Received: May 16, 2023. Revised: July 23, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Securing IoT Networks with NTRU Cryptosystem: A Practical Approach on ARM-based Devices for Edge and Fog Layer Integration

M. J. Al-Doori[1]* **M. F. Al-Gailani[1]**

[1]*AlNahrain University, College of Information Engineering, Iraq*
* Corresponding author's Email: m.j.aldoori@gmail.com

Abstract: In this paper, we have presented a comprehensive study optimizing the performance of the N-th degree truncated polynomial ring units (NTRU) post-quantum cryptography (PQC) on advanced RISC machines (ARM)-based internet of things (IoT) devices, with specific focus on Raspberry Pi boards, for integration within edge layer and fog-based IoT architecture. Our objective was to expedite encryption and decryption processing times for five security levels. To achieve this, we adopted Shamir's Secret Sharing (SSS) to facilitate secure NTRU public key exchanges and utilized run-length encoding (RLE) to compress ciphertext effectively. Our results confirmed a compression ratio of approximately 69.25%, suggesting a strong potential for efficient data transfer. The empirical outcomes of our research evidenced considerable improvements in NTRU performance on ARM-based devices, with encryption and decryption times improvement by 80% and 33%, respectively, for 128-bit payloads.

Keywords: IoT security, Edge-fog, Secret sharing, Run-length encoding, NTRU, ARM.

1. Introduction

Quantum computing has made robust PQC essential for protecting communication and data from powerful quantum computers[1]. PQC algorithms, such as NTRU [2], are meant to combat quantum attacks, guaranteeing long-term data protection. The internet of things (IoT) requires substantial security measures owing to the abundance of networked devices that process massive volumes of data [3]. Specifically, with modular architecture comprised from more than fog-cloud layers [4]. Given the inherent constraints of IoT devices, including limited processing power, memory, and energy resources, there is a pressing need for efficient cryptographic solutions that can operate effectively within these constraints. Among PQC algorithms, NTRU cryptosystem a lattice-based approach, has shown considerable promise for IoT environments[5].

NTRU differs from RSA and other classical public-key cryptosystems like elliptic curve-based systems in two ways. First, it looks that NTRU can withstand Shor's algorithm attacks from a quantum computer, making it a plausible post-quantum option. Second, unlike modular exponentiation on 3072-bit integers (required for RSA with 128-bit security) or scalar multiplication in a 256-bit elliptic-curve group, the main arithmetic operation of NTRU is the multiplication ("convolution") of polynomials of degree 438 (for 128-bit security) with small integer coefficients, which is much cheaper. Exponentiation and scalar multiplication cost $O(n3)$ for n-bit operands, while conventional multiplication of two polynomials of degree n costs $O(n2)$. NTRU is suited for low-power devices like smart cards, wireless sensor nodes, and RFID tags.[6]. Its robust security against quantum attacks and relatively fast encryption and decryption speeds, combined with low computational complexity, make it an ideal solution for secure data transfer[7]. ARM processors, frequently utilized in IoT devices, have become increasingly popular due to their energy efficiency and high performance. In this context, the amalgamation of the NTRU cryptosystem and ARM processors can potentially provide a robust and efficient security solution for resource-constrained IoT devices[8].

The problem lies in the reluctance to adopt post-quantum algorithms like NTRU due to ciphertext length overhead. Current literature relies on classical cryptographic algorithms, which are vulnerable to quantum attacks, for confidentiality in IoT. Key exchange methods, such as Diffie-Hellman, suffer from weaknesses like man-in-the-middle attacks and centralized control, unsuitable for IoT's decentralized nature. A pressing need exists to overcome these limitations, develop a solution that combines post-quantum cryptography and IoT requirements, and establish secure key exchange.

The contributions of this research are summarized as follows:

- Real-time implementation of NTRU post-quantum cryptosystem for IoT environment.
- Utilization of Shamir's Secret Sharing (SSS) for secure NTRU public key exchange.
- Adoption of Run-Length Encoding (RLE) for overcoming ciphertext length overhead.
- Testing the proposed system against various security levels.

The remaining paper is organized as follows: Section 2 delivers the related literature. Detailed information on SSS, RLE and the fundamental construction of NTRU cryptosystem is provided in section 3. The details of the proposed system, performance evaluation and security analysis are given in section 4 followed by the conclusion and future work presented in section 5.

Table 1 involves notations used in the research article.

Table 1. Notations

Notation	Description
T	Truncated Polynomial Rings
N	number of polynomial coefficients
p	a chosen small modulus
q	an integer number power of 2
f, g	polynomials generated by Fog node
r	polynomials generated by Edge node
f_p^{-1} , f_q^{-1}	Invers $f\ mode\ p$, Invers $f\ mode\ q$
h	Public key polynomial
a, b	Polynomials calculated by Fog node
e	Polynomial of the ciphertext
CTX	Ciphertext
m	Polynomial of the message
M	Original message
k	Threshold of SSS
PK, PR	Public key, Private key
S	Secret to be shared
s_i	Share of node i
GF	Galois Field
l_i	Lagrange factor
(x, y)	x-axis and y-axis coordinates of share
ENC, DEC	Encryption, Decryption
RLE(D)	Run Length Encoding (Decoding)
ECTX (DCTX)	Encoded (Decoded) CTX

