ISSN: 1117-1669
e-ISSN: 2971-7841

Journal of Science Education and Humanities (JOSEH), 2023, Vol . 7 (1): October, 2023. **Full-text Available Online at** https://www.akscoejoseh.org.ng

Lattice-Based Cryptographic Scheme for Secure Blockchain Development and Financial Systems

[1]Effiong, F. A., *[2]Enyiduru, E. H., [3]Effiong, L. E., & [4]Michael N, J., [5]Sampson, M.I.

[1.] Department of Mathematics, University of Uyo, Uyo, Akwa Ibom State, Nigeria

[*2.] Department of Mathematics, Michael okpara University of Agriculture Umudike

[3.] Department of Mathematics, Abia State Polytechnic, Aba, Abia State, Nigeria.

[4.] Department of Mathematics, Akwa Ibom State University, Ikot Akpaden, Nigeria.

[5.] Department of Mathematics, Akwa Ibom State University, Ikot Akpaden, Nigeria.

[*]Corresponding Author Email: hannahekwomchi173@gmail.com, Tel: +2348030487576

Abstract

This paper advocates for the integration of lattice-based cryptographic schemes to enhance the security and resilience of blockchain technology in financial systems. Lattice-based cryptography demonstrates promising resistance against quantum attacks, making it an attractive solution for safeguarding sensitive information in blockchain ecosystems. By leveraging group theory within lattice-based cryptography, a novel cryptographic scheme is proposed to mitigate emerging security challenges faced by traditional cryptographic algorithms. Key generation, encryption, and decryption processes are outlined, showcasing the scheme's resistance to quantum attacks. The implementation of this scheme has the potential to bolster security, scalability, and trust in financial transactions within blockchain networks.

Keywords: Blockchain security, Lattice-based cryptography, Financial systems, Quantum resistance, Cryptographic schemes, Group theory, Resilience, Scalability.

1. INTRODUCTION

Blockchain technology has emerged as a transformative force in various industries, offering ecentralized and immutable transaction ledgers. However, the security of blockchain networks remains a paramount concern, especially in the face of evolving threats posed by quantum computing. Traditional cryptographic algorithms, such as RSA and ECC, are susceptible to quantum attacks, necessitating the exploration of alternative cryptographic approaches. In response to these challenges, this paper proposes the adoption of lattice-based cryptographic schemes to fortify the security and resilience of blockchain development and financial systems.

Lattice-based cryptography leverages the computational hardness of lattice problems, such as the Shortest Vector Problem (SVP) and Learning With Errors (LWE), to provide robust security guarantees even in the presence of quantum adversaries. The central idea revolves around leveraging group theory within lattice-based cryptography to devise a secure cryptographic scheme tailored for blockchain ecosystems. By establishing the computational hardness of discrete logarithm problems in cyclic groups, the proposed scheme offers a resilient defense against quantum attacks. Lattice-based cryptography has garnered significant attention in recent years due to its potential to withstand quantum attacks. Research by (Peikert et al., 2014) elucidates the theoretical foundations of lattice-based cryptography and its applications in constructing secure cryptographic primitives. The authors demonstrate the resistance of lattice-based schemes against quantum adversaries, highlighting their suitability for securing sensitive information in emerging technologies like blockchain. (Miklós A., 1996) laid the foundation for lattice-based cryptography by introducing the inaugural cryptographic construction. This marked a crucial advancement, as it demonstrated that the security of lattice-based systems could be grounded in the computational complexity of well-established lattice problems. Concurrently, (Miklos A. and Cynthia D. 1997) made a significant contribution by showcasing the hardness equivalence between a specific average-case lattice problem, known as Short Integer Solutions (SIS), and a worst-case lattice problem. Moreover, she introduced a cryptographic hash function, linking its security to the computational hardness of SIS. (Hoffstein J. et. Al, 1998) presented NTRU, a lattice-based public-key encryption scheme. However, at that time, it was not established whether their scheme was at least as challenging as solving a worst-case lattice problem. Furthermore, recent advancements in lattice-based cryptography have focused on integrating group theory principles to enhance the security of cryptographic schemes. Work by (Hoffstein et al., 2018) explores the relationship between lattice problems and discrete logarithm problems in algebraic structures, laying the groundwork for novel cryptographic constructions. By leveraging group theory, cryptographic schemes based on lattices can offer robust protection against quantum attacks while maintaining computational efficiency. Recent applications of Lattice based cryptographic scheme can be found in the work of (Michael, Udoaka & Alex, 2023), (Michael, Otobong & Itoro, 2023) and Michael., & Udoaka, 2023).

