DLSU Research Congress 2025
De La Salle University, Manila, Philippines
June 26 to 28, 2025

PhilSys ID Data Authentication via NTRUEncrypt: A Lattice-Based and Post-Quantum Cryptographic Implementation

John Trixie M. Ocampo[1,*] and Romie C. Mabborang[2]
[1] *Department of Mathematics, Pamantasan ng Lungsod ng Maynila*
[2] *Chairman, Department of Mathematics, Pamantasan ng Lungsod ng Maynila*
**Corresponding Author: jtmocampo2021@plm.edu.ph*

Abstract: With the advent of quantum computing, classical cryptographic algorithms such as RSA and ECC are at risk with various data and sensitive information vulnerable to unwanted attacks. With that, the Philippine National Identification System (PhilSys), a digital identity system for Filipinos, needs a quantum-resistant encryption algorithm that can withstand future quantum attacks. NTRUEncrypt, a lattice-based algorithm developed by Hoffstein et. al., was chosen for its strong resistance to quantum attacks, efficient key generation, and scalable encryption capabilities. This study integrates NTRUEncrypt into PhilSys via a web-based system developed using Python to demonstrate secure encryption, storage, and decryption of sensitive data. The research employed a descriptive-experimental approach, testing the algorithm across three security levels—moderate, high, and highest, each with varying parameters. Key metrics such as encryption speed, decryption accuracy, and resistance to brute-force attacks were analyzed. The results showed 100% decryption accuracy, with encryption and decryption times scaling efficiently with increasing security levels. NTRUEncrypt outperformed classical cryptographic methods and proved competitive with other post-quantum algorithms. This study confirms that NTRUEncrypt can significantly enhance PhilSys security, providing a foundation for quantum-resistant digital identity systems. Future work could expand its application to biometric data and nationwide deployment.

Key Words: asymmetric cryptography; encryption; decryption; quantum computing; web-based system

1. INTRODUCTION

1.1 Background of the Study

As digital identity systems grow, so do concerns about data security and integrity. Governments worldwide are adopting unified ID systems like the Philippine Identification System (PhilSys) to streamline services across agencies (Aceron, 2021). However, as identity information becomes increasingly digitalized, it faces heightened risks—especially with the rise of quantum computing, which threatens traditional cryptographic algorithms such as RSA and ECC. These classical systems rely on mathematical problems that quantum computers could potentially solve with ease, making them vulnerable to future attacks.

To counter this, Post-Quantum Cryptography (PQC) offers a promising defense. Among the leading PQC methods is NTRUEncrypt, a lattice-based

encryption system known for its speed, small key size, and resistance to quantum attacks. Unlike RSA or ECC, which rely on factorization or logarithmic problems, NTRUEncrypt is built on hard lattice problems, which remain secure even in the quantum era (Bernstein & Lange, 2017). Integrating NTRUEncrypt into PhilSys not only strengthens data protection but also ensures that only authorized users can access sensitive personal information. This approach provides a future-proof foundation for national ID systems amid evolving cybersecurity threats.

The study aims to investigate the integration of NTRUEncrypt, a post-quantum cryptographic algorithm, into the authentication framework of the Philippine National Identification System (PhilSys). Performance metrics include encryption/decryption speed, data integrity, computational efficiency, and resistance to quantum attack simulations. The overarching problem centers on how the current system can be made resistant to quantum-based attacks while maintaining efficiency and scalability (Chen et. al., 2016).

1.2 Research Paradigm

The following flowchart will serve as the conceptual framework of the NTRU-Authenticated Philippine Identification System (NAPhilSys). A website for the NAPhilSys will serve as the front-end of the system and all data input will be stored in the NAPhilSys database.

