Privacy-Preserving Hierarchical Deterministic Key Generation based on a Lattice of Rings in Public Blockchain

[1]Banupriya S, [2*]Kottilingam Kottursamy, [3,4]Ali Kashif Bashir
[1,2] School of Computing, SRM Institute of Science and Technology, Tamilnadu, India
[3]Department of Computing and Mathematics, Manchester Metropolitan University, UK
[4]School of Information and Communication Engineering, University of Electronics Science and Technology of China (UESTC), China
[1]bs9093@srmist.edu.in, [2]kottilik@srmist.edu.in, [3,4] dr.alikashif.b@ieee.org
*Corresponding Author

***Abstract*—Blockchain has revolutionized numerous fields, which include financial services, health care, the Internet of things, academia and supply chain management. Blockchain technology enables us to have an immutable, distributed ledger for managing the transactions of untrusted users. However, the technology has many open challenges, such as privacy leaks, scalability, and energy consumption. User identity can be easily tracked using network analysis, as transactions are accessible to everyone, which is a serious concern of blockchain. In this paper, we propose a new efficient, privacy-preserving, and quantum-resistant key generation algorithm, namely, lattice-based hierarchical deterministic key generation (LB-HDKG), for maintaining user privacy in the public blockchain. The LB-HDKG scheme generates many cryptographic keys in a tree-like structure from a single seed to hide the links between transactions of the same user. Our proposal uses the lattice NTRU cryptosystem, the security of which relies on the shortest vector problem (SVP) and closest vector problem (CVP) over the polynomial ring. Operations on the lattice NTRU cryptosystem are efficient and secure against classical computers and quantum computers. Security and performance analyses of our scheme show that the model is more secure and efficient and should replace current models to safeguard data from quantum computers.**

***Index Terms*—Blockchain, hierarchical deterministic, lattices, NTRU, privacy, quantum cryptography**

I. INTRODUCTION

Blockchain is a distributed database [1] that stores user transactions in a chain of blocks and can be implemented as a private blockchain or public blockchain. The private blockchain uses a centralized control system to restrict user entry with the help of access control management. The access control policy can be set to satisfy the requirements of user privacy protection. The public blockchain is an open network where anyone can enter and view the stored records and transactions [2]. There is no restriction in accessing the system, and user identity must be protected by hiding the links between transactions of the same user. The identity privacy requirement should be considered in public distributed systems such as Bitcoin, where the transactions are open to all [3]. In Bitcoin, the transactions are tied up with the hashed value of the user's cryptographic public key. Therefore, financial status can be easily tracked through the public address of a user. This can be avoided by including a new public key in every new transaction. Although this solves the privacy issue, it increases the overhead to the users due to the management of a huge number of key pairs.

Blockchain use cases extend beyond Bitcoin, e.g., to supply chain management, real estate, power grid management, the Internet of things, and government systems [4-8]. Numerous studies have improved the scalability, privacy protection, and interoperability and reduced the energy consumption of blockchain [9]. The hierarchical deterministic key generation method is proposed for resolving the privacy issue. It enables users to create any number of keys and simplifies the key management process [10]. Keys are derived as tree-like structures from a single secret seed. The master key is derived based on this secret input. Based on the user requirement, many child keys are derived from the derived master key. A node can derive any number of child nodes using the extended key or chain code. Moreover, the user can recover the entire key from a single input. The leaf nodes of the tree are used as a public address. The transactions are signed with the private key of the corresponding leaf node but not with the internal node's private key or chain code. The generation of multiple keys helps hide user identity and helps subdivide a financial account into multiple accounts for easy management, auditing, and ownership sharing with subordinate levels [11]. The centralized mixing scheme, decentralized mixing scheme, ring signature,

homomorphic encryption, and zero-knowledge proof are a few of the other solutions that have been proposed for resolving privacy issues of the public blockchain. These solutions depend on a centralized server/other user or a time-consuming process [12-16].

