Table of Contents

Abstract

NTRU is one of the few public key cryptosystems which is supposedly quantum-computer resistant. Its security is based on the presumed difficulty of a lattice problem, namely, the shortest vector problem. This project describes the NTRU cryptosystem and its cryptanalysis. The project describes how NTRU is a fast implementing algorithm and can be used as an alternative in the current cryptosystem landscape along with RSA or ECC. Finally, a comparison of the performance of the NTRU with other public key cryptosystems RSA and ECC is presented. The differences in performance is calculated along various metrics and their result is presented in tabular and graphical form. The project consists of a full and deep research and analysis on NTRU algorithm.

Introduction

Quantum computers are well known to be able to break cryptosystems that are based on the integer factorization problem or some discrete logarithm problem in polynomial time. This affects RSA and ECC and also number field cryptosystems. The NTRU cryptosystem is an alternative algorithm based on a completely different mathematical problem from RSA and ECC called the closest lattice vector problem. As a result, NTRU is very fast and resistant to quantum computers. NTRU cryptosystem - n^{th} *degree truncated polynomial ring unit*, relies on the presumed difficulty of factoring certain polynomials in a truncated polynomial ring into a quotient of two polynomials having very small coefficients. In comparison to widely known systems such as RSA 2 or ECC 3, the main advantage of NTRU is that its time complexity quadratic order $O(N^2)$ in worst case. his system is considered as the fastest cryptosystem, and its strong points are short key size and the speed of encryption and decryption. Since both encryption and decryption use only simple polynomial multiplication, these operations are very fast compared to other asymmetric encryption schemes, such as RSA, Megamall and elliptic curve cryptography. The security of NTRU is based on intractability of hard problems in certain type of lattices, called convolutional modular lattices, therefore, it is also categorized as a lattice-based cryptosystem.

History:

The NTRUEncrypt Public Key Cryptosystem is a relatively new cryptosystem. The first version of the system, which was simply called NTRU, was developed around 1996 by three mathematicians (Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman). In 1996 these mathematicians together with Daniel Lieman founded the NTRU Cryptosystems, Inc. and were given a patent[1] (now expired) on the cryptosystem. During the last ten years' people have been working on improving the cryptosystem. Since the first presentation of the cryptosystem, some changes were made to improve both the performance of the system and its security. Most performance improvements were focused on speeding up the process. Up till 2005 literature can be found that describes the decryption failures of the NTRUEncrypt. As for security, since the first version of the NTRUEncrypt, new parameters have been introduced that seem secure for all currently known attacks and reasonable increase in computation power. Now the system is fully accepted to IEEE P1363 standards under the specifications for lattice-based public-key cryptography (IEEE P1363.1). Because of the speed of the NTRUEncrypt Public Key Cryptosystem and its low memory use it can be used in applications such as mobile devices and Smart-cards. In April 2011, NTRUEncrypt was accepted as a X9.98 Standard, for use in the financial services industry.

Besides being quantum-computer resistant (at least as of this writing), the encryption/decryption speed is faster than that of other practical cryptosystems; in addition, smaller key sizes make the system attractive. Therefore, any application that requires the 2 aspects of fast performance and/or high-levels of security for the next 10 years would benefit from the NTRU solution. Currently, NTRU satisfies these crucial aspects of security that apply to payment systems, secure messaging and email, mobile e-commerce, vehicle communications (V2V, V2I),

military/aerospace, web browsers and servers, remote backup solutions, online presentations/virtual classrooms, utility meters and cloud providers/datacenters.

LITERATURE SURVEY

- An overview of the NTRU Cryptographic System by Hien Ba Nguyen:
  NTRU is one of the few public key cryptosystems which is supposedly quantum-computer resistant. Its security is based on the presumed difficulty of a lattice problem, namely, the shortest vector problem. This research paper describes the NTRU cryptosystem and its cryptanalysis. Finally, a comparison of the performance of the NTRU with other public key crytosystems such as RSA, Elliptic Curve, and McEliece, is presented.

- Analytical study of implementation issues of NTRU by Praveen Gauravraman:

  Nth -Dimensional Truncated Polynomial Ring (NTRU) is a lattice-based public-key cryptosystem that offers encryption and digital signature solutions. It was designed by Silverman, Hoffstein and Pipher. The NTRU cryptosystem was patented by NTRU Cryptosystems Inc. (which was later acquired by Security Innovations) and available as IEEE 1363.1 and X9.98 standards. NTRU is resistant to attacks based on Quantum computing, to which the standard RSA and ECC public-key cryptosystems are vulnerable to. In addition, NTRU has higher performance advantages over these cryptosystems. In particular, they show some non-trivial issues that should be considered towards a secure and efficient NTRU implementation.

PROPOSED METHOD

➢ NTRU KEYS AND PARAMETERS

N - the polynomials in the ring R have degree N-1. (Non-secret) q - the large modulus to which each coefficient is reduced. (Non- secret) p - the small modulus to which each coefficient is reduced. (Non- secret) f - a polynomial that is the private key.

g - a polynomial that is used to generate the public key h from f (Secret but discarded after initial use) h - the public key, also a polynomial r - the random "blinding" polynomial (Secret but discarded after initial use)

d - coefficient

➢ PUBLIC KEY GENERATION

Sending a secret message from Alice to Bob requires the generation of a public and a private key. The public key is known by both Alice and Bob and the private key is only known by Bob. To generate the key pair two polynomials **f** and **g**, with degree at most N – 1 and with coefficients in {- 1,0,1} are required. They can be considered as representations of the residue classes of polynomials modulo $X^N - 1$ in *R*. The polynomial must satisfy the additional requirement that the inverses modulo *q* and modulo *p* (computed using the Euclidean algorithm) exist, which means that $f.f_p = 1(\text{mod}$

$p)$ and $f.f_q = 1(\text{mop } q)$ must hold. So, when the chosen **f** is not invertible, Bob has to go back and try another **f**.