2. Related literature

Recent literature focus on IoT security challenges, where the core network architecture constructed from additional auxiliary layers like edge and fog layers[4] . Others has emphasized the practicality of post-quantum cryptography for IoT devices, for addressing these challenges, despite their resource constraints [9]. These studies highlight its effectiveness in balancing robust security needs against the limited computational capacity and energy efficiency requirements of such devices. As one of post-quantum candidates, many literatures studied the suitability and effectiveness of adapting these algorithms on resource-constrained environment like IoT. In [10] authors implemented numerous state-of-the-art lattice-based authentication protocols on smart cards and a popular microcontroller and provided ideas on how to create or optimize lattice-based schemes for restricted devices. It also explains NTRU encryption and LP-LWE encryption performance and their settings. However, it does not propose a specific system or scheme to evaluate in terms of practical implementation or security levels. Authors in [8] presents four different NTRUEncrypt implementations on an ARM Cortex M0-based microcontroller and provides performance and memory footprint figures for different security parameters. The authors also claim that NTRUEncrypt is suitable for use in battery-operated devices. Authors in [11] proposed post-quantum datagram transport layer security (DTLS) for most cyber physical systems (CPS). CPS entities employ NTRU for transport. In extremely limited contexts like CPSs, their method can integrate with the whole internet engineering task force (IETF) protocol stack. However, an adapted 112-bit security level in the proposed protocol consider a challenge in post-quantum era. For comparative analysis purposes, authors in [12] performed a comparison between classical cryptography and post-quantum one, namely RSA and NTRU, with six security levels. However, authors efforts focus on cloud-native environment, which inherit with high-computation resources.

Received: May 16, 2023. Revised: July 23, 2023.

Lattice-based implementations, just like other post-quantum methods, need to store and make use of large keys, and therefore include substantial ciphertext overheads. This is because the keys themselves are so huge. Lattice-based schemes, such as NTRU or NewHope, for instance, frequently need the management of keys that are on the order of a few thousand bits in length [13]. The trade-off between security and performance is a crucial consideration when implementing lattice-based cryptography. Balancing the need for strong security with practical considerations of performance and efficiency is an ongoing challenge in post-quantum cryptography research and implementation in resource-constrained environment.

For both classical and post-quantum techniques, public key cryptography uses two keys: a public one for encryption and a private one for decryption. Public key dissemination to the other communication party(s) ensures algorithm security. For instance, authors in [14] proposed a secure IoT end-to-end communication. They adapted NTRU as PQ algorithm and Raspberry Pi 3B+ as IoT node. Nevertheless, they sent NTRU public key in clear, thus, compromising the overall system.

Other literature suggested methods for secure public key distribution. Authors in [15] proposed NTRU and Falcon quantum algorithms for key distribution. Two channels—classical and quantum—and three phases assure the distribution. The unnecessary exchanges between fog node and receiver place a large computational burden on resource-constrained IoT nodes, making the proposed solution unsuitable for IoT applications. Where authors in [16] suggested NTRU PQ authentication for IoT. The intermediary gateway node receives the public key from each side. Thus, exposing the public key's privacy, allowing an attacker to spoof the communication.

To the best of our knowledge, none of the previous literature discussed the challenge of NTRU ciphertext length, which consider one of the barriers of NTRU implementation in real-world IoT environment. Secure distribution of NTRU public key, is another issue that require more research efforts to preserve the end-to-end communication privacy among the parties.

3. Preliminaries

3.1 Shamir secret sharing (SSS)

Shamir secret sharing (SSS) is a cryptographic mechanism for partitioning a secret into numerous shares that must be reconstructed [17]. Polynomial interpolation over finite fields makes SSS a secure and fault-tolerant secret distribution system developed by Adi Shamir [18], SSS can be used for key distribution in IoT security, providing an extra layer of safety for key exchange in resource-constrained contexts. The SSS algorithm consists of two phases:

- *Distribution phase*: This phase involves taking some secret data denoted by S, and determining the number of nodes NS that will receive the secret parts s_i. A threshold value k, based on these data, must be determined in order to reconstruct S. The next step is to construct a polynomial function $f(x_i)$ with degree $k-1$, as shown in equation (1), to calculate the secret pieces NS. The $k-1$ coefficients are random integers chosen from $GF(S)$:

$$f(x_i) = \left(\sum_{j=0}^{k-1}\left(a_j x^j\right)\right) mod(S) \tag{1}$$

For $i = \{1, \dots\dots, NS\}$, $a_0 = S$, $\{a_1, \dots., a_{k-1}\} \in GF(S)$
The final step is to distribute the pieces $NS = \{s_1, s_2, \dots ., s_{ns}\}$ to the nodes.

- *Reconstruction phase*: This phase involves reconstructing S from k pieces of data s_i. This require constructing the original f(x), thanks to LaGrange interpolation, that renders this using the formula:

$$l_i = \prod_{M \neq j}\left(\frac{x - x_M}{x_j - x_M}\right) \tag{2}$$

$$f(x) = \sum_{i=0}^{k-1}(y_j l_i) \tag{3}$$

3.2 Run length encoding

Run length encoding (RLE) is a simple yet effective lossless data compression technique that excels in compressing data containing long sequences of repeating elements. It operates by replacing consecutive identical elements, or runs, with a single instance of the element followed by the run's length. RLE is especially suitable for compressing data with low entropy or large areas of uniformity, such as images with solid colours or binary data with long runs of zeros or ones [19]. In the context of IoT security, RLE can be employed to address the challenge of large ciphertext lengths generated by cryptographic algorithms like NTRU [20]. By compressing the encrypted messages using RLE, transmission overhead and storage requirements can be significantly reduced without compromising the security and integrity of the encrypted data. This allows for efficient and secure communication within resource-constrained IoT networks, particularly when dealing with devices with limited processing power, memory, and energy resources.

3.3 NTRU-based PQ cryptosystem

Post-quantum cryptography (PQC) mitigates quantum computing threats to classical cryptography through quantum-resistant encryption algorithms [21, 22]. It capitalizes on mathematical complexities unresolvable by quantum computers across five branches [22]: Multivariate-based, hash-based, code-based, Isogeny-based and lastly, lattice-based cryptography relies on quantum-immune lattice issues like shortest vector problem and learning with errors [23].

NTRU is a lattice-based public-key cryptosystem designed to provide strong security against both classical and quantum computing attacks. It has gained attention in the post-quantum cryptography domain due to its relatively fast encryption and decryption speeds and low computational complexity. NTRU operates in three main stages: key generation, encryption, and decryption [2].