Quantum computers operate differently from traditional computers [17]. They work at the atomic level and can overcome the physical restrictions that affect traditional computer chips. Quantum computers use qubits instead of bits. A qubit can represent 0 & 1 simultaneously, and very few qubits can accelerate various types of computation by an enormous amount. Information that is encrypted with prime factorization and discrete logarithm-based algorithms could be easily read by anyone with access to a quantum computer [18]. The world must transform the traditional cryptography system to a quantum-resistant cryptographic procedure before quantum computers become available. Few cryptographic algorithms are available that rely on hash-based, code-based, lattice-based, and multivariate-based approaches [19-22]. These algorithms are comparatively safe against outbreaks from quantum computers.

In this paper, we propose a new efficient, privacy-preserving, and quantum-resistant lattice-based hierarchical deterministic key generation (LB-HDKG) algorithm for deriving child keys from a single master key. The lattice-based NTRU (Nth degree truncated polynomial ring) scheme is used for key generation algorithms, which are faster than RSA- and ECC-based algorithms. The method generates a new public key for each transaction, eliminates the overhead of storing and maintaining the multiple key pairs, and avoids the risk of key leakage. All the child keys are derived from the master key with only a single seed input.

The remainder of the paper is organized as follows. Section II details related studies on previously proposed solutions to privacy issues. Section III presents the preliminaries of lattice cryptosystems. The proposed LB-HDKG system model is presented in Section IV. Sections V and VI describe the security analysis and performance analysis, respectively, of the proposed model. Finally, the conclusions of the paper are presented in Section VII.

II. RELATED WORK

Hierarchical key management was first proposed for maintaining many keys based on a partial-ordered set [23]. Later, many approaches were proposed for hierarchical key management. Based on symmetric key encryption, which is not suitable for distributed systems, hierarchical key management services by a central authority (CA) are at risk of losing user privacy [24]. Other schemes have been proposed for generating a pairwise shared key using predistributed keys, which are suitable primarily for wireless sensor networks [25], [26]. In [27-30], significant hierarchical identity-based encryption schemes are proposed, among which those in [27] and [28] are based on Weil-pairing and the random oracle method and that in [29] is based on pairing without the random oracle method. Pairing with the elliptic curve encryption method is prone to quantum attacks. The lattice-based hierarchical identity-based encryption (HIBE) scheme is believed to be resistant to quantum computers but has a few pitfalls [30]. The binary tree structure is maintained for the hierarchy; hence, a parent node or master node is restricted to having only two child nodes. The depth of the tree is predefined, which limits the tree growth beyond the predefined depth. Child keys are created directly from the original parent, which necessitates the storage of key information of all tree nodes by either a user or a third party. Other schemes are based on polynomial interpolation, partial order, and Laplace noise injection [31-33]. All these schemes are well suited for user group management with the help of a third-party key management service (KMS) for identifying a user, assigning user access, and revoking access but are not suitable for multiple key management for a single user.

The BIP32 specification presented a solution for privacy issues by hierarchical deterministic (HD) key derivation for Bitcoin public electronic cash systems. The HD key generation method uses the SHA-512 hash function and elliptic curve cryptography (ECC) to derive the root node's key pair from the chosen seed. The parent key properties are combined with chain code to derive the child keys in further levels. Later, BIP44 was proposed for handling multiple cryptocurrencies, in which the address is derived via hierarchical deterministic derivation. Improvements of BIP32 are presented in [34], [35]. Although BIP32 has many advantages, it also has a few disadvantages. The system security is weak against quantum computers, as BIP32 relies on ECC. Single master private key leakage helps the attacker derive the child key pairs of all levels, and the attacker can spend funds using the derived private keys. The hierarchical deterministic key generation method has been implemented in various HD wallets in Ledger Nano X, Electrum, Mycelium, Trezor, KeepKey, Jaxx, and Atomic wallet.