Both **f** and f_p (and f_g) are Bob's private key. The public key **h** is generated computing the quantity

$$\mathbf{h} = p\mathbf{f}_q \cdot \mathbf{g} \pmod{q}.$$

Example: In this example the parameters (N, p, q) will have the values $N = 11$, $p = 3$ and $q = 32$ and therefore the polynomials **f** and **g** are of degree at most 10. The system parameters (N, p, q) are known to everybody. The polynomials are randomly chosen, so suppose they are represented by

$$\mathbf{f} = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$

$$\mathbf{g} = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Using the Euclidean algorithm, the inverse of **f** modulo p and modulo q, respectively, is computed

$$\mathbf{f}_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9 \pmod{3}$$

$$\mathbf{f}_q = 5 + 9X + 6X^2 + 16X^3 + 4X^4 + 15X^5 + 16X^6 + 22X^7 + 20X^8 + 18X^9 + 30X^{10} \pmod{32}$$

Which creates the public key **h** (known to both Alice and Bob) computing the product

$$\mathbf{h} = p\mathbf{f}_q \cdot \mathbf{g} \pmod{32} = 8 - 7X - 10X^2 - 12X^3 + 12X^4 - 8X^5 + 15X^6 - 13X^7 + 12X^8 - 13X^9 + 16X^{10} \pmod{32}$$

➢ ENCRYPTION

Alice, who wants to send a secret message to Bob, puts her message in the form of a polynomial **m** with coefficients {-1,0,1}. In modern applications of the encryption, the message polynomial can be translated in a binary or ternary representation. After creating the message polynomial, Alice chooses randomly a polynomial **r** with small coefficients (not restricted to the set {- 1,0,1}), that is meant to obscure the message.

With Bob's public key **h** the encrypted message **e** is computed:

$$\mathbf{e} = \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \pmod{q}$$

This ciphertext hides Alice's messages and can be sent safely to Bob.

Example: Assume that Alice wants to send a message that can be written as polynomial

$$\mathbf{m} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

and that the randomly chosen 'blinding value' can be expressed as

$$\mathbf{r} = -1 + X^2 + X^3 + X^4 - X^5 - X^7$$

The ciphertext **e** that represents her encrypted message to Bob will look like

$$\mathbf{e} = \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \pmod{32} = 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10} \pmod{32}$$

➢ DECRYPTION

Anybody knowing **r** could compute the message **m**; so, **r** must not be revealed by Alice. In addition to the publicly available information, Bob knows his own private key. Here is how he can obtain **m**: First he multiplies the encrypted message **e** and part of his private key **f**

$$\mathbf{a} = \mathbf{f} \cdot \mathbf{e} \pmod{q}$$

By rewriting the polynomials, this equation is actually representing the following computation:

$$\mathbf{a} = \mathbf{f} \cdot \mathbf{e} \pmod{q}$$

$$\mathbf{a} = \mathbf{f} \cdot (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \pmod{q}$$

$$\mathbf{a} = \mathbf{f} \cdot (\mathbf{r} \cdot p\mathbf{f}_q \cdot \mathbf{g} + \mathbf{m}) \pmod{q}$$

$$\mathbf{a} = p\mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{m} \pmod{q}$$

Instead of choosing the coefficients of **a** between 0 and $q - 1$ they are chosen in the interval $[-q/2, q/2]$ to prevent that the original message may not be properly recovered since Alice chooses the coordinates of her message **m** in the interval $[-p/2, p/2]$. This implies that 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 that the original message may be recovered properly.

The next step will be to calculate **a** modulo p:

$$\mathbf{b} = \mathbf{a} \pmod{p} = \mathbf{f} \cdot \mathbf{m} \pmod{p}$$

because pr.g (mod p) = 0

Knowing **b** Bob can use the other part of his private key (f_p) to recover Alice's message by multiplication of **b** and f_p**.**

$$\mathbf{c} = \mathbf{f}_p \cdot \mathbf{b} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \pmod{p}$$

$$\mathbf{c} = \mathbf{m} \pmod{p}$$

because the property $f.f_p = 1 \pmod p$ was required for f_p.

Example: The encrypted message **e** from Alice to Bob is multiplied with polynomial **f**

$$\mathbf{a} = \mathbf{f} \cdot \mathbf{e} \pmod{32} = 3 - 7X - 10X^2 - 11X^3 + 10X^4 + 7X^5 + 6X^6 + 7X^7 + 5X^8 - 3X^9 - 7X^{10} \pmod{32},$$

where Bob uses the interval [-*q*/2, *q*/2] instead of the interval [0, *q* – 1] for the coefficients of polynomial **a** to prevent that the original message may not be recovered correctly.

Reducing the coefficients of **a** mod *p* results in

$$\mathbf{b} = \mathbf{a} \pmod{3} = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \pmod{3}$$

which equals b = f.m (mod 3).

In the last step the result is multiplied with f_p from Bob's private key to end up with the original message **m**

$$\mathbf{c} = \mathbf{f}_p \cdot \mathbf{b} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \pmod{3} = \mathbf{m} \pmod{3}$$

$$\mathbf{c} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

Which indeed is the original message Alice has sent to Bob!