NTRU parameters, N , p and q must be established as an initialization step. N represents the number of polynomial coefficients used in the algorithm, and has an essential effect on the security of the algorithm. A larger N means a higher security level produces by NTRU. Integer coefficients must range between [-1; 1], which means that the coefficients are the numbers -1, 0, or 1.

The parameter P is the prime number used in the modulus operation to keep the range of polynomial coefficients within the range of $N-1$. The last parameter, q is used to satisfy the requirements for choosing the polynomials used in the key generation. The parameter q must be chosen to be a multiple of 2, to calculate the invers modulus of the polynomials

3.3.1. Key generation

In the key generation stage, a private key and a public key are generated for each participant in the communication process [2]. The private key consists of two polynomials, f , and g, while the public key is derived from these polynomials using a public parameter q and a polynomial h. The polynomial f must satisfy two requirements. First, the number of 1's not equal the number of -1's. Second, it has inverses under modulus p and under modulus q. Choosing g polynomial also must satisfy the requirement of equality between the number of 1's and -1's. Eqs. (4) and (5) show these requirements.

$$f f_p = 1 \, mod \, p \tag{4}$$

$$f f_q = 1 \, mod \, q \tag{5}$$

After satisfying the requirements, the public key h is measured by calculating the modular multiplication of $p\, f_q$ and g, modulus q, as shown in Eq. (6), after the last pre-step to make the coefficients of the resulted polynomial positive.

$$h = p f_q g \; (mod \; q) \tag{6}$$

The private key f, f_p and g, is kept secret, whereas the public key h is shared with other participants.

3.3.2. Encryption

To encrypt a message (plaintext) using NTRU, the fog node first encodes the message as a polynomial m. This step involves converting the plaintext into integers using american standard code for information interchange (ASCII), then into binary of 1's and 0's. The fog node then generates a random polynomial, r, which serves as the ephemeral key for this encryption instance. The ciphertext is created by multiplying the recipient's public key (polynomial h) by the ephemeral key r, using modular multiplication, and then adding the encoded message m, using modular addition [2], as presented in Eq. (7).

$$e = rh + m \; (mod \; q) \tag{7}$$

The ciphertext e, the encrypted message, can be transmitted securely over the network.

3.3.3. Decryption

Upon receiving the ciphertext, the recipient can decrypt it using their private key (polynomials f and f_q). The recipient first multiplies the ciphertext e by their private key f, using modulus multiplication, and then applies a modular reduction to obtain the product of the encoded message m and the ephemeral key r [2]. These steps involve making the coefficients of the output polynomial positive and shifting them to be in the range $(-q/2, +q/2)$. Finally, the recipient uses the second private key, polynomial g, to recover the original message m by multiplying it with the product, using modulus multiplication, and applying another modular for reduction, and by shifting the coefficients to be within the range $(-p/2, +p/2)$. Eqs. (8), (9) and (10) show the mathematical calculations for this stage.

$$a = fe \ (mod \ q) \tag{8}$$

$$b = a \ (mod \ p) \tag{9}$$

$$m = f_p b \ (mod \ p) \tag{10}$$

The strength of the NTRU cryptosystem lies in the difficulty of solving the underlying lattice problem, which is believed to be resistant to both classical and quantum attacks. This makes NTRU a promising candidate for securing communications in the era of quantum computing, especially in resource-constrained environments such as IoT devices.

4. Proposed system

The proposed system includes two IoT architectural layers: Edge and Fog layers. The edge layer consists of four edge nodes, and the fog layer involves one fog node. Raspberry Pi 3B+ is chosen as the IoT device for this study due to its widespread use, affordability, and accessibility. It is equipped with an ARM Cortex-A53 processor, which balances the processing power and energy efficiency, making it an ideal candidate for evaluating the performance of the NTRU cryptosystem in resource-constrained environments. NTRU encryption and decryption operations are implemented on the Raspberry Pi 3 B + using five sets of parameters, these parameters, which include the polynomial degree, modulus, and other values, were carefully chosen to ensure robust security while maintaining an acceptable performance on an ARM processor. It is crucial to strike the right balance between security and computational complexity because overly conservative parameters may lead to a higher processing overhead, which is undesirable in IoT scenarios.

4.1 Proposed system phases

The proposed system involves five phases, as the following:

4.1.1. Key pair generation phase

It is the first phase in the proposed system, where Fog node initializes this phase by selecting the parameters N , p , and q and generates two polynomials, f and g, as shown in Eqs. (11) and (12), taking into account the requirements mentioned in section 3.3.1. After calculating the inverse modulus p and q for the f polynomial, the fog node then performs the modular multiplication of polynomial g and the inverse $mod \ q$ of the f polynomial, f_q, as shown in Eqs. (13) and (14) respectively. This phase ends with the generation of two keys: the public key h, as shown in Eq. (15) and the private key f and f_q , as illustrated by Eq. (16).

$$f = a_0 + a_1x^1 + a_2x^2 + \cdots + a_{N-1}x^{N-1} \tag{11}$$

$$g = b_0 + b_1x^1 + b_2x^2 + \cdots + b_{N-1}x^{N-1} \tag{12}$$

$$ff_p = 1(mod \ p) \tag{13}$$

$$ff_q = 1(mod \ q) \tag{14}$$

$$\text{Pubic key } h = pf_q g(mod \ q) \tag{15}$$

$$= c_0 + c_1x^1 + c_2x^2 + \cdots + c_{N-1}x^{N-1}$$

$$\text{Private key } (f, f_p) \tag{16}$$

4.1.2. Shares generation phase

SSS is used in this stage to exchange the NTRU-pubic key that have been generated in the fog node to share it securely with the edge nodes. A Eq. (3) of Eq. (4) threshold scheme is used in this stage, where the fog node generates 4 shares, derived from the NTRU public key, and sends one share per edge node, using the MQTT protocol, via the broker resides on the master edge node, with a highly trust degree. Each edge node then subscribes to the topic forwarded by the master edge node, based on the edge node's ID to prevent sending more than one share to a particular edge node. Hence, each edge node must, in addition to its share, get two more shares to be able to recover the public key of the NTRU algorithm. This method can help mitigate DoS attacks in the key exchange process. An attacker would need to compromise at least Eq. (3) edge nodes to disrupt the key exchange, which is a more difficult task than targeting a single point of failure. Eq. (17) illustratrs that NTRU public key polynomial, h:

$$h = c_0 + c_1x^1 + c_2x^2 + \cdots + c_{N-1}x^{N-1} \tag{17}$$

As shown in Eq. (18), coefficients of h polynomial is extracted and concatenated to construct the secret S that need to be distributed implicitly across its shares.