Several other solutions have been proposed for identity privacy preservation, such as centralized mixing services, decentralized mixing services, ring signatures, and noninteractive zero-knowledge proofs. Centralized mixing services rely on a single central system and have the possibility for privacy leaks [36-38]. Users mix their public addresses using various decentralized mixing techniques without relying on an untrusted party [39-41]. Users depend on other users in a decentralized mixing procedure to complete the mixing, which delays the initiation

of the transaction. Stealth addresses facilitate the generation of a new address for every new transaction [42]. The generated address hides the receiver's identity but does not hide the sender's identity. In the homomorphic encryption method, computations are conducted on encrypted data without requiring access to a secret key [43]. The computing and storage costs for this approach are high. Pederson commitment prevents message leakage by using a random blinding factor and hides only the message content and not the sender or receiver identity [44]. Ring signatures were designed to hide the signer identity in group shared data [45]. Later, many modified versions of ring signatures were proposed, among which the most notable method is traceable ring signatures [46], [47]. The anonymity and likability properties of ring signatures are implemented in blockchain protocols to avoid identity privacy leakage [48], [49]. To generate a ring signature, a signer must wait until other users share their public keys. Each user depends on other users' availabilities. The noninteractive zero-knowledge proof technique is the best solution for privacy issues and is implemented in many cryptocurrency systems to avoid transaction graph analysis [50]. The zero-knowledge proof method requires additional computation time for proof generation. These solutions are time-consuming, as the user depends on an external system when using centralized mixing services and depends on other users in the case of decentralized mixing or ring signatures. Similarly, the zero-knowledge proof method requires more computation time for both signature generation and verification.

The ECC and RSA algorithms are popular cryptographic algorithms that are used for secure communication and data storage. Since the proposal of Shor's algorithm for quantum computation, recent studies have focused on lattice-based cryptography, since Shor's algorithm can find prime factors, and on discrete algorithms [51-54]. We propose a quantum-safe hierarchical deterministic key generation algorithm that is similar to BIP32 and is based on the lattice NTRU key generation scheme. NTRU key generation is faster than ECC and RSA and is resistant to quantum computers [55], [56].

III. PRELIMINARIES

A. Lattice

Lattice-based cryptography originates from recent progress in the field of quantum computing. The lattice is a linear algebraic structure in which the geometric grid of points extends infinitely in all directions. The identification of two lattice points that are relatively close together in the high dimensions of the grid is practically impossible and hard for both traditional and quantum computers. A lattice is defined formally as follows.
Lattice: Given *n* linearly independent vectors$b_1, b_2, \dots, b_n \in R_m$, their lattice is defined as

$$L\,(b_1, b_2, \dots, b_n) = x_i b_i / x_i \in Z$$

We refer $b_1, b_2, \dots, b_n$as a basis of the lattice. All possible weighted sums of those vectors are scaled by integers. Given the integer constraints and lattices, we can define problems that are hard to solve, as the hardness of factoring renders RSA highly secure. By finding hard problems that are easy to construct but hard to solve, we can develop a new method for public-key cryptography. No efficient algorithms are available in classical or quantum computers for solving lattice problems in better than exponential time. The shortest vector problem (SVS), closest vector problem (CVP), and covering radius are lattice-based hard problems that are used to secure public-key cryptosystems.

B. NTRU public key cryptosystem

NTRU is an efficient public-key cryptosystem that is based on a lattice. It is a faster key generation method than RSA and ECC. The NTRU cryptosystem provides long-term privacy, communication efficiency, and higher performance with sustainability. The NTRU cryptosystem is secure against quantum computers, as no quantum algorithm is available for breaking lattice-based cryptography systems. Operations of NTRU are constructed using the objects in a truncated polynomial ring$(R, +, \circledast)$,

$$R = Z[X]/(X^N - 1) \quad \dots \quad (1)$$

with polynomial degree of at most N-1: $a_0 + \ a_1 x^1 + \ a_2 x^2 + \cdots + \ a_{N-1} x^{N-1}$. The hard problem of NTRU is to define $h \ = \ f^{-1} \circledast \ g$, where f and g are ternary polynomials. Given $h \ \in \ R$, find 'short' f such that $f^{-1} \ \circledast \ g$ is also 'short'.