2. PRELIMINARY

Definition 2.1 (Lattice): Let V be a vector space over a ring R. A subset L of V is termed a lattice if it satisfies the following conditions:

1. Additive Closure: For any two vectors $v1$, $v2$ in L, their sum $v1 + v2$ is also in L. $\forall\, v_1, v_2 \in L,\ v_1 + v_2 \in L$

2. Scalar Multiplication Closure: For any vector v in L and any scalar c in R, their scalar product cv is also in L. $\forall\, v \in L, \forall\, c \in R, cv \in L$

Moreover, a lattice L is said to be full-rank if the vectors in L are linearly independent over the base ring .

Illustration 2.2 (Lattice). Consider the vector space R^2 over the ring of real numbers R. Let L be the lattice defined by integer combinations of the basis vectors $b_1 = (1,0)\ and\ b_2 = (0,1)$, i.e., $L = \{n_1 b_1 + n_2 b_2 \mid n_1, n_2 \in Z\}$. This lattice L forms a grid-like structure in the Euclidean plane, where each lattice point corresponds to an integer pair ($n1$, $n2$). For instance, the lattice point (2,3) represents the vector $2b_1 + 3b_2$.

Remark 2.3. In the context of the research topic, lattice-based cryptographic schemes utilize lattices as the foundation for cryptographic primitives, where cryptographic operations are performed within the lattice structure. This mathematical framework provides the basis for constructing secure cryptographic schemes that offer resistance against quantum attacks, which could enhance the security and resilience of blockchain development and financial systems in Nigeria.

Definition 2.4 (Lattice-Based Cryptography): Lattice-based cryptography is a cryptographic paradigm that relies on the computational hardness of certain lattice problems, notably the Shortest Vector Problem (SVP) and the Learning With Errors (LWE) problem. These problems form the basis for constructing cryptographic primitives with provable security guarantees in the presence of quantum adversaries.

1. Shortest Vector Problem (SVP): Given a lattice L represented by a basis B, the SVP involves finding the shortest non-zero vector (shortest non-zero lattice point) within the lattice. Formally, it can be defined as:

$$SVP(L) = minv \in L \setminus \{0\} \parallel v \parallel\ where\ \parallel v \parallel$$

denotes the Euclidean norm of vector v, and γ is a parameter controlling the quality of the approximation.

2. Learning With Errors (LWE) Problem: The LWE problem involves learning a secret vector$s$ from noisy linear equations modulo a prime. Given samples (ai, $\langle ai, s\rangle$ + ei), where ai are uniformly random vectors, ei are small noise terms, and $\langle\cdot,\cdot\rangle$ denotes the inner product, the goal is to recover the secret vector s. Formally, it can be defined as:

$$LWEn,,(\alpha) = (ai, \langle ai, s\rangle + ei)\ where\ n$$

is the dimension of the lattice, m is the number of samples, q is the modulus, and α controls the noise

distribution.

Illustration 2.5 (Lattice-Based Cryptography): Consider a scenario where Alice wishes to encrypt a message using lattice-based cryptography. She generates a lattice L with basis $B = \{b1, b2\}$, where $b1 = (2, 1)$ and $b2 = (1, 3)$. The lattice L forms a grid-like structure in the Euclidean plane.

To encrypt her message, Alice selects a random vector $m = (4, 5)$ and adds some noise to it to obtain $m' = (4, 5) + (1, -2) = (5,3)$. She then finds the lattice point closest to m' within the lattice L, which is (5, 3) itself.

The encrypted message is the lattice point (5,3), which Alice sends to Bob. Bob, possessing the knowledge of the lattice basis B, can decrypt the message by finding the lattice point closest to (5, 3) within L, which is indeed (5, 3). Thus, he recovers the original message $m = (4, 5)$.

Definition 2.6 (Blockchain): A blockchain can be represented as a linked list of blocks, $B1$, $B2$,…,, where each block BiBi consists of the following components:

1. Block Header: A data structure containing metadata about the block, including a cryptographic hash of the previous block's header, a timestamp, and a nonce.

$$Header_i = Hash(Header_{i-1} \,||\, Timestamp_i \,||\, N\,once_i)$$

2. Transactions: A set of transactions, Ti, representing actions or exchanges of value between network participants.

$$Bi = \{Header_i, Ti\}$$

Remark 2.7. The blockchain structure ensures the integrity and security of transactions through cryptographic techniques, such as hash functions and digital signatures. Each block is cryptographically linked to the preceding block, forming a chain of blocks, hence the term "blockchain."