$$S = c_0 \| c_1 \| c_2 \| \ldots \| c_{N-1} \tag{18}$$

Using total nodes n , and k as a threshold for reconstructing the secret, with Fp_2 which is the finite field of size p_2 , Fog node choose $d_i <$

p_2, and set $d_0 = S$, and construct a polynomial f, as shown in Eq. (19):

$$f = d_0 + d_1x^1 + d_2x^2 + \cdots + d_{k-1}x^{k-1} \quad (19)$$

The next step is calculating the shares as shown in Eq. (20)

$$S_{x-1} = (x, f(x) mod\, p_2) \quad (20)$$

The final step is sending these shares to their associated nodes, based on their ID.

4.1.3. Public key reconstruction

After gathering k-shares from the other nodes, each node starts the phase of reconstructing the public key by calculating the Lagrange factor for each share, as follow in Eq. (21):

$$l_i(x) = \prod_{\substack{0<m<k \\ m\neq j}} \frac{(x-x_0)}{(x_j-x)} \frac{(x-x_{j-1})}{(x_j-x_{j-1})} \frac{(x-x_{j+1})}{(x_j-x_{j+1})} \cdots \frac{(x-x_k)}{(x_j-x_k)} \quad (21)$$

The next step is calculating the Lagrange interpolated polynomial $f'(x)$ as shown in Eq. (22):

$$f'(x) = \sum_{j=0}^{n-1} y_j\, l_i(x) \quad (22)$$
$$= c_0' + c_1'x^1 + c_2'x^2 + \cdots + c_{k-1}'x^{k-1} (mod\, p_2)$$

Then, from Eq. (22), the reconstructed secret S', which is the public key polynomial, is extracted by concatenating the coefficients of polynomial $f'(x)$ as shown below:

$S' = c_0'$

$= c_0 \| c_1 \| c_2 \| \ldots \| c_{N-1}$

The final step is converting this concatenated value into polynomial, aided with indexing information to kept the corrected sequence of the coefficients, to reconstruct h' polynomial as illustrated by Eq. (23).

$$h' = c_0 + c_1x^1 + c_2x^2 + \cdots + c_{N-1}x^{N-1} \quad (23)$$

4.1.4. Encryption

This phase involves converting the sensing data, which represents the message M into ASCII code, and further, into binary digits. The 1's digits of these binary data construct the coefficients of the message polynomial m . Next, encrypted message e is calculated as shown in Eq. (24):

$$e = rh' + m(mod\, q) \quad (24)$$
$$= e_0 + e_1x^1 + e_2x^2 + \cdots + e_{N-1}x^{N-1}$$

The next step is extracted the coefficients of e polynomial and concatenating them, and store the result in variable E, in order to converting the created number into binary digits, which used later as input to RLE, for compressing purpose. The final output is Encoded Ciphertext ($ECTX$), which now ready for sending to fog node.

$E = e_0 \| e_1 \| e_2 \| \ldots \| e_{N-1}$

4.1.5. Decryption

The final phase in the proposed system is decryption phase, which is done at the fog layer. After receiving an encoded ciphertext $ECTX$, Decompression stage is initiated using RLD, in order to output Decoded Ciphertext $DCTX$, which represents e polynomial.

The next step is calculating the polynomials a, b and m' , which represents the polynomial of decrypted message, as shown in Eqs. (25), (26) and (27) respectively:

$$a = fe(mod\, q) \quad (25)$$

$$b = a(mod\, p) \quad (26)$$

$$m' = f_p b(mod\, p) \quad (27)$$

The final step involves converting m' polynomial coefficients back into binary digits, and further, into ASCII code, which led to recovering the original message M.

Fig. 1 illustrate the workflow among the communicating parties in the proposed system.

4.2 Security analysis

In this section, we discuss potential attacks that the proposed cryptographic scheme aims to mitigate or overcome when securing communication in edge-fog computing and IoT environments.

- **Secure Key Distribution**: By using SSS to distribute the NTRU public key among edge nodes and the fog node, the proposed model ensures a secure and robust key distribution mechanism that is resistant to MITM DoS attacks because the attacker must attack more that (n/2) nodes from n edge nodes. In addition, it is resistant to quantum attacks because SSS relies on polynomial interpolation, which is not known to be vulnerable to quantum computing.
- **Hardness of Lattice Problems**: The security of NTRU depends on the hardness of lattice problems, such as the Shortest Vector Problem (SVP) and Learning with Errors (LWE) problem.

Received: May 16, 2023. Revised: July 23, 2023.

