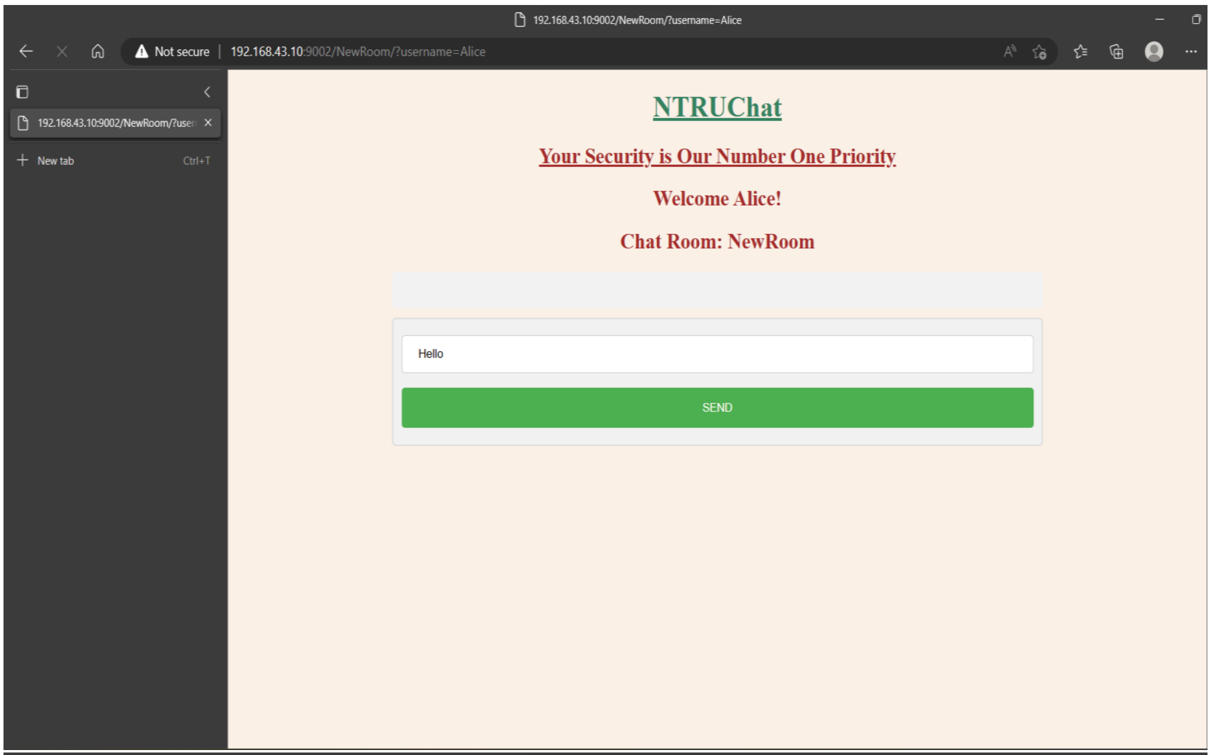
Alice


192.168.43.10:9002
Not secure | 192.168.43.10:9002
NTRUChat
Your Security is Our Number One Priority
Room Name
NewRoom
Nickname
Bob
GO

4. New room created and messaging interface is now available:


192.168.43.10:9002/NewRoom/?username=Alice
Not secure | 192.168.43.10:9002/NewRoom/?username=Alice
192.168.43.10:9002/NewRoom/?user
New tab Ctrl+T
NTRUChat
Your Security is Our Number One Priority
Welcome Alice!
Chat Room: NewRoom
Enter your message and press ENTER/click SEND
SEND