➢ Why the Decryption Works?

If the NTRU parameters (N, p, q, d – public parameters) are chosen to satisfy

$$q > (6d + 1)\, p$$

this inequality ensures that decryption never fails.

Example: (N, p, q, d) = (7, 3, 41, 2).

According to the example: 41 = q > (6d + 1)p = 39

COMPARISON OF NTRU WITH OTHER PUBLIC KEY CRYPTOSYSTEMS

Public-key cryptography uses two separate keys for encryption and decryption. Even though it is not the same key, the two parts of this key pair are mathematically linked. The public key is used to encrypt plaintext or to verify a digital signature which can be announced to the public. The private key used to decrypt ciphertext or to create a digital signature must be kept secret. Public key algorithm are usually based on the computational complexity of hard problems. For example, the RSA cryptosystem is based on the difficulty of the integer factorization problem, Elliptic curve cryptography (ECC) is based on the difficulty of number theoretic problems involving elliptic curves.

RSA CRYPTOSYSTEM

RSA is one of the first public key cryptosystem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman and was proposed in 1977. The three main steps of RSA algorithm are key generation, encryption and decryption.

Public key cryptosystems are mainly used to exchange either a secret key or a short message. RSA and NTRU work in units of message blocks, and in both system, a single message block is large enough to maintain a short message or a secret key at high security. Therefore, in comparison between these two systems, we will evaluate the length of time to encrypt and decrypt a single message block.

Table shows the security and timing comparisons of NTRU versus the RSA cryptosystem. Comparisons also include the time for a cryptosystem to encrypt, decrypt and create a key. Key generation is the process of generating keys for cryptography. A key is used to encode and decode any data being encoded/decoded.

Comparison of NTRU cryptosystem vs. the RSA cryptosystem

System	Security(yrs)	Key Size(bits)	CreateKey(msecs)	Encrypt(blks/sec)	Decrypt(blks.sec)
RSA 512	$4.00*10^5$	512	260	2441	122
NTRU 167	$2.08*10^6$	1169	4	5941	2818
RSA 1024	$3.00*10^{12}$	1024	1280	932	22
NTRU 263	$4.61*10^{14}$	1841	7.5	3676	1619
RSA 2048	$3.00*10^{21}$	2048	4195	310	3
NTRU 503	$3.38*10^{35}$	4024	17.3	1471	608

- "NTRU n" refers to an implementation of the NTRU public key cryptosystem using the domain parameters n, that is, using polynomials of degree n − 1 in the ring Z[x]/(Xn − 1).

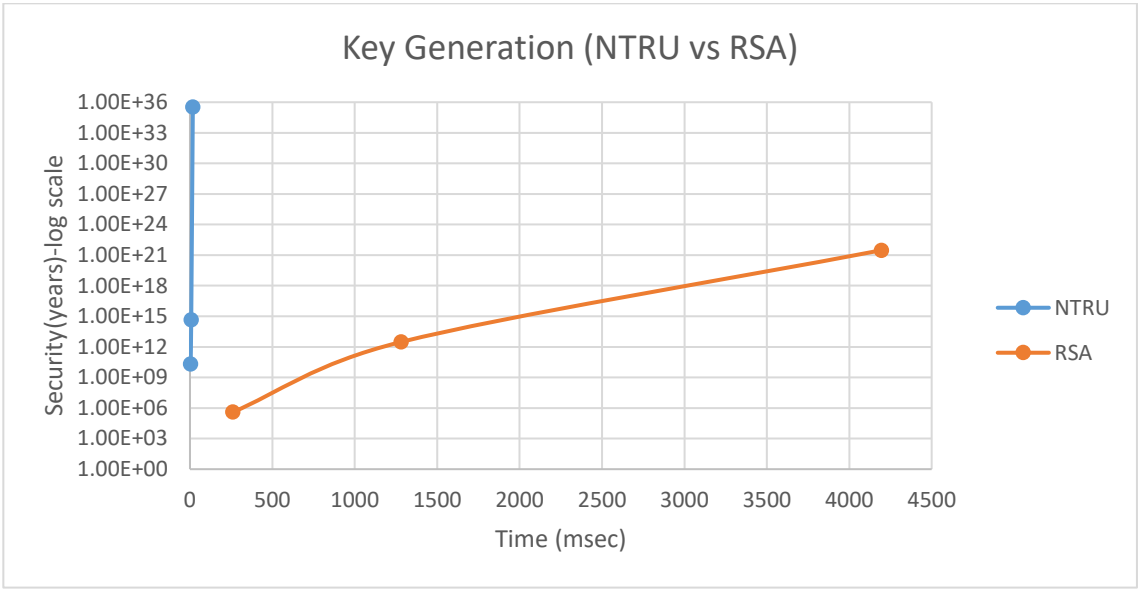
-Security is measured in MIPS-years required to break the system. Note that all security times are estimates, based on extrapolation of the breaking time for lower security levels.

-Key size refers to public key in bits. NTRU private key are shorter than their public keys. RSA public and private keys are the same size.