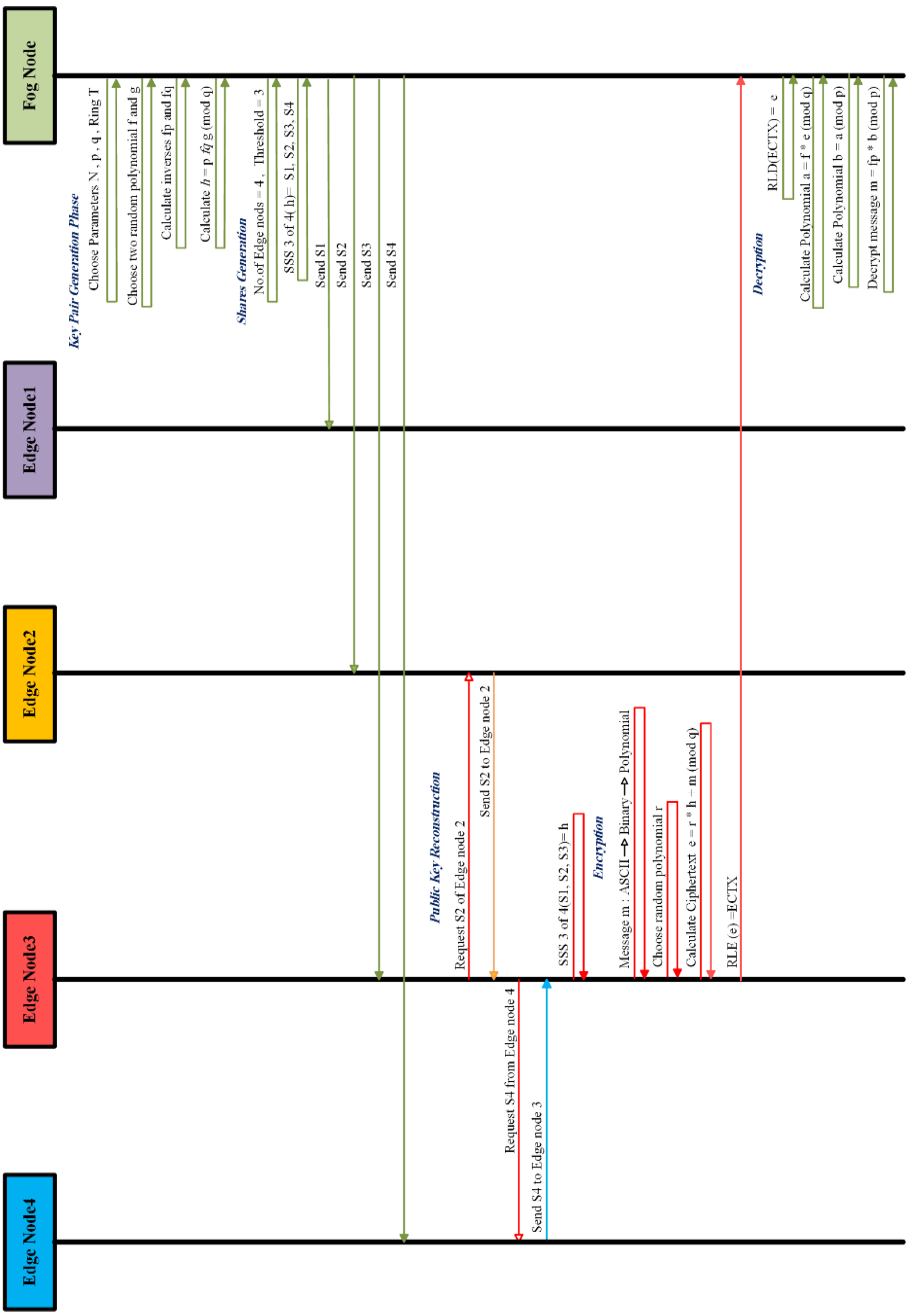

Fog Node
Edge Node1
Edge Node2
Edge Node3
Edge Node4
Key Pair Generation Phase
Choose Parameters N , p , q , Ring T
Choose two random polynomial f and g
Calculate inverses fp and fq
Calculate h = p fq g (mod q)
Shares Generation
No.of Edge nods = 4 , Threshold = 3
SSS 3 of 4(h)= S1, S2, S3, S4
Send S1
Send S2
Send S3
Send S4
Public Key Reconstruction
Request S2 of Edge node 2
Send S2 to Edge node 2
Request S4 from Edge node 4
Send S4 to Edge node 3
SSS 3 of 4(S1, S2, S3)= h
Encryption
Message m : ASCII → Binary → Polynomial
Choose random polynomial r
Calculate Ciphertext e = r * h + m (mod q)
RLE (e) =ECTX
Decryption
RLD(ECTX) = e
Calculate Polynomial a = f * e (mod q)
Calculate Polynomial b = a (mod p)
Decrypt message m = fp * b (mod p)

Figure. 1 System workflow

Table 2. Performance comparison with other implementations

Work	Platform	PK (Bit)	PR (Bit)	Security Estimation	Security Level	N, p, q	Encryption (sec)	Decryption (sec)
[10]	ARM7TDMI	1820	6057	Moderate	128	439,3,2048	0.141	0.203
[8]	ARMv6-M	1838	6158	Moderate	128	443,3,2048	0.018376	0.029699
[8]	ARMv6-M	2434	7970	High	192	587,3,2048	0.032516	0.051150
[8]	ARMv6-M	3078	10250	Highest	256	743,3,2048	0.044111	0.074282
our	ARMv8	4411	1272	Lower	112	401, 3, 2048	0.002697	0.019528
our	ARMv8	4829	1392	Moderate	128	439,3,2048	0.003752	0.019887
our	ARMv8	5357	1543	Standard	160	487,3,2048	0.004919	0.021949
our	ARMv8	6523	1881	High	192	593, 3, 2048	0.005942	0.027615
our	ARMv8	8173	2356	Highest	256	743, 3, 2048	0.006005	0.032110

These problems are thought to be difficult, even for quantum computers, as there are no efficient quantum algorithms currently known to solve them.

- **Resistance to Quantum Attacks**: Unlike RSA and ECC, which are vulnerable to Shor's algorithm, NTRU is considered resistant to known quantum attacks. Grover's algorithm, which is likely to speed up brute-force attacks, will not provide a significant advantage against NTRU, since the security parameters can be increased to maintain an adequate level of security.
- **Ciphertext Length-based Attacks**: The use of the RLE algorithm mitigates the risks associated with long NTRU-based ciphertext lengths by compressing the ciphertext and reducing the likelihood of attacks by exploiting the increased overhead or transmission time.