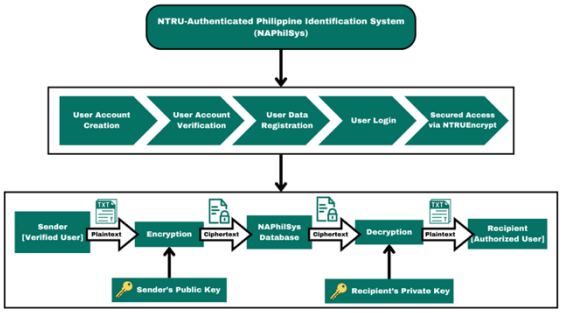

Figure 1: NAPhilSys Conceptual Framework

2. METHODOLOGY

2.1 Implementation of the NTRUEncrypt Algorithm

2.1.1 Generation of NTRU Key Pair

To generate NTRU Keys, the researcher begins by choosing integer values for N, p, and q, with $N-1$ serving as the degree of the polynomials and will dictate the security and performance of the algorithm, together with $\mathrm{q} \geq \mathrm{p}$ and coprimes p and q, that is $\gcd(\mathrm{p}, \mathrm{q}) = 1$. These parameters vary per security level, but take $N = 107$, $p = 3$, and $q = 64$ which are parameters under "moderate" security. This would mean that the polynomials will have degrees ≤ 106, and p and q satisfies the requirement as $gcd(3,64) = 1$. The researcher then chooses two polynomials f and g from the ring $R = Z[X]/(X^N - 1)$ with f being invertible modulo p and q, and g being randomly chosen. For this, a small polynomial f and another small random polynomial g is then chosen with coefficients $-1,0,1$, randomly distributed across all terms of a polynomial with degree ≤ 106, for this we take $f(x) = 1 - x + x^3 - x^5 + x^{10}$ and $g(x) = 1 + x^2 - x^4 + x^8 - x^{12}$. Next, polynomials f_p and f_q, which are polynomial inverses of f modulo p and q, respectively. Continuing with the previous example, the inverses of f and g, that is $f_p(x) = f(x)^{-1} \bmod p$ across $Z_3[X]$ would be $f_p(x) = 1 + x + 2x^3 + 2x^5 + x^{10}$ and that $f_q(x) = f(x)^{-1} \bmod q$ across $Z_{64}[X]$ would be $f_q(x) = 1 - x + x^2 - x^4 + x^6$. The public key can now be computed as $h(x) = p\, f_{q(x)} \circledast g(x) \mod q$ ($\circledast$ denotes Polynomial Convolution), while the private keys are simply f and f_p. Plugging in the earlier values, we get $h(x) = 3(1 - x + x^2 - x^4 + x^6) \circledast (1 + x^2 - x^4 + x^8 - x^{12}) \bmod 64$, resulting to the public key $h(x) = 3 + 3x^2 - 3x^4 + 3x^6 - 3x^8 \bmod 64$. With that, we now have the public key $h(x) = 3 + 3x^2 - 3x^4 + 3x^6 - 3x^8$, the private keys $f(x) = 1 - x + x^3 - x^5 + x^{10}$ and $f_p(x) = 1 + x + 2x^3 + 2x^5 + x^{10}$.

2.1.2 Encryption Algorithm

Before the researcher starts encrypting, plaintext is first converted into a message polynomial

$m(x)$ via the American Standard Code for Information Interchange (ASCII) value. Each character is converted into its ASCII value, and stored into a polynomial as its coefficients. Take the plaintext "Hello Trixie", this can be converted into its message polynomial with its ASCII values as coefficient, resulting to the polynomial $m(x) = 72 + 101x + 108x^2 + 108x^3 + 111x^4 + 32x^5 + 84x^6 + 114x^7 + 105x^8 + 120x^9 + 105x^{10} + 105x^{11}$. This polynomial can now be reduced by reducing the coefficients modulo p. With the parameter earlier of $p = 3$ we get the reduced polynomial message $m(x) = 2x + 2x^5 + 2x^{11}$ which can now be used for encryption and decryption. To encrypt, the researcher generates a random polynomial $r(x)$ with small coefficients, take $r(x) = 1 - x + x^3 - x^5$. This can now be used together with the public key $h(x)$ and the message polynomial $m(x)$ to compute the ciphertext using the formula $e(x) = r(x) \circledast h(x) + m(x) \mod q$. We first convolve $r(x)$ and $h(x)$, that is $r(x) \circledast h(x) = (1 - x + x^3 - x^5) \circledast (3 + 3x^2 - 3x^4 + 3x^6 - 3x^8)$ then reduce the product modulo $q = 64$. This results to $r(x) \circledast h(x) = 3 + 3x - 3x^2 + 3x^3 + 3x^5 \mod 64$. We then add the message polynomial $m(x)$, resulting to the ciphertext $e(x) = 3 + 5x - 3x^2 + 3x^3 + 5x^5 + 2x^{11}$.