5. Message is written and sent. Each message has a timestamp:
<u>Alice:</u>



192.168.43.10:9002/NewRoom/?username=Alice

NTRUChat

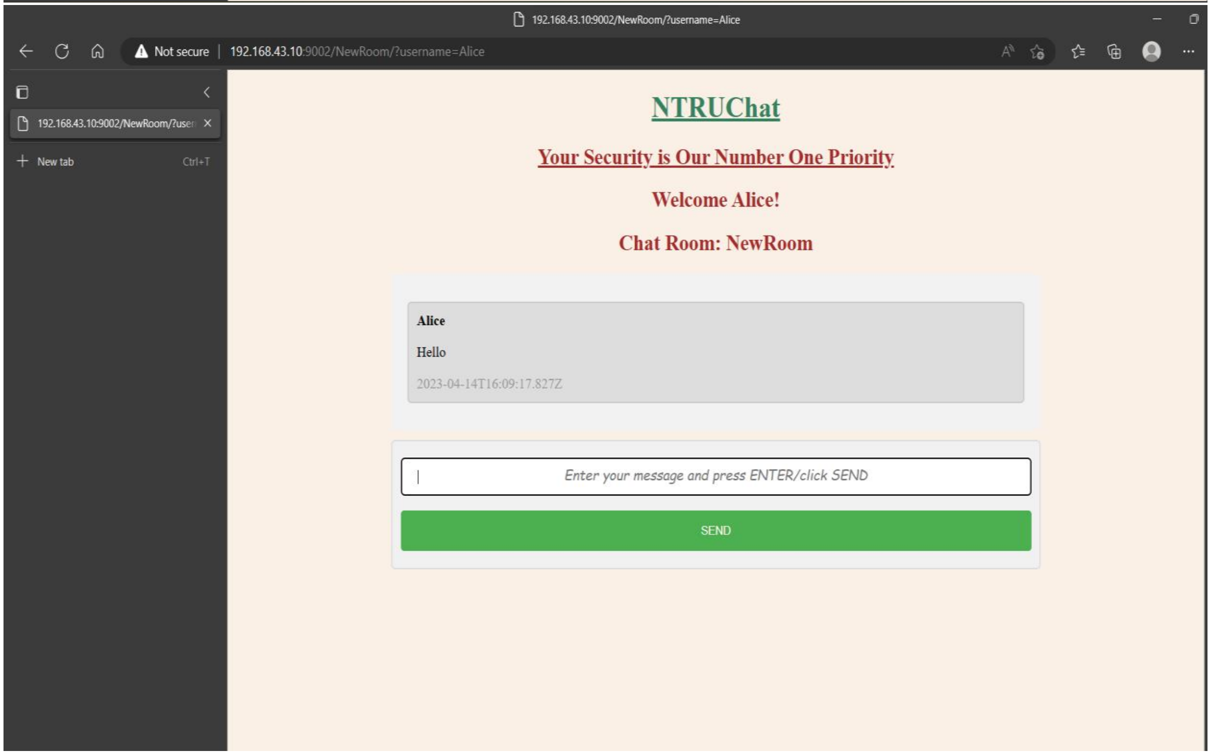
Your Security is Our Number One Priority

Welcome Alice!

Chat Room: NewRoom

Hello

SEND



192.168.43.10:9002/NewRoom/?username=Alice

NTRUChat

Your Security is Our Number One Priority

Welcome Alice!