4.3 Performance evaluation

This section involves an evaluation for the performance of the proposed system, against other related literatures. The proposed system is tested against five security levels, namely Low, Moderate, Standard, High and Highest. And for system validation, both NTRU encryption-decryption together are tested on ARM-based raspberry pi 3b+. Five sets of NTRU parameters are used for that purpose. As shown in Table 2, comparing the proposed system to the relevant literature shows that our ARMv8 platform has faster encryption and decryption timings for 128-bit payloads than the other platforms.

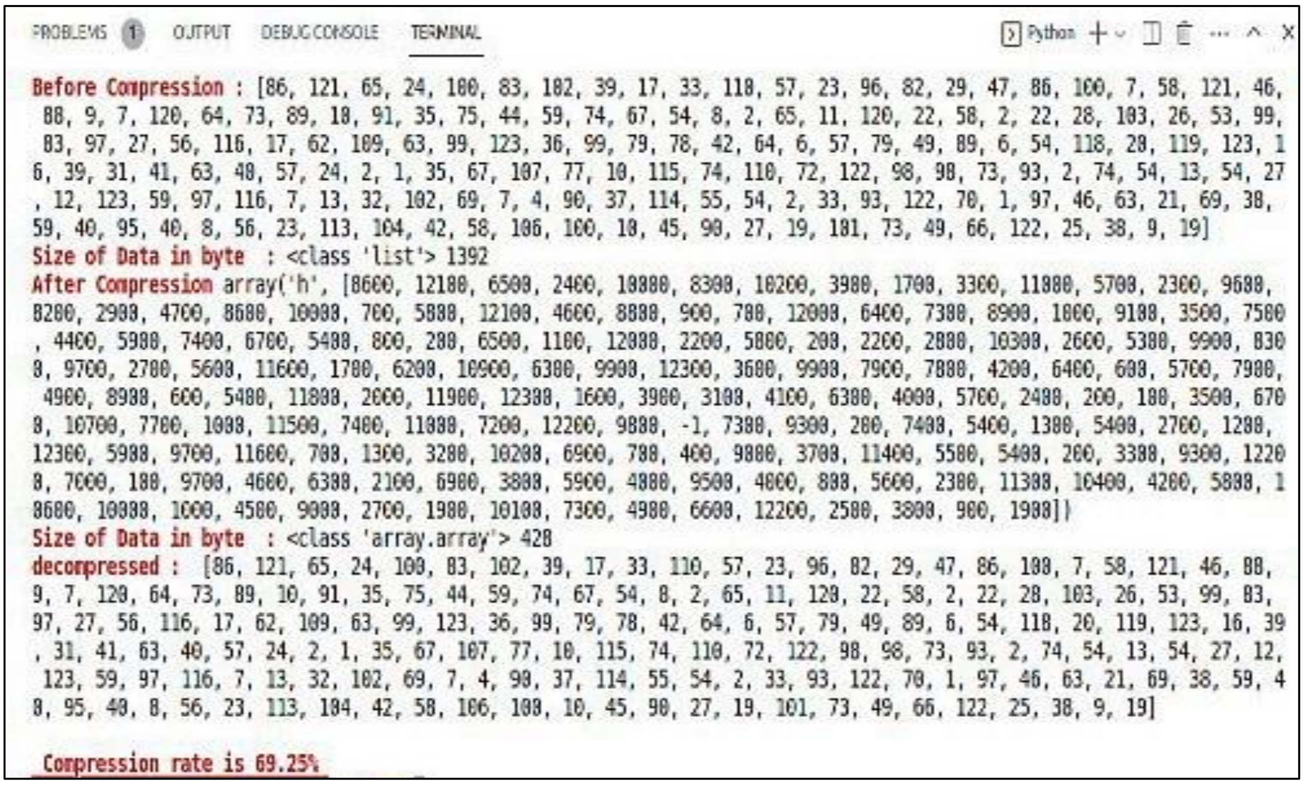
The ARM7TDMI platform [8] encrypts and decrypts 128-bit payloads in 0.141 and 0.203 seconds with moderate security estimation. The ARMv6-M platform [7] has 128-bit payload encryption and decryption latencies of 0.018376 and 0.029699 seconds for moderate security. Our proposed ARMv8 platform, with modest security and 128-bit payload size, reduces encryption and decryption times. Our system encrypts in 0.003752 seconds faster than the ARM7TDMI and ARMv6-M platforms. The ARM7TDMI and ARMv6-M platforms' encryption times improved by 96% and 80%, respectively. Again, our system decrypted in 0.019887 seconds. Our approach improves ARM7TDMI and ARMv6-M decryption times by 90% and 33%, respectively. The suggested ARMv8-based system reduces 128-bit payload encryption and decryption times compared to the literature. Our safe cryptographic transaction system for IoT networks works well. Our proposed method works in real-world IoT networks where efficiency and security are key. Encryption-Decryption time of the other security levels like 160, 192, and 256-bit shows the applicability of the proposed system in IoT networks.

In asymmetric cryptographic systems like NTRU, it's common to see different processing times for encryption and decryption. This is primarily due to the underlying mathematical operations involved in each process. NTRU's encryption process is typically quicker because it involves simpler polynomial multiplication and addition operations.

In NTRU, the encryption process generally involves creating a random polynomial, multiplying it by the public key, and then adding another small polynomial. On the other hand, the decryption process in NTRU is more computationally intensive. It involves polynomial multiplication, modular reduction, and the use of the private key to recover the original message. Modular reduction, especially, can be a complex operation depending on the size of the numbers involved.

In talking about NTRU limitation of ciphertext length, the proposed system overcome this limitation using RLE with compression ratio about %69.25. Fig. 2 shows the ciphertext at edge layer, before and after compression process. It shows that size of ciphertext is 1392 bytes, before compression, and 428 bytes,

Received: May 16, 2023. Revised: July 23, 2023.


PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```
Before Compression : [86, 121, 65, 24, 100, 83, 102, 39, 17, 33, 110, 57, 23, 96, 82, 29, 47, 86, 100, 7, 58, 121, 46,
 88, 9, 7, 120, 64, 73, 89, 10, 91, 35, 75, 44, 59, 74, 67, 54, 8, 2, 65, 11, 120, 22, 58, 2, 22, 28, 103, 26, 53, 99,
 83, 97, 27, 56, 116, 17, 62, 109, 63, 99, 123, 36, 99, 79, 78, 42, 64, 6, 57, 79, 49, 89, 6, 54, 118, 20, 119, 123, 1
6, 39, 31, 41, 63, 40, 57, 24, 2, 1, 35, 67, 107, 77, 10, 115, 74, 110, 72, 122, 98, 98, 73, 93, 2, 74, 54, 13, 54, 27
, 12, 123, 59, 97, 116, 7, 13, 32, 102, 69, 7, 4, 90, 37, 114, 55, 54, 2, 33, 93, 122, 70, 1, 97, 46, 63, 21, 69, 38,
59, 40, 95, 40, 8, 56, 23, 113, 104, 42, 58, 106, 100, 10, 45, 90, 27, 19, 101, 73, 49, 66, 122, 25, 38, 9, 19]
Size of Data in byte  : <class 'list'> 1392
After Compression array('h', [8600, 12100, 6500, 2400, 10000, 8300, 10200, 3900, 1700, 3300, 11000, 5700, 2300, 9600,
8200, 2900, 4700, 8600, 10000, 700, 5800, 12100, 4600, 8800, 900, 700, 12000, 6400, 7300, 8900, 1000, 9100, 3500, 7500
, 4400, 5900, 7400, 6700, 5400, 800, 200, 6500, 1100, 12000, 2200, 5800, 200, 2200, 2800, 10300, 2600, 5300, 9900, 830
0, 9700, 2700, 5600, 11600, 1700, 6200, 10900, 6300, 9900, 12300, 3600, 9900, 7900, 7800, 4200, 6400, 600, 5700, 7900,
 4900, 8900, 600, 5400, 11800, 2000, 11900, 12300, 1600, 3900, 3100, 4100, 6300, 4000, 5700, 2400, 200, 100, 3500, 670
0, 10700, 7700, 1000, 11500, 7400, 11000, 7200, 12200, 9800, -1, 7300, 9300, 200, 7400, 5400, 1300, 5400, 2700, 1200,
12300, 5900, 9700, 11600, 700, 1300, 3200, 10200, 6900, 700, 400, 9000, 3700, 11400, 5500, 5400, 200, 3300, 9300, 1220
0, 7000, 100, 9700, 4600, 6300, 2100, 6900, 3800, 5900, 4000, 9500, 4000, 800, 5600, 2300, 11300, 10400, 4200, 5800, 1
0600, 10000, 1000, 4500, 9000, 2700, 1900, 10100, 7300, 4900, 6600, 12200, 2500, 3800, 900, 1900])
Size of Data in byte  : <class 'array.array'> 428
decompressed :  [86, 121, 65, 24, 100, 83, 102, 39, 17, 33, 110, 57, 23, 96, 82, 29, 47, 86, 100, 7, 58, 121, 46, 88,
9, 7, 120, 64, 73, 89, 10, 91, 35, 75, 44, 59, 74, 67, 54, 8, 2, 65, 11, 120, 22, 58, 2, 22, 28, 103, 26, 53, 99, 83,
97, 27, 56, 116, 17, 62, 109, 63, 99, 123, 36, 99, 79, 78, 42, 64, 6, 57, 79, 49, 89, 6, 54, 118, 20, 119, 123, 16, 39
, 31, 41, 63, 40, 57, 24, 2, 1, 35, 67, 107, 77, 10, 115, 74, 110, 72, 122, 98, 98, 73, 93, 2, 74, 54, 13, 54, 27, 12,
 123, 59, 97, 116, 7, 13, 32, 102, 69, 7, 4, 90, 37, 114, 55, 54, 2, 33, 93, 122, 70, 1, 97, 46, 63, 21, 69, 38, 59, 4
0, 95, 40, 8, 56, 23, 113, 104, 42, 58, 106, 100, 10, 45, 90, 27, 19, 101, 73, 49, 66, 122, 25, 38, 9, 19]