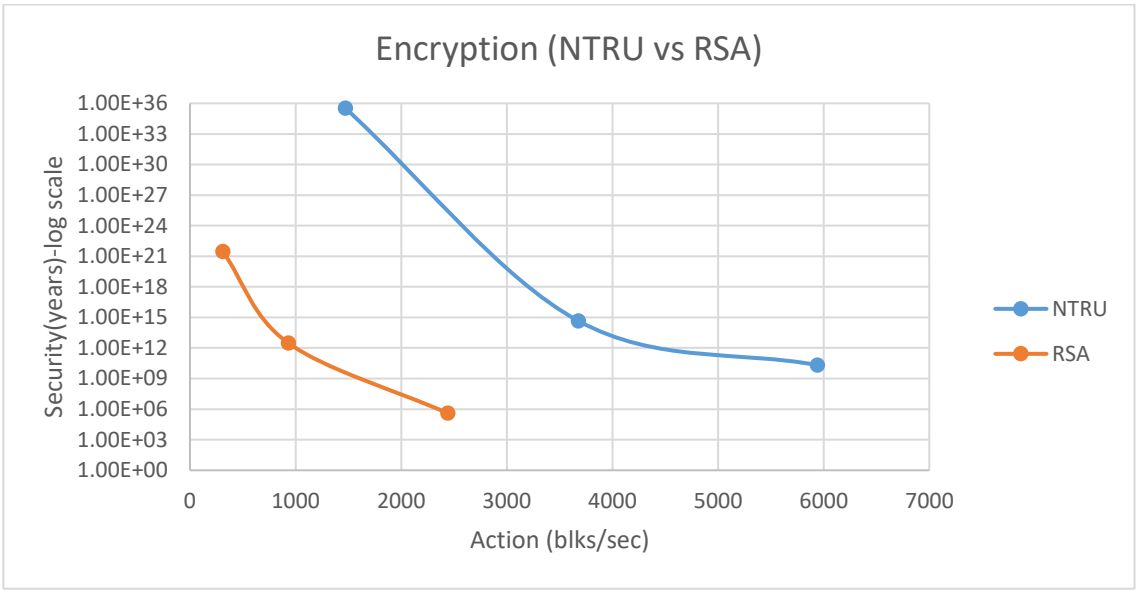
-NTRU encryption, decryption and key creation performed using Tao Group's Tumbler 24 implementation of the NTRU algorithm programmed in C and running on a 300 MHz P entium II operating under Linux.

-RSA key creation done on a 255 MHz Digital Alpha Station.

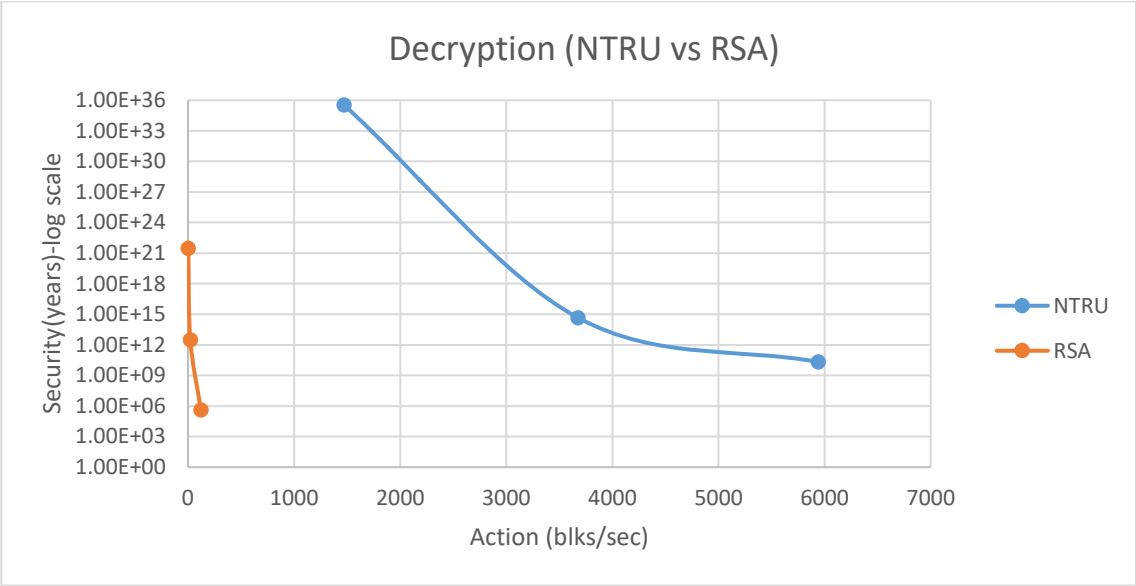
-RSA encryption/decryption programmed in Microsoft Visual C++ 5.0 (optimized for speed, Pentium Pro code generation), and run on a P entium II 266 MHz machine under Windows NT 4.0. RSA encryption uses exponent 17 to increase speed