2.1.3 Decryption Algorithm

For decryption, we now set $a(x) = f(x) \circledast e(x) \mod q$, that is $a(x) = (1 - x + x^3 - x^5 + x^{10}) \circledast (3 + 5x - 3x^2 + 3x^3 + 5x^5 + 2x^{11}) \mod 64$. This results to $a(x) = 6 + 2x + 3x^3 + 4x^5 + 2x^{11}$. After this, we get $b(x) = a(x) \mod p$ with $p = 3$, which results to $b(x) = 2x + x^5 + 2x^{11}$. From this, we can now recover the original message using the formula $m(x) = f_p(x) \circledast b(x) \mod p$. Convolving $f_p(x)$ and $b(x)$ results to $f_p(x) \circledast b(x) = (1 + x + 2x^3 + 2x^5 + x^{10}) \circledast (2x + x^5 + 2x^{11})$ which is equal to $m(x) = 2x + 2x^5 + 2x^{11}$, the original ciphertext.

2.2 Respondents of the Study

The study requires a database of each individual's personal information to test NTRUEncrypt's efficiency to encrypt user data. To formulate the test database, the researcher defined the respondents to be those who already registered for their Philippine National ID (PhilSys).

Given the nature of the number of registered individuals among the students, a voluntary manner of data collection among the population was implemented. This resulted to a total of 24 respondents. All respondents will be divided into three, eight for each security level and parameter.

2.3 Data Gathering Instruments

Given that a web-based application was built to cater for the user's information input, the application itself can be used as the primary data gathering instrument. The formulated NTRUEncrypt code via Python will directly serve to encrypt user data before submitting to the server database.

Consequently, a survey questionnaire via Google Forms was used as the primary data collection tool for the database. Link for this questionnaire was sent to those who qualify as respondents—individuals who already have their PhilSys ID for them to accomplish the forms, voluntarily. Those who answered said forms are subject to privacy and confidentiality by also attaching consent forms prior to the surveying proper for proper handling of the respondents' information. The following information were collected from the respondents: Name; PhilSys Card Number (PCN); Birthdate; Email Address; Residential Address; Sex; Blood Type; Marital Status; Place of Birth.

Using the above-mentioned information and the specified research instruments, the researcher was able to complete the mentioned NTRUEncrypt-Authenticated PhilSys ID framework and database.

2.4 Data Gathering Procedure

Starting with the creation of the web-based application itself, the researcher utilized an open-source code (ProtDos, 2023) for the skeletal Python code of the automated NTRUEncrypt encryption and decryption. After several modifications to be used for the web-based application, using Flask and its sub-library Flask-mysqldb, Python entirely made up the back-end framework of the application and allowed this application to connect to a MySQL database via

the MySQL Workbench. The HTML and CSS codes for the website's skeleton and aesthetics were then done including the website's login, register, home, profile, edit, and encrypted profile pages. The parameters were then modified in the private and public key files accessed by the NTRU python code to align with the "moderate" security parameters for NTRU according to (Hoffstein et. al., 1999).

After testing the website and confirming its reliability, the researcher disseminated the survey questionnaire via Google Forms where the background of the study and the purpose of data collection was thoroughly explained to the selected respondents.

3. RESULTS AND DISCUSSION

The system analysis for the NTRUEncrypt implementation was evaluated across 24 respondents, divided into three security levels: moderate, high, and highest with the following parameters (Hoffstein et. al., 1999):

Table 1. Parameters per Security Level

	Moderate	High	Highest
N	107	167	503
p	3	3	3
q	64	128	256

All 24 respondents' data will be divided into each of the three security levels corresponding to 8 respondent data per level. Each respondent will be given a respondent ID. Respondents R1 to R8 will be encrypted with "moderate" security level, R9 to R16 will be for "high" security level, and R17 to R24 will be for "highest" security level. Analysis will be done on these according to accuracy, performance, security, and benchmark comparison.

3.1 Accuracy Analysis

To analyze the accuracy for each security level, the percentage accuracy metric was employed to calculate the percentage of correctly decrypted fields.

Table 2. Percentage Accuracy per Security Level

	Fields Tested	Correctly Decrypted Fields	Average Percentage Accuracy
Moderate	9	9	100%
High	9	9	100%
Highest	9	9	100%

Table 2 shows the percentage accuracy per security level. Each row shows the average percentage accuracy across all 8 respondents per security level. Each field showed 100% decryption accuracy, which includes alphanumeric characters and several symbols.

3.2 Performance Analysis

To measure performance, the system is subjected to computational efficiency compared to its time complexity.