Chat Room: NewRoom

Alice

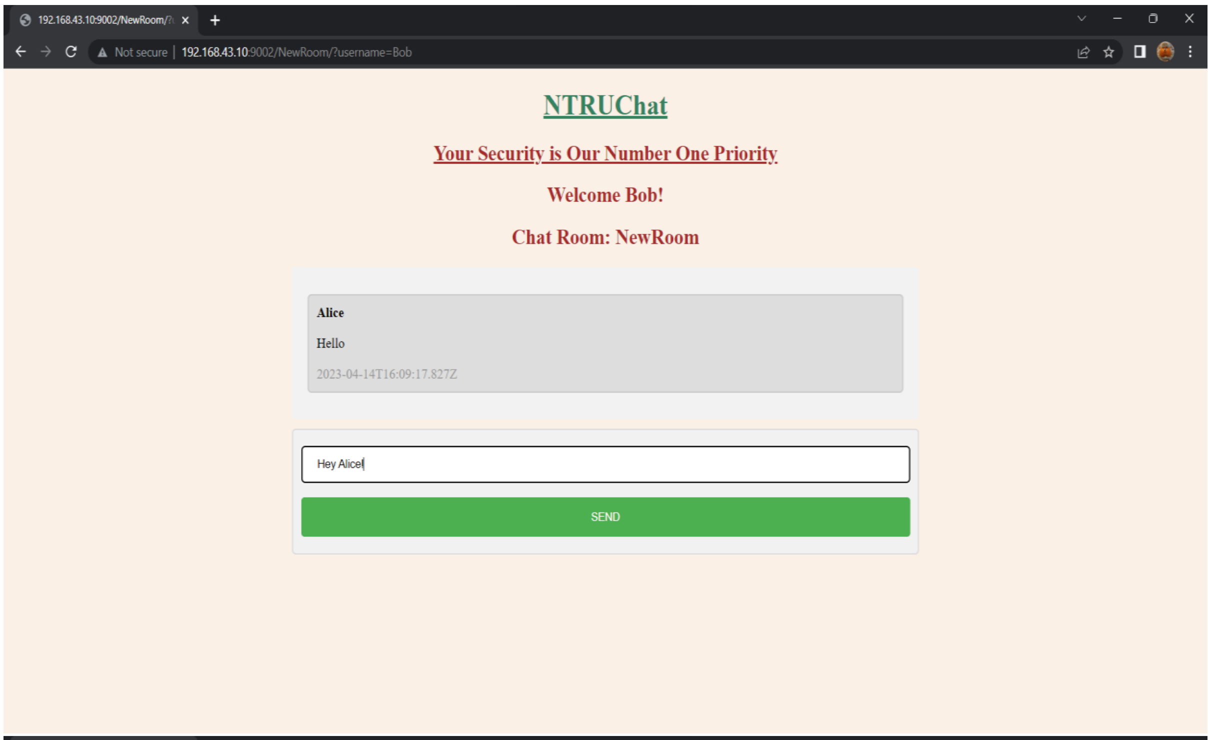
Hello

2023-04-14T16:09:17.827Z

Enter your message and press ENTER/click SEND

SEND

Bob:



192.168.43.10:9002/NewRoom/?username=Bob

NTRUChat

Your Security is Our Number One Priority

Welcome Bob!