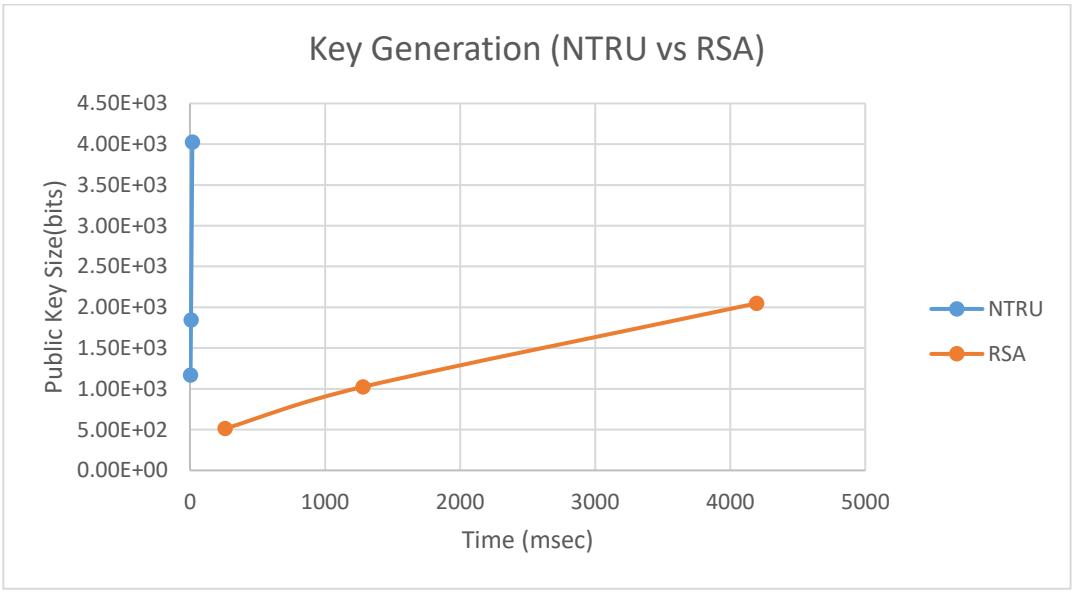
Key Generation Graph of NTRU vs. RSA



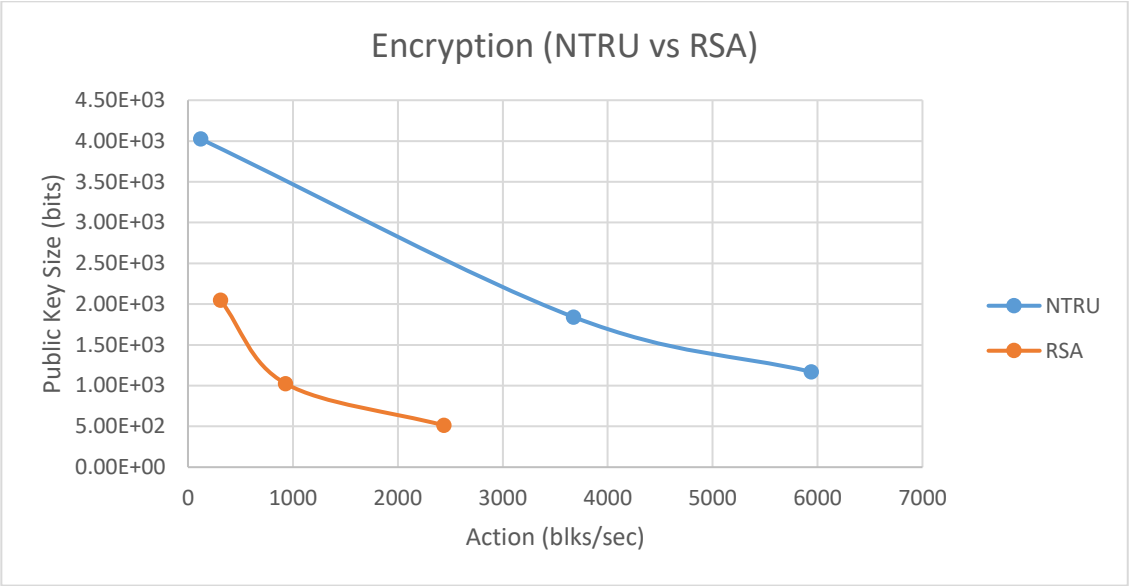
Encryption Graph of NTRU vs. RSA



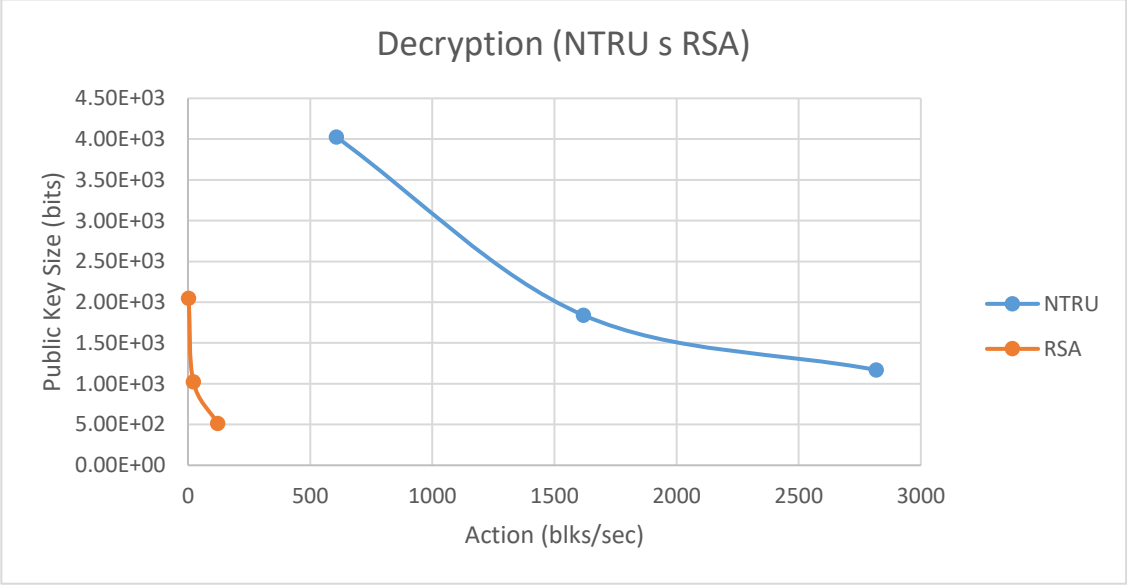
Decryption Graph of NTRU vs. RSA



Key Size vs. Key Generation Graph of NTRU vs. RSA



Key Size vs. Encryption Graph of NTRU vs. RSA



Key Size vs. Decryption Graph of NTRU vs. RSA

Clearly, we can conclude that the performance of NTRU is better than RSA in terms of key generation, encryption and decryption. We can also see that the key generation of NTRU is faster than in RSA. Furthermore, NTRU encrypts more messages than RSA with the same key sizes. Finally, the NTRU also decrypts more messages than RSA with the same key sizes.