C. Notation and parameters

Parameters $N, p, q \in Z$ are defined, where N is the degree of a set of polynomials L with trinary coefficient T, which is denoted as $L \ \in \ T$.

$$L = \sum_{k=0}^{N-1} a_k x^k \ : \ a_k \in \{-1, 0, 1\} \textit{ for ternary} \quad \dots \quad (2)$$

Polynomial ring R = Z [X] / ($X^N - 1$) is a polynomial $Z[X]$ with modulo $X^N - 1$. The symbol $*$ denotes the polynomial multiplication operation. Polynomials f, g and $h \in R$ and f, g and $h \in T$ have highest degree $N - 1$. $f = [\, f_1, f_2, \ldots . f_N \,]$, $g = [\, g_1, g_2, \ldots . g_N \,]$ and $h = [\, h_1, h_2, \ldots . h_N \,]$ and f_p^{-1} denotes the polynomial inverse of f with modulo p.

$$f_p^{-1} = [\, (f_p^{-1})_1, (f_p^{-1})_2, \ldots . . (f_p^{-1})_N] \;\; : \;\; (f_p^{-1})_i \geq 0 \; and \; (f_p^{-1})_i < p$$

.

D. NTRU encryption and digital signature

Select the public parameters $N, p,$ and q and the secret polynomials f and g. Both p and q are prime numbers, q is a large modulus, p is a small modulus and $gcd(n, q)$ and $gcd(n, p)$ should be 1. The secret polynomials f and g are selected randomly from the set of polynomials L manually or using a polynomial generator. The chosen polynomials f and g are small relative to the value of N. Polynomial f is selected such that there exists inverses of $f \in R$ for both modulo p and modulo q. The extended Euclidean algorithm is used to check the existence of inverses for the chosen f. Since both random polynomials f and g are "short", and the product of their result is also "short". The inverse of f results in randomization, and multiplication with g makes the result h more random. Therefore, given the public value h, it is difficult to find secret values f and g. NTRU key generation, encryption, and decryption algorithms are presented as Algorithms 1, 2, and 3, respectively. Algorithm 1 accepts secret polynomials and $N, p,$ and q as input and generates the public key. Algorithm 2 accepts the public key and input message as input and outputs the encrypted message. Algorithm 3 accepts the secret polynomial f and the encrypted message as input and decrypts the message. The NTRUSign key generation, digital signature generation, and verification algorithms are presented as Algorithms 4, 5, and 6, respectively. Algorithm 4 accepts secret polynomials and parameters as input and generates the public key. Algorithm 5 accepts private polynomial f and a message as input and generates the signature. Algorithm 6 accepts the public key, message, and signature s as input and returns the verification status.

Algorithm 1: $Key\ generation$ (NTRUEncrypt_KeyGen)

Input: Secret polynomials f and g, parameters $N, p,$ and q
Output: Public key h

1. Find the inverse of f (f_p^{-1}) using the extended Euclidean algorithm.
2. Generate public key h by, h = $f_p^{-1} \circledast$ g (mod q).

Algorithm 2: $Encryption$

Input: public key h, message $m \in L, m \in T$ in the form $m = [m_1, m_2, \ldots . m_N] \; : \; m_i \geq 0 \; and \; m_i < q$
Output: encrypted message $e(m)$

1. Choose a random polynomial $r \in R \; where \; r = [r_1, r_2, \ldots r_N] \in T$.
2. Encrypt the message m by, $e(m) = r \circledast h + m \; (mod \; q)$.

Algorithm 3: $Decryption$

Input: Secret polynomial f, encrypted message $e(m)$
Output: Message m

1. Compute $r1 = f \circledast e(m) \; (mod \; q)$.
2. Compute $r2 = r1 \; (mod \; p)$.
3. Compute $m = f_p^{-1} \circledast r2 \; (mod \; p)$.

Algorithm 4: $Key\ generation$ (NTRUSign_KeyGen)

Input: Secret polynomials f and g, parameters N and q
Output: Public key h

1. Find polynomials F and G such that $f * G - g * F = q$.
2. Find the inverse of f (f_p^{-1}) using the extended Euclidean algorithm.
3. Generate public key h by, h = F $* \; f_q^{-1}$ (mod q).