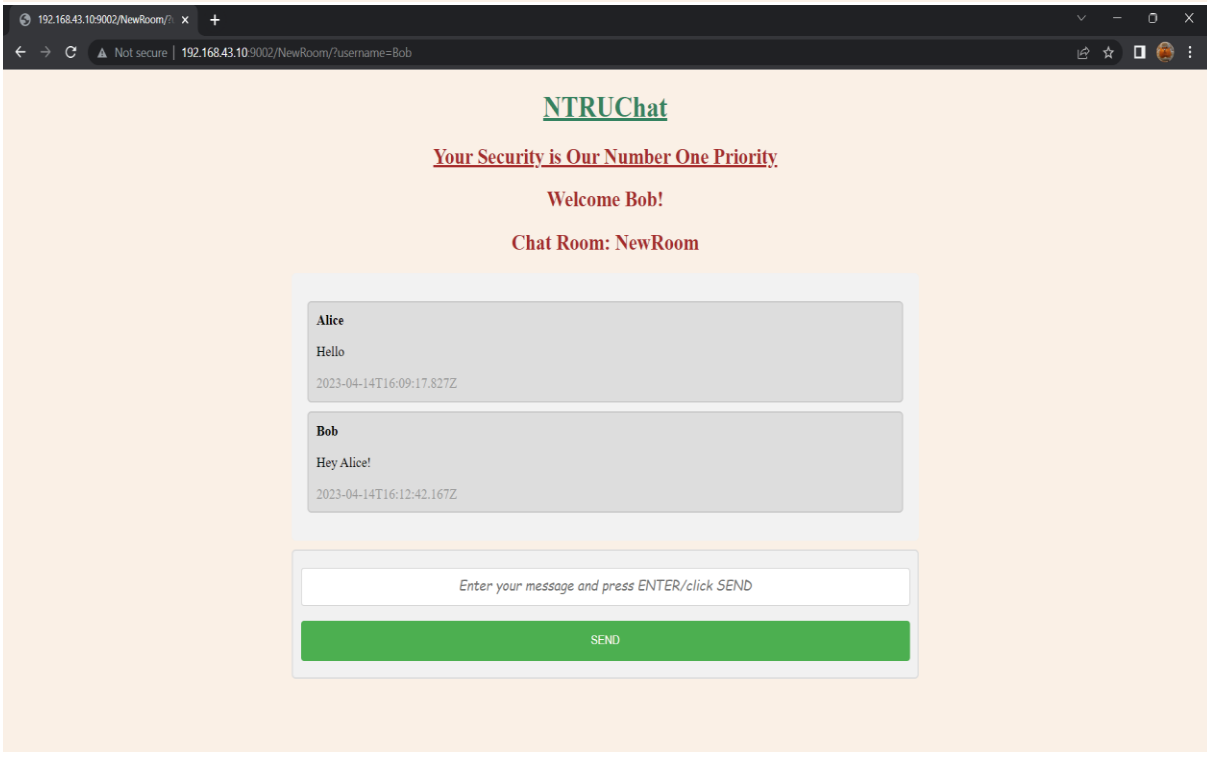
Chat Room: NewRoom

Alice

Hello

2023-04-14T16:09:17.827Z

Hey Alice!



192.168.43.10:9002/NewRoom/?username=Bob

NTRUChat

Your Security is Our Number One Priority

Welcome Bob!

Chat Room: NewRoom

Alice

Hello

2023-04-14T16:09:17.827Z

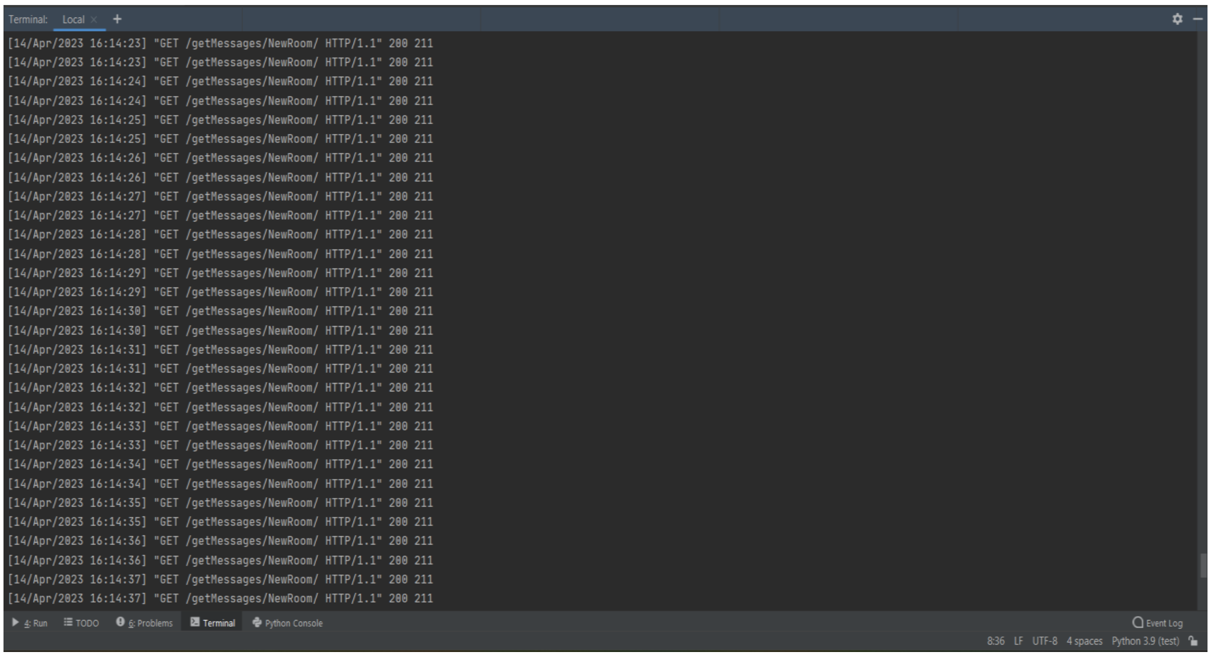
Bob

Hey Alice!