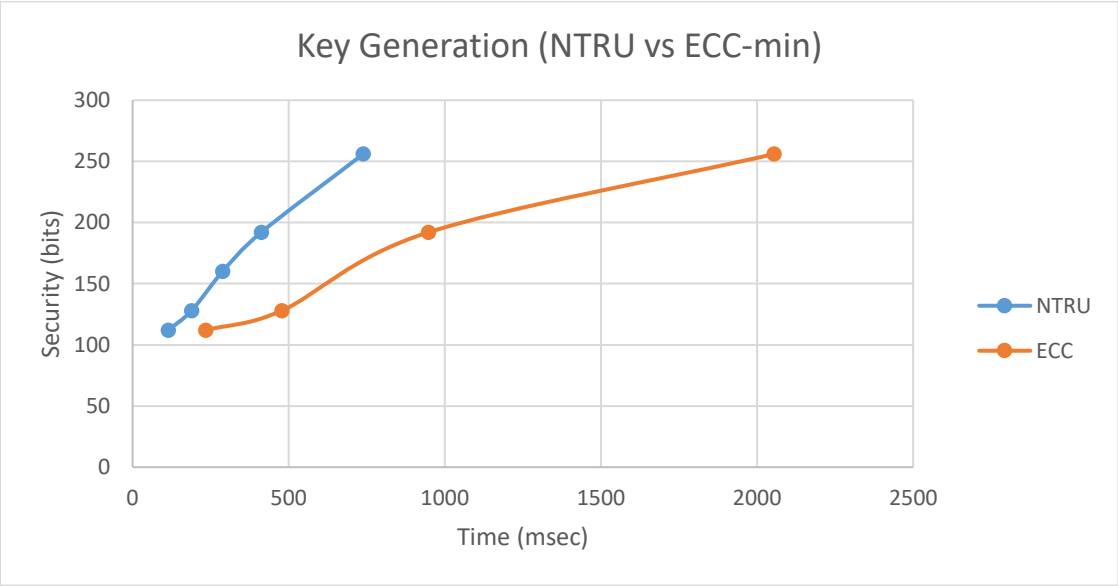
ECC CRYPTOSYSTEM

Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. They are also used in several integer factorization algorithms based on elliptic curves that have applications in cryptography, such as Lenstra elliptic-curve factorization.

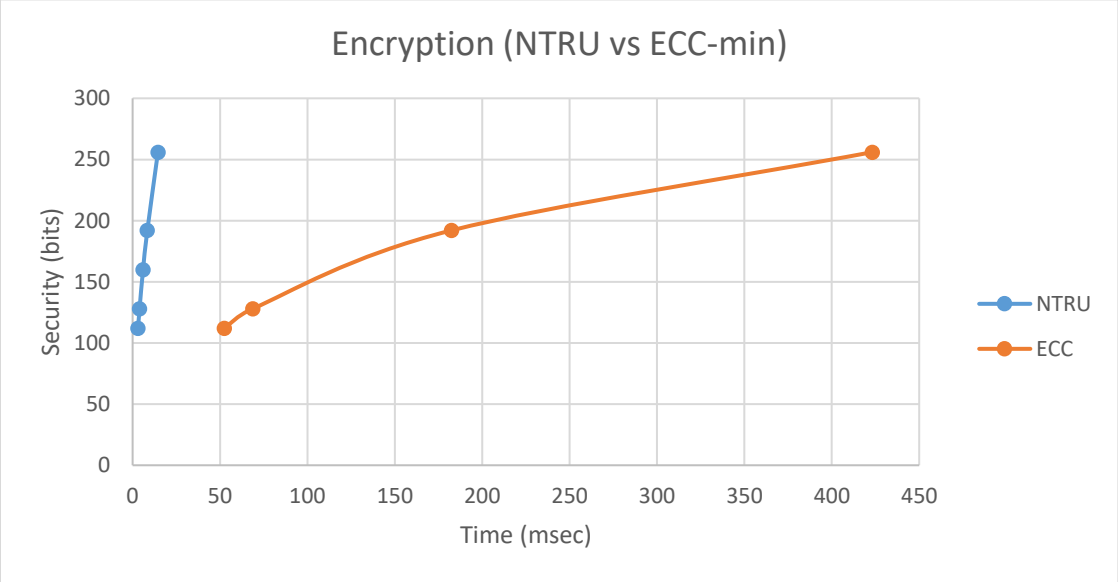
Table shows us the time requirement for key generation, encryption and decryption of NTRU and ECC cryptosystem where the code was compiled and run on a personal computer of Windows XP Professional OS, P4 2.80 GHz CPU and 1GB RAM.