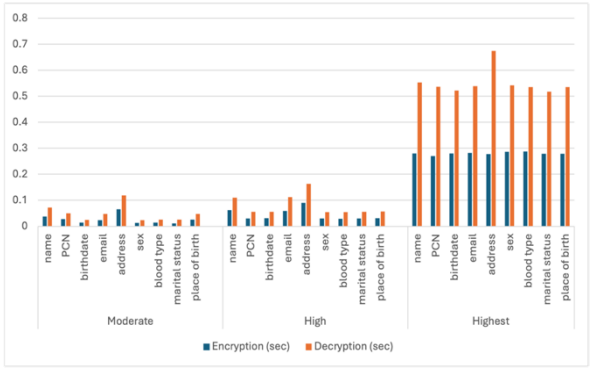

Fig 2. Encryption and Decryption Time per Field

Figure 2 shows the encryption and decryption time per field for each security level. It depicts the increasing encryption and decryption time as the parameters for encryption increases. With the "highest" security level showing a big jump from the other two levels.

Table 3. Encryption, Decryption, and Key Generation Time

	Ave. Encryption Time	Ave. Decryption Time	Key Generation
Moderate	0.0259	0.0483	0.25384
High	0.0435	0.0797	0.690171
Highest	0.2798	0.5504	7.230209

As shown in table 3, the average encryption time for "moderate" security is 0.0259 seconds while decryption takes 0.0483 seconds on average with a maximum of 0.1185 seconds. For "high" security, the average is 0.0435 second, while decryption takes 0.0797 seconds with a maximum of 0.1118 seconds. Lastly, for "highest", encryption takes 0.2798 seconds on average, while decryption takes 0.5504 seconds with a maximum of 0.6744 seconds.

3.3 Security Analysis

According to (Hoffstein et. al., 1999), an attacker can brutely guess the private key by trying all possible polynomials $f \in \mathcal{L}_f$ and testing if $f \circledast h\ (mod\ q)$ has small values, or by trying all polynomials $g \in \mathcal{L}_g$ and testing if $g \circledast h^{-1} (mod\ q)$ has small values. On a similar note, an attacker can recover a message by brutally trying all possible $\phi \in \mathcal{L}_\phi$ and testing if $e - \phi \circledast h\ (mod\ q)$ has small entries. The key and message security can then be computed using the formula:

$$Key\ Security = \sqrt{\#\mathcal{L}_g} = \frac{1}{d_g!}\sqrt{\frac{N!}{(N-2d_g)!}}$$

$$Message\ Security = \sqrt{\#\mathcal{L}_\phi} = \frac{1}{d!}\sqrt{\frac{N!}{(N-2d)!}}$$

With the mentioned security parameters, such numbers were set due to the low probability of decryption failure at 5×10^{-5}. At moderate security, meet-in-the-middle security levels will be at 2^{50} for key security and $2^{26.5}$ for message security. At the highest security, meet-in-the-middle security levels soar at 2^{285} for key security and 2^{170} for message security.

Table 4. Brute Force Attack Time per Security Level

	p	q	N	Time (seconds)	Time (years)
Moderate	3	64	107	1.4×10^{181} seconds	4.44×10^{173} years
High	3	128	167	2.1×10^{340} seconds	6.67×10^{333} years
Highest	3	256	503	1.1×10^{1199} seconds	3.49×10^{1191} years

Table 4 shows the staggering amount of time it takes to recover via brute force attack, each encrypted information at varying security levels. These numbers were computed by taking the key space size (q^N), and then dividing it by 10^{12} attempts per second which is the supposed capability of a theoretical GPU.

3.4 Benchmark Comparison

To fully visualize the strengths (and weaknesses) of NTRUEncrypt, comparison against classical algorithms such as the RSA and ECC, and another lattice-based cryptographic algorithm: Kyber. First, an equivalence should be done for each key size.

Table 5: Key Size Equivalence per Security Level

	NTRU (N)	RSA	ECC	Kyber
Moderate (~112 bits)	107	2048 bits	256 bits	800 bits
High (~128 bits)	167	3072 bits	384 bits	1184 bits
Highest (~256 bits)	503	4096 bits	521 bits	1568 bits

Table 5 shows the equivalence for each cryptographic algorithm in terms of key size, based on the cryptographic guidelines of the National Institute of Standards and Technology (NIST) and the European Union Agency for Cybersecurity (ENISA). Each key size is based on the security level they can achieve measured in bits of security, which correspond to the computational effort required to break the cryptosystem.