2023-04-14T16:12:42.167Z

Enter your message and press ENTER/click SEND

6. Backend Request log:



```
[14/Apr/2023 16:14:23] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:23] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:24] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:24] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:25] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:25] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:26] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:26] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:27] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:27] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:28] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:28] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:29] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:29] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:30] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:30] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:31] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:31] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:32] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:32] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:33] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:33] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:34] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:34] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:35] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:35] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:36] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:36] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:37] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
[14/Apr/2023 16:14:37] "GET /getMessages/NewRoom/ HTTP/1.1" 200 211
```

Hence, secure web chat application was implemented successfully.

8. Conclusion

The implementation of the NTRU algorithm through both standalone and web application using Django stack was successful and has demonstrated the versatility and potential of this encryption algorithm. The standalone implementation of NTRU algorithm provides a straightforward way to encrypt and decrypt messages on a local machine, while the web application implementation using Django stack allows for easy access and usage from any device with an internet connection. The implementation highlighted the importance of security, user interface design, scalability, and accessibility. These considerations are essential to ensure that the encryption algorithm is effective, user-friendly, and can be used by a wide range of users. Furthermore, the implementation has significant implications for future secure online communication ,i.e. in the era of commercial quantum computers. As the demand for secure and quantum-resistant communication would continue to increase, the NTRU algorithm will provide a promising solution for encrypting and decrypting messages. Overall, the implementation of the NTRU algorithm through both standalone and web application using Django stack has demonstrated the potential of this encryption algorithm for secure and quantum resistant online communication. With further development and optimization in terms of computation speed, the NTRU algorithm has the potential to become a widely used encryption algorithm for secure communication in various domains.

9. References

[1] A Novel Cryptosystem for Files Stored in Cloud using NTRU Encryption Algorithm. (2020). *International Journal of Recent Technology and Engineering Regular Issue, 9*(1), 2127-2130. doi:10.35940/ijrte.a2536.059120

[2] M. E. (2016). Modifying Shor's algorithm to compute short discrete logarithms.

[3] A. P. (2007). Performance Analysis of Public key Cryptographic Systems RSA and NTRU.

[4] Gerjuoy, E. (2005). Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers. *American Journal of Physics*, *73*(6), 521-540. doi:10.1119/1.1891170

[5] Perlner, R. A., & Cooper, D. A. (2009). Quantum resistant public key cryptography. *Proceedings of the 8th Symposium on Identity and Trust on the Internet - IDtrust 09*. doi:10.1145/1527017.1527028

[6] Ekerå, M., & Håstad, J. (2017). Quantum Algorithms for Computing Short Discrete Logarithms and Factoring RSA Integers. *Post-Quantum Cryptography Lecture Notes in Computer Science*, 347-363. doi:10.1007/978-3-319-59879-6_20

[7] Kaur, A., & Singh, N. (2015). SMS Encryption using NTRU Algorithm. *International Journal of Advanced Research in Computer Science & Technology*.

[8] Duits, I. (2019). The Post-Quantum Signal Protocol Secure Chat in a Quantum World.

[9] Gaithuru, J. N., & Bakhtiari, M. (2014). Insight into the operation of NTRU and a comparative study of NTRU, RSA and ECC public key cryptosystems. *2014 8th. Malaysian Software Engineering Conference (MySEC)*. doi:10.1109/mysec.2014.6986028

[10] Hermans, J., Vercauteren, F., & Preneel, B. (2010). Speed Records for NTRU. *Topics in Cryptology - CT-RSA 2010 Lecture Notes in Computer Science,* 73-88. doi:10.1007/978-3-642-11925-5_6

[11] NTRUEncrypt – A Quantum Proof Replacement to RSA Cryptosystem. (2020). *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(5), 7676-7679. doi:10.30534/ijatcse/2020/109952020

[12] Lee, M., Song, J. E., Choi, D., & Han, D. (2010). Countermeasures against Power Analysis Attacks for the NTRU Public Key Cryptosystem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E93-A*(1), 153-163. doi:10.1587/transfun.e93.a.153