Comparison of NTRU cryptosystem vs. the ECC cryptosystem

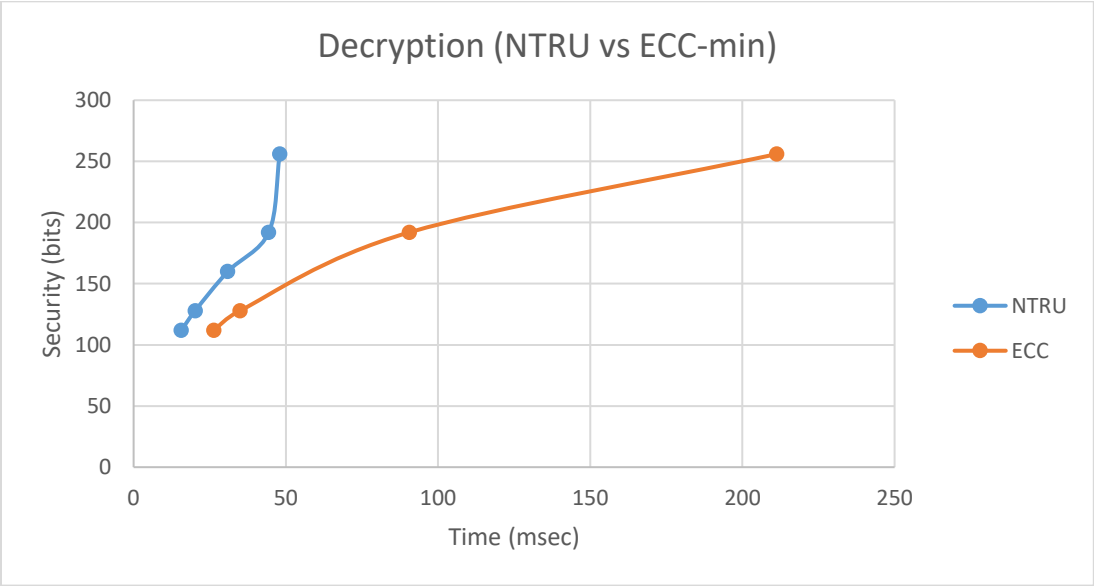
Cryptosystems	Security(bits)	Key Generation(msecs)	Encrypt(msec)	Decrypt(msec)
NTRU 347	112	114.16	3.11	15.70
ECC 224	112	234.11-367.98	52.52-164.50	26.35-81.52
NTRU 397	128	188.92	3.97	20.26
ECC 256	128	478.22-656.63	68.72-223.29	35.00-111.16
NTRU 491	160	288.31	5.97	30.96
NTRU 587	192	412.10	8.42	44.42
ECC 384	192	946.43-1429.11	182.35-586.20	90.61-290.94
NTRU 787	256	738.75	14.49	48
ECC 521	256	2055.04-3175.87	432.25-1257.56	211.35-626.33



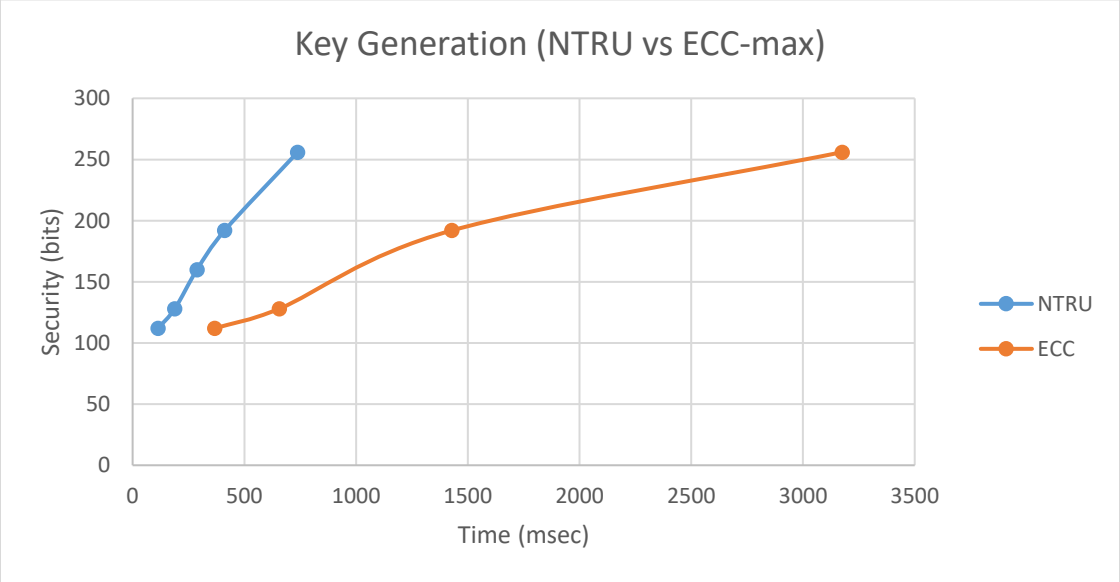
Key Generation Graph of NTRU vs. ECC-min