 Compression rate is 69.25%
```

Figure. 2 Ciphertext Compression Ratio

Table 3. Compression ratio comparison with other implementation

Work	Platform	PQ	Compression Technique	Before, After Compression	Compression Rate
[20]	Cortex M0	R-LWE	Custom lattice	768, 460	< % 40
our	CortexM3	NTRU	RLE	1392, 428	% 69.25

after compression. Table 3 compare our proposed system with other related literature.

4.4 Experimental environment

The experimental environment of the proposed system involved four devices, three edge nodes, and one fog node. Raspberry Pi 3B is used as a development board, representing an edge node. It has a 1.2 GHz CPU, belongs to the Cortex-A53 family, 1GB RAM, and 32GB storage space. Raspbian is the operating system used in these boards. On the other hand, Lenovo with 2.2 GHz Ci5 and 8GB RAM, and 500GB storage space are the specifications of the fog node used in this environment. Ubuntu 16.4 LTS is used as an operating system for that fog node.

5. Conclusion and future work

In conclusion, the NTRU post-quantum cryptosystem was successfully implemented in this study on an ARM-based architecture typically used in IoT networks. Our method greatly outperformed the encryption and decryption timings of other comparable literature by addressing the security-performance trade-off, a common issue in IoT applications, especially with quantum cryptography algorithms. Compared to the ARM7TDMI and ARMv6-M platforms, our solution improved encryption times by about 96% and 80%, and decryption times by about 90% and 33% for 128-bit payloads.

Key distribution in IoT cryptosystems was made more secure by the system's implementation of Shamir's secret sharing, which reduced the impact of attacks like Man in the Middle and Denial of Service. Furthermore, our system's use of run-length encoding to compress ciphertext efficiently overcame the problem of NTRU's excessively high ciphertext size, resulting in a compression ratio of roughly 69.25%. Our suggested system is well-suited for practical implementations of IoT networks thanks to these changes, which led to secure and efficient cryptographic solutions while maintaining speed and security.

Our system's modular design, featuring distinct edge and fog layers, significantly decreased the typical lag time associated with cloud-based IoT networks. Future research into improving the security of IoT networks will focus on further optimizing for different types of IoT devices, integrating with other post-quantum techniques, and exploring hybrid cryptographic systems

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The paper background work, conceptualization, methodology, implementation, result analysis and comparison, preparing and editing draft, have been done by the first author. The supervision, review of work and project administration, have been done by the second author.

References

[1] L. Malina, L. Popelova, P. Dzurenda, J. Hajny, and Z. Martinasek, "On Feasibility of Post-Quantum Cryptography on Small Devices", *IFAC-PapersOnLine*, Vol. 51, No. 6, pp. 462–467, Jan. 2018.

[2] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 1423, pp. 267–288, 1998.

[3] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen, and B. Stiller, "Landscape of IoT security", *Comput. Sci. Rev.*, Vol. 44, p. 100467, May 2022.

[4] A. K. Jumani, J. Shi, A. A. Laghari, Z. Hu, A. U. Nabi, and H. Qian, "Fog computing security: A review", *Secur. Priv.*, Mar. 2023.

[5] T. M. F. Carames, "From Pre-Quantum to Post-Quantum IoT Security: A Survey on Quantum-Resistant Cryptosystems for the Internet of Things", *IEEE Internet Things J.*, Vol. 7, No. 7, pp. 6457–6480, Jul. 2020.

[6] H. Cheng, J. Grobschadl, P. B. Ronne, and P. Y. A. Ryan, "Avrntru: Lightweight NTRU-based Post-Quantum Cryptography for 8-bit AVR Microcontrollers", In: *Proc. of Design, Autom. Test Eur. DATE*, Vol. 2021-Febru, pp. 1272–1277, Feb. 2021.

[7] J. Senor, J. Portilla, and G. Mujica, "Analysis of the NTRU Post-Quantum Cryptographic Scheme in Constrained IoT Edge Devices", *IEEE Internet Things J.*, 2022.

[8] O. M. Guillen, T. Poppelmann, J. M. B. Mera, E. F. Bongenaar, G. Sigl, and J. Sepulveda, "Towards post-quantum security for IoT endpoints with NTRU", In: *Proc. of 2017 Des. Autom. Test Eur. DATE 2017*, pp. 698–703, 2017.

[9] Z. Liu, K. K. R. Choo, and J. Grossschadl, "Securing Edge Devices in the Post-Quantum Internet of Things Using Lattice-Based Cryptography", *IEEE Commun. Mag.*, Vol. 56, No. 2, pp. 158–162, 2018.

[10] A. Boorghany, S. B. Sarmadi, and R. Jalili, "On Constrained Implementation of Lattice-Based Cryptographic Primitives and Schemes on Smart Cards", *ACM Trans. Embed. Comput. Syst.*, Vol. 14, No. 3, 2015.

[11] J. Sepulveda, S. Liu, and J. M. B. Mera, "Post-Quantum Enabled Cyber Physical Systems", *IEEE Embed. Syst. Lett.*, Vol. 11, No. 4, pp. 106–110, 2019.

[12] B. Harjito, H. N. Tyas, E. Suryani, and D. W. Wardani, "Comparative Analysis of RSA and NTRU Algorithms and Implementation in the Cloud", *Int. J. Adv. Comput. Sci. Appl.*, Vol. 13, No. 3, pp. 157–164, 2022.

[13] T. M. F. Carames and P. F. Lamas, "Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks", *IEEE Access*, Vol. 8, pp. 21091–21116, 2020.

[14] Y. M. Agus, M. A. Murti, F. Kurniawan, N. D. W. Cahyani, and G. B. Satrya, "An efficient implementation of NTRU encryption in post-quantum internet of things", In: *Proc. of 2020 27th Int. Conf. Telecommun. ICT 2020*, 2020.

[15] A. Prakasan, K. Jain, and P. Krishnan, "Authenticated-Encryption in the Quantum Key Distribution Classical Channel Using Post-Quantum Cryptography", In: *Proc. of 2022 6th Int. Conf. Intell. Comput. Control Syst. ICICCS 2022*, pp. 804–811, 2022.

[16] S. H. Jeong, K. S. Park, Y. H. Park, and Y. H. Park, "An efficient NTRU-based authentication protocol in IoT environment", *Adv. Intell. Syst. Comput.*, Vol. 857, pp. 1262–1268, 2019.

[17] P. Sarosh, S. A. Parah, and G. M. Bhat, "Utilization of secret sharing technology for secure communication: a state-of-the-art review", *Multimed. Tools Appl.*, Vol. 80, No. 1, pp. 517–541, 2021.

[18] A. Shamir, "How to share a secret", *Commun. ACM*, Vol. 22, No. 11, pp. 612–613, Nov. 1979.

[19] K. Sayood, *Introduction to Data Compression*, 4th Edition, Elsevier, Jan. 2012.

[20] M. J. O. Saarinen, "Ring-LWE ciphertext compression and error correction tools for lightweight post-quantum cryptography", In: *Proc. 3rd ACM Int. Work. IoT Privacy, Trust. Secur. Co-Located with ASIA CCS 2017*, Vol. 17,

pp. 15–22, Apr. 2017.

[21] R. Bavdekar, E. J. Chopde, A. Agrawal, A. Bhatia, and K. Tiwari, "Post Quantum Cryptography: A Review of Techniques, Challenges and Standardizations", In: *Proc. of Int. Conf. Inf. Netw.*, Vol. 2023-Janua, pp. 146–151, 2023.

[22] M. Kumar, "Post-quantum cryptography Algorithm's standardization and performance analysis", *Array*, Vol. 15, p. 100242, Sep. 2022.

[23] S. Khot, "Hardness of approximating the shortest vector problem in lattices", *J. ACM*, Vol. 52, No. 5, pp. 789–808, Sep. 2005.