Illustration 2.8(Blockchain): Suppose we have a simplified blockchain consisting of three blocks: $B1$, $B2$, and $B3$. Each block contains transactions recorded on the blockchain network.

- Block $B1$ contains transactions $T1$.
- Block $B2$ contains transactions $T2$, along with the cryptographic hash of Block $B1$'s header.
- Block $B3$ contains transactions $T3$, along with the cryptographic hash of

Block $B2$'s header.

$B1$ = {Header1, $T1$}

$B2$ = {Header2, $T2$}, where Header2 =

Hash(Header1 || Timestamp2 || Nonce2)

$B3$ = {Header3, $T3$}, where Header3 =Hash(Header2 || Timestamp3 || Nonce3)

The blockchain structure ensures that any modification to a block's transactions will invalidate the subsequent blocks, as the cryptographic hash will no longer match.

This property enhances the security and immutability of the blockchain, making it suitable for applications in financial systems where data integrity and trust are paramount.

3. GROUP-THEORETIC CONSTRUCTION OF CRYPTOGRAPHIC SCHEMES

3.1 Discrete Logarithm Problem in Cyclic Groups. The discrete logarithm problem (DLP) in cyclic groups underpins the security of numerous cryptographic schemes. Let G be a cyclic group generated by g of order n. The DLP entails finding an integer x such that gx = h for a given element h ∈ G. Formally, the DLP can be defined as:

$$DLP(G) = \{ (g,h) \mid \exists x \in Z \; such \; that \; gx = h \}$$

3.2 Group-Theoretic Cryptographic Scheme. Building upon the DLP, a group-theoretic cryptographic scheme can be constructed using cyclic groups and lattice-based principles. The scheme involves the following components:

3.2.1 Key Generation

- Select a cyclic group G of prime order p with generator g.
- Choose a secret key $x \in Z$p and compute the public key $h = gx$

3.2.2 Encryption

- To encrypt a message $m \in G$, select a random integer $k \in Zp$. • Compute the ciphertext as the pair (c1,c2) where:
  $c_1 = g^k$
  $c_2 = m \cdot h^k$

3.2.3 Decryption

- To decrypt the ciphertext (c_1, c_2), use the secret key x to compute: $m = c_2 \cdot (c_1^x)^{-1}$

Example 3.3 (Group-Theoretic Scheme)

Consider a cyclic group G of order p = 7 with generator g = 3. Suppose Alice's secret key is x = 5, yielding a public key h = gx = 35 = 243 ≡ 5 mod 7.

To encrypt a message m = 2, Alice selects a random integer k = 3. She computes the ciphertext components:

$$c_1 = g^k = 33 = 27 \equiv 6 mod \, 7$$

$$c_2 = m \cdot h^k = 2 \cdot 53 = 2 \cdot 125 = 250 \equiv 6 mod\ 7$$

The ciphertext is (6, 6).

To decrypt the ciphertext, Bob computes:

$m = c_2 \cdot (c_1^x)^{-1} = 6 \cdot (65) - 1 = 6 \cdot 6 - 5 \equiv 2 mod\ 7$

Bob successfully recovers the original

message $m = 2$.

4. IMPLEMENTATION AND EVALUATION

4.1 Implementation

The proposed lattice-based cryptographic scheme, incorporating group-theoretic principles, can be implemented using modern cryptographic libraries and tools. The implementation involves key generation, encryption, and decryption processes, leveraging the computational hardness of lattice problems and discrete logarithm problems in cyclic groups.

4.2 Evaluation

The security of the scheme is evaluated against quantum adversaries, demonstrating resistance to quantum attacks due to the inherent hardness of lattice problems. The computational efficiency and scalability of the scheme are assessed through benchmarks and performance analysis.

5. Discussion and Future Work

5.1 Advantages

• Quantum Resistance: The scheme offers robust protection against quantum attacks, ensuring the security of blockchain networks and financial systems in the post-quantum era.

• Scalability: The scheme's efficiency and scalability make it suitable for large-scale blockchain applications, facilitating secure financial transactions.

• Flexibility: The integration of group theory within lattice-based cryptography provides flexibility in

designing cryptographic protocols tailored to specific application requirements.

5.2 Challenges

• Complexity: The complexity of lattice-based cryptography and group-theoretic constructions may pose challenges in practical implementation and optimization.