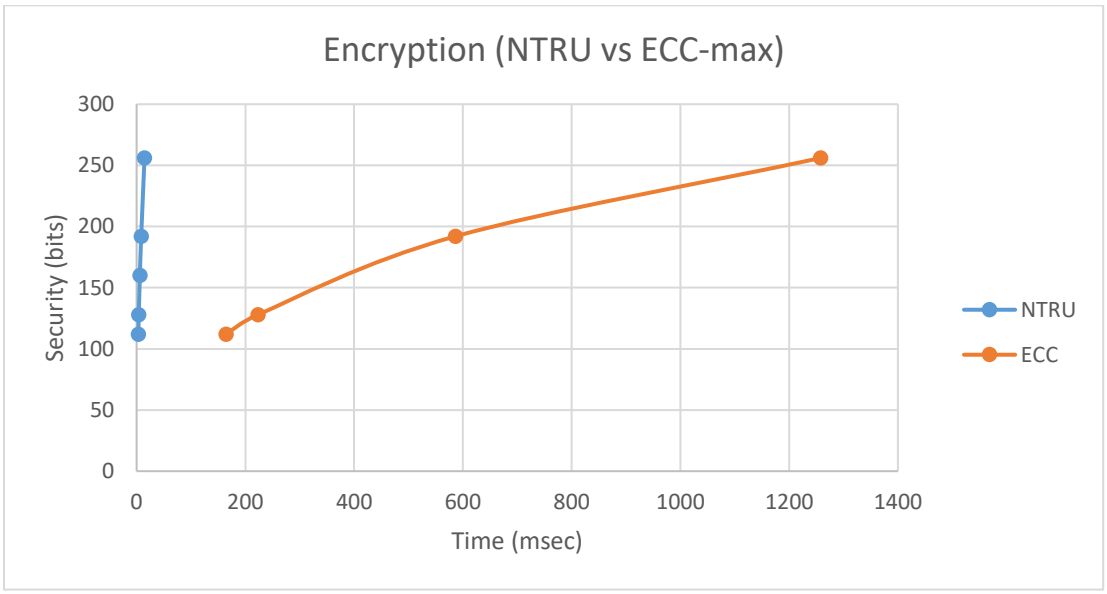
Encryption Graph of NTRU vs. ECC-min



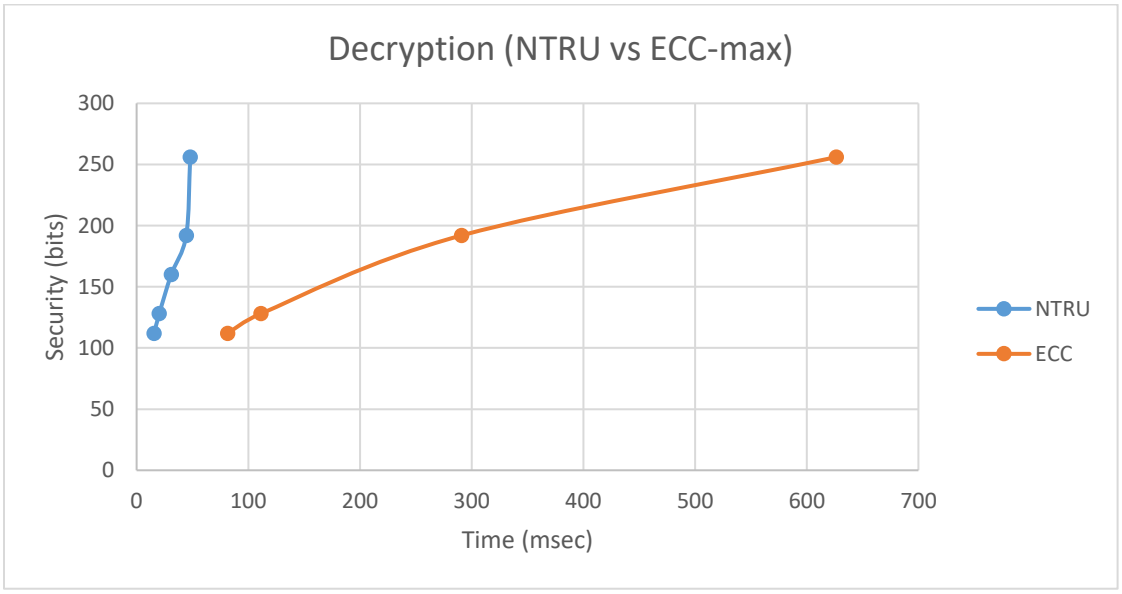
Decryption Graph of NTRU vs. ECC-min



Key Generation Graph of NTRU vs. ECC-max



. Encryption Graph of NTRU vs. ECC-max



Decryption Graph of NTRU vs. ECC-max

From Table above, we discovered that the NTRU is much faster than the ECC with all levels of security. From the figures we also concluded that the performance of NTRU is superior in comparison to the performance of minimum values of ECC timings. Similarly, we also concluded that the performance of NTRU is superior in comparison to the performance of maximum values of ECC timings.

Conclusion

NTRU is very fast and secure compared to other public key cryptosystems such as RSA and ECC. The first part of this project described the NTRU cryptosystem parameters, private key, public key, encryption and decryption. Then we demonstrated how to implement a program in python which is capable of performing NTRU encryption, decryption as well as the key generation processes. Following this, in the project we compared NTRU with RSA and ECC. In this comparison, it was concluded that NTRU was better than RSA and ECC along several parameters like key generation, encryption and decryption with several different constraints like time taken in years(to crack the key) or according to the public key size. To help visualize these data points we implemented scatter point graphs for all different test scenarios. Thus, through all the analysis we concluded that NTRU has a very high potential encryption and has wide applications in cryptography. Add to this, the fact that NTRU has proven to be quantum resistant till now (which to iterate, both RSA and ECC can be broken quite easily using Shor's algorithm in a quantum computer), and will be for quite the foreseeable future its appeal is only increasing day by day.

References

- https://www.wikiwand.com/en/NTRUEncrypt
- https://sdsudspace.calstate.edu/bitstream/handle/10211.3/135700/Nguyen_sdsu_0220N_10605.pdf?sequence
- http://people.scs.carleton.ca/~maheshwa/courses/4109/Seminar11/NTRU _presentation.pdf
- https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1693