Algorithm 5: $Signature\ Generation$

Input: private key f, message m ∈ L, m ∈ T in the form $m = [m_1, m_2, \ldots . m_N] \ : \ m_i \geq 0 \ and \ m_i < q$
Output: Signature s

1. Map the input message to the hash function to obtain m'.
2. Set $x = \left(-\frac{1}{q}\right) m' * g \ and \ y = \left(\frac{1}{q}\right) m' * f$.
3. Set $e = -\{x\} \ and \ e' = -\{y\}$.
4. Compute signature s as $s = ef + e'g$.

Algorithm 6: $Verification$

Input: Public key h, message m, signature s, balancing vector b and norm bound n
Output: Status 0 or 1

1. Map the input message to the hash function to obtain m'.
2. Compute $t = (s * h) \ mod \ q$.
3. Calculate Norm v = min ($\| s + k_1 q$, ($s * h - m$) + $k_2 q \|$) for $k_1, k_2 \in R$.
4. If $v \leq n$ set the status as 0 (valid); otherwise, set it to 1 (invalid).

IV. SYSTEM MODEL

The hierarchical key derivation algorithm is applied to the end-user side and does not require any modification to the other users in the network. The seed value is a generic English sentence that can be remembered easily without storage in any electronic device. The polynomial generator accepts the seed value input and generates the secret polynomials f and g for each user with a highest polynomial degree of $N - 1$. $hf_l \ and \ hf_r$ denote the left and right half values of hf. $hg_l \ and \ hg_r$ denote the left and right half values of hg.$hcg_l \ and \ hcg_r$ denote the left and right half values of hcg. $hcf_l \ and \ hcf_r$denote the left and right half values of hcf.

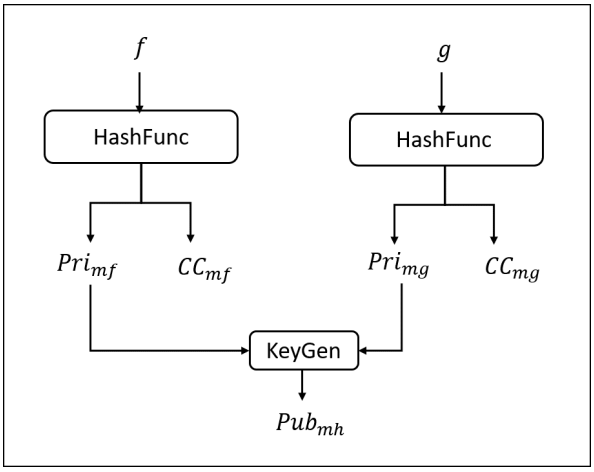

Fig. 1. Master key generation

To create a key, the user chooses the public parameters $N, p,$ and q and generates polynomials $f \ and \ g$ from the seed phrase. The master keys and chain code are generated using $Master\_KeyGen$ which is presented as Algorithm 7. The master key generation algorithm computes the master private key values Pri_{mf} and Pri_{mg}, the master public key value Pub_{mh}, and the master chain code values CC_{mf} and CC_{mg}. Fig. 1 illustrates the master key generation process. The generated keys and chain codes are

$$Master \ private \ keys \ (f \ and \ g) \ \rightarrow \ Pri_{mf}, Pri_{mg}$$

$$Master \ chain \ code \ (f \ and \ g) \ \rightarrow \ CC_{mf}, CC_{mg}$$

$$Master \ public \ key \ \rightarrow \ Pub_{mh}$$

The child keys and chain code values are generated using $Child\_KeyGen$, which is presented as Algorithm 8, by sending master key input(Pri_{mf}, $Pri_{mg}, CC_{mf}, CC_{mg}$)/parent key input($Pri_{cf}$, $Pri_{cg}, CC_{cf}, CC_{cg}$) and the index number of child node i The child key generation algorithm computes the child private key valuesPri_{cf} and