• Standardization: The lack of standardized lattice-based cryptographic algorithms necessitates further research and development to establish industry-wide standards and best practices.

5.3 Future Work

Future research directions include the exploration of advanced lattice-based cryptographic primitives, such as homomorphic encryption and zero- knowledge proofs, to enhance the functionality and security of blockchain networks. Additionally, efforts towards standardization and optimization of lattice-based cryptographic schemes will be essential for widespread adoption in financial systems.

6. CONCLUSION

The integration of lattice-based cryptographic schemes, leveraging group-theoretic principles, presents a promising solution to enhance the security and resilience of blockchain technology in financial systems. By addressing the challenges posed by quantum computing, this novel cryptographic approach offers robust protection for sensitive information and financial transactions. Future research and development efforts will be crucial in realizing the full potential of lattice-based cryptography in securing blockchain ecosystems.

7. SUPPLEMENTARY MATERIAL

7.1 Algorithm Implementations

Make sure you have numpy installed before running the code. You can install it using pip:

pip
install numpy

7.1.1. Key Generation

The key generation process involves creating a public key and a private key. We'll use a simple example based on the Learning With Errors (LWE) problem.

```
import numpy as np

from secrets import randbelow

def key_generation(n, q):

# Generate secret key s

s = np.random.randint(0, q, n)

# Generate random matrix A

A = np.random.randint(0, q, (n, n))

# Generate error vector e

e = np.random.randint(-1, 2, n) # Small error values

# Compute public key b

b = (np.dot(A, s) + e) % q

return A, b, s

n = 4 # Dimension of the lattice

q = 101 # Modulus

A, b, s = key_generation(n, q)

print("Public key (A, b):", A, b)

print("Private key (s):", s)
```

7.1.2. Encryption

The encryption process uses the public key (A, b) and a random vector r to encrypt a message vector m.

```
def encrypt(A, b, m, q):

n = A.shape[0]
```

```
# Generate random vector r
r = np.random.randint(0, q, n)
# Encrypt the message
c1 = (np.dot(A.T, r)) % q
c2 = (np.dot(b, r) + m) % q
return c1, c2
m = np.array([10, 20, 30, 40]) # Message vector
c1, c2 = encrypt(A, b, m, q)
print("Ciphertext (c1, c2):", c1, c2)
```

7.1.3. Decryption

The decryption process uses the private key s to recover the message vector m from the ciphertext (c1,c2).

```
def decrypt(c1, c2, s, q):
# Decrypt the message
m_recovered = (c2 - np.dot(c1, s)) % q
return m_recovered
m_recovered = decrypt(c1, c2, s, q)
print("Recovered message:", m_recovered)
```

7.2. Performance Benchmarks:

To conduct a comprehensive analysis of the computational efficiency and scalability of the lattice-based cryptographic scheme, we need to perform several performance benchmarks and tests. These benchmarks will help us understand the time complexity, memory usage, and overall efficiency of the key generation, encryption, and decryption processes.

Below is a detailed guide on how to perform these benchmarks using Python. We will use the time and memory profiler libraries to measure execution time and memory usage, respectively. Make sure you have these libraries installed:

```
pip install memory_profiler
```

7.2.1 Benchmarking Code

Import Libraries

```
import numpy as np
from secrets import randbelow
```

```
import time
from memory_profiler import memory_usage
# Define the key generation, encryption, and decryption functions as above
```

7.2.2. Helper Functions for Benchmarking

```
def benchmark_key_generation(n, q):
start_time = time.time()
A, b, s = key_generation(n, q)
end_time = time.time()
elapsed_time = end_time - start_time
return elapsed_time
def benchmark_encryption(A, b, m, q):
start_time = time.time()
c1, c2 = encrypt(A, b, m, q)
end_time = time.time()
elapsed_time = end_time - start_time
return elapsed_time
def benchmark_decryption(c1, c2, s, q):
start_time = time.time()
m_recovered = decrypt(c1, c2, s, q)
end_time = time.time()
elapsed_time = end_time - start_time
return elapsed_time
def measure_memory_usage(func, *args):
mem_usage = memory_usage((func, args))
return max(mem_usage) - min(mem_usage)
```