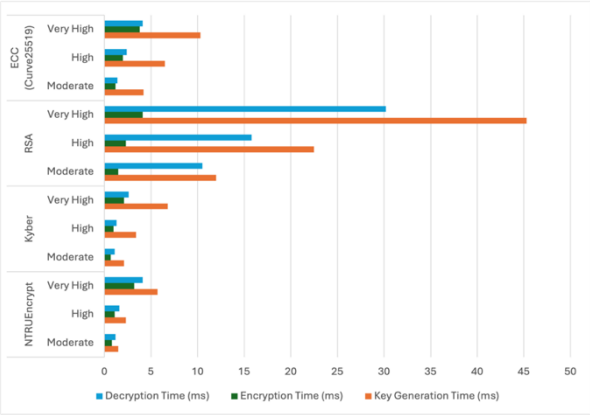

Fig. 3. Benchmark Comparison of NTRU, Kyber, RSA, and ECC for Encryption Time

Figure 3 shows a bar graph comparing the encryption time of NTRU against Kyber, RSA, and ECC at equivalent key sizes. We can see an obviously shorter encryption time for NTRUEncrypt versus the two classical algorithms, and a visibly shorter encryption time of NTRUEncrypt against its post-quantum competitor, Kyber. One can also see a large encryption time for RSA due to its requirement to generate large prime numbers. ECC competes very well against NTRUEncrypt and Kyber but is generally slower than both.

4. CONCLUSIONS

With the development of such system, several analyses showed that NTRUEncrypt generated keys and underwent then process of encryption and decryption, significantly faster than another post-quantum algorithm and several classical algorithms. Furthermore, brute force analysis shows a staggering amount of time to crack encrypted data and generated keys in accordance with several levels of security, differing in parameters. Comparing NTRUEncrypt's performance shows that this algorithm can compete well with mainstream algorithms and among its equal in the post-quantum world.

To improve the system architecture of the web application that has been described in the thesis, the design should be modular for streamlined functionality and scalability. The application can be split into different modules, such as an authentication and encryption module that uses NTRUEncrypt for secure data handling, a user management module to handle registration and role management, a PhilSys integration module for the verification of IDs, and a data storage and logging module that ensures secure and scalable storage. Furthermore, given that the system in this study is only accessed via localhost, future researchers can deploy such project for scalability and access.

While this study successfully demonstrates NTRUEncrypt's technical viability and performance for securing PhilSys ID data, its real-world deployment introduces practical challenges that warrant further consideration. Key management is paramount. For a national system like PhilSys, the secure generation, distribution, storage, rotation, and revocation of NTRUEncrypt keys for millions of citizens and agencies presents a significant undertaking. This necessitates robust, potentially distributed, key management infrastructure and meticulous design for secure key recovery. Compliance with data privacy laws, such as the Philippine Data Privacy Act of 2012 (Republic Act No. 10173), is also critical. While encryption enhances security, the system must ensure adherence to principles like data minimization, consent, and data subject rights (access, rectification, erasure). Mechanisms for auditability and accountability in handling encrypted data are essential for demonstrating regulatory compliance. Further challenges include seamless system integration and interoperability with existing PhilSys infrastructure, potentially involving legacy components. Nationwide scalability demands rigorous testing under high transaction loads to maintain performance and availability. Finally, user experience and accessibility must be prioritized to ensure that cryptographic complexities do not hinder adoption and efficient service delivery for citizens and government personnel. A long-term post-quantum transition strategy, including algorithm updates and cryptographic agility, will also be vital for future-proofing the system.

5. ACKNOWLEDGMENTS

The authors would like to acknowledge every Filipino taxpayer through the efforts of the Department of Science and Technology – Science Education Institute (DOST-SEI) for the guidance and financial support.

6. REFERENCES

Aceron, V. P. (2021). An Evaluative Study on the Citizen's perception towards the Philippine Identification System through the lens of Technology Acceptance Model.

Bernstein, D. J., & Lange, T. (2017). Post-quantum cryptography. Nature, 549(7671), 188-194.

Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R., & Smith-Tone, D. (2016). Report on Post-Quantum Cryptography. Report on Post-Quantum Cryptography. https://doi.org/10.6028/nist.ir.8105

Hoffstein, J., Pipher, J., & Silverman, J. (1999). NTRU: A Ring-Based Public Key Cryptosystem. https://www.ntru.org/f/hps98.pdf

Pangan, A. M., & Lacuesta, I. (2024). View of Authenticating Data Transfer Using RSA-Generated QR Codes. Ej-Compute.org. https://www.ej-compute.org/index.php/compute/article/view/73/33

Shor, P. W. (1944) Algorithms for quantum computation: Discrete logarithms and factoring. In 35th Annual Symposium on Foundations of Computer Science