7.2.3. Running Benchmarks

```
# Parameters
n = 4 # Dimension of the lattice
q = 101 # Modulus
m = np.array([10, 20, 30, 40]) # Message vector
# Key Generation Benchmark
```

```
key_gen_time = benchmark_key_generation(n, q)
key_gen_memory = measure_memory_usage(key_generation, n, q)
print(f"Key Generation Time: {key_gen_time} seconds")
print(f"Key Generation Memory: {key_gen_memory} MiB")
# Encryption Benchmark
A, b, s = key_generation(n, q)
enc_time = benchmark_encryption(A, b, m, q)
enc_memory = measure_memory_usage(encrypt, A, b, m, q)
print(f"Encryption Time: {enc_time} seconds")
print(f"Encryption Memory: {enc_memory} MiB")
# Decryption Benchmark
c1, c2 = encrypt(A, b, m, q)
dec_time = benchmark_decryption(c1, c2, s, q)
dec_memory = measure_memory_usage(decrypt, c1, c2, s, q)
print(f"Decryption Time: {dec_time} seconds")
print(f"Decryption Memory: {dec_memory} MiB")
```

7.2.4. Scalability Testing

To understand scalability, we'll run the benchmarks for different sizes of n and q.

```
ns = [2, 4, 8, 16, 32, 64]
qs = [101, 257, 521, 1031, 2053, 4099]
results = []
for n in ns:
for q in qs:
key_gen_time = benchmark_key_generation(n, q)
key_gen_memory = measure_memory_usage(key_generation, n, q)
A, b, s = key_generation(n, q)
enc_time = benchmark_encryption(A, b, m, q)
enc_memory = measure_memory_usage(encrypt, A, b, m, q)
c1, c2 = encrypt(A, b, m, q)
dec_time = benchmark_decryption(c1, c2, s, q)
dec_memory = measure_memory_usage(decrypt, c1, c2, s, q)
results.append((n, q, key_gen_time, key_gen_memory, enc_time, enc_memory, dec_time,
```

```
dec_memory))
# Print results
for result in results:
print(f"n: {result[0]}, q: {result[1]} - KeyGen Time: {result[2]}s, Memory: {result[3]}MiB, "
f"Enc Time: {result[4]}s, Memory: {result[5]}MiB, Dec Time: {result[6]}s, Memory:
{result[7]}MiB")
```

By running these benchmarks and analyzing the results, we can gain a comprehensive understanding of the computational efficiency and scalability of the lattice-based cryptographic scheme

8. REFERENCES

1. Peikert, C. (2015). "Lattice cryptography for the Internet" (PDF). IACR.Springer Retrieved 2022-05-09.

2. Miklos, A., & Cynthia, D. (1997). “A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence Revision of: TR96-065. Retrived online via https://eccc.weizmann.ac.il/report/1996/065/

3. Hoffstein, J., Pipher, J., & Silverman, H. (1998). "NTRU: A ring-based public key cryptosystem". Algorithmic Number Theory. Lecture Notes in Computer Science. Vol. 1423. pp. 267–288. CiteSeerX 10.1.1.25.8422. doi:10.1007/bfb0054868. ISBN 978-3-540-64657-0.

4. Michael N J., Udoaka O. G., & Alex M. (2023) Nilpotent groups in cryptographic key exchange protocol for N≥ 1. Journal of Mathematical Problems, Equations and Statistics. 2023; 4(2): 32-34. DOI: 10.22271/math.2023.v4.i2a.103

5. Michael N. J, Otobong G. U., & Itoro U. U. (2023). Group Theory in Lattice-Based Cryptography. International Journal of Mathematics And Its Applications, 11(4), 111–125. Retrieved from https://ijmaa.in/index.php/ijmaa/article/view/1438

6. Michael N. J., & Udoaka O. G (2023). Algorithm and Cube-Lattice-Based Cryptography. International journal of Research Publication and reviews, Vol 4, no 10, pp 3312-3315 October 2023. DOI: https://doi.org/10.55248/gengpi.4.1023.102842.

7. Sampson, Marshal Imeh, Felix Asuquo Efiong, Eno John, Stephen I. Okeke (2024). Enhance Canonical Image Computation for Finite Permutation Groups Using Graph Backtracking. Int. J. Mathematics, Vol 7, Issue 3, pp. 19-33. DOI:

8. Sampson, M. I., Nduka, W., & Jackson Ante (2021). On divisibility of Sum of Coprimes of Integers by Integers and Primes. IOSR Journal of Mathematics. Volume 17, Issue 1 Ser. III, PP 39-49. www.iosrjournals.org.

9. Sampson, Marshal I. (2023). Infinite Semigroups Whose Number of Independent Elements is Larger than the Basis. International Journal of Science & Engineering Development Research (www.ijrti.org) | Volume 8, Issue 7 | ISSN: 2456-3315. Pg. 1001 - 1005.