Pri_{cg}, the child public key value Pub_{ch}, the child chain code values CC_{cf} and the child chain code CC_{cg}. Fig. 2 illustrates the child key generation process. The generated keys and chain codes are

$$Child\ private\ keys\ (f\ and\ g) \rightarrow\ Pri_{cf}, Pri_{cg}$$

$$Child\ chain\ code\ (f\ and\ g) \rightarrow\ CC_{cf}, CC_{cg}$$

$$Child\ public\ key\ \rightarrow\ Pub_{ch}$$

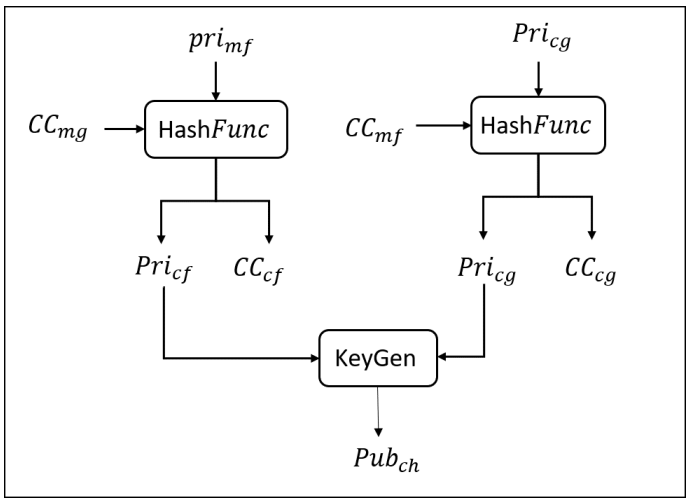

Fig. 2. Child key generation

Algorithm 7: *Master_KeyGen*

Input: Polynomials $f\ and\ g$ and parameters $N, p\ and\ q$
Output:Pri_{mf}, Pri_{mg}, CC_{mf}, CC_{mg} and Pub_{mh}

1. Reset $pbit$ and initialize it to 0.
2. Compute hf = HashFunc ($f\ ||\ pbit$) and split hf into $hf_l\ and\ hf_r$.
3. Check that hf_l is invertible, $\gcd(hf_l, p) = 1$ and $\gcd(hf_l, q) = 1$; if any condition fails then increment $pbit$ and repeat step 2.
4. Assign $Pri_{mf} = hf_l$.
5. Assign $CC_{mf} = hf_r$.
6. Reset $pbit$ and initialize it to 0.
7. Compute hg = HashFunc ($g\ ||\ pbit$) and split gf into $gf_l\ and\ gf_r$.
8. Check that hg_l is invertible, $\gcd(hg_l, p) = 1$ and $\gcd(hg_{l_l}, q) = 1$; if any condition fails then increment $pbit$ and repeat step 7.
9. Assign $Pri_{mg} = hg_l$.
10. Assign $CC_{mg} = hg_r$.
11. Invoke key generation algorithm $Pub_{mh} = KeyGen(Pri_{mf}, Pri_{mg}, N, p, q)$.

Algorithm 8: *Child_KeyGen*

Input: Pri_{mf}, Pri_{mg}, CC_{mf}, CC_{mg} and child index i
Output: Pri_{cf}, Pri_{cg}, CC_{cf}, CC_{cg} and Pub_{ch}

1. Reset $pbit$ and initialize it to 0.
2. Compute hcf = HashFunc ($pri_{mf}\ ||\ CC_{mg}\ ||\ i\ ||\ pbit$) and split hcf into $hcf_l\ and\ hcf_r$.
3. Check that hcf_l is invertible, $\gcd(hcf_l, p) = 1$ and $\gcd(hcf_l, q) = 1$; if any condition fails then increment $pbit$ and repeat step 2.
4. Assign $Pri_{cf} = hcf_l$.
5. Assign $CC_{cf} = hcf_r$.
6. Reset $pbit$ and initialize it to 0.
7. Compute hcg = HashFunc ($pri_{mg}\ ||\ CC_{mf}\ ||\ i\ ||\ pbit$) and split hcg into $hcg_l\ and\ hcg_r$.

8. Check that hcg_l is invertible, $\gcd(hcg_{l_l}, p) = 1$ and $\gcd(hcg_{l_l}, q) = 1$; if any condition fails then increment $pbit$ and repeat step 7.
9. Assign $Pri_{cg} = hcg_l$.
10. Assign $CC_{cg} = hcg_r$.
11. Invoke key generation algorithm $Pub_{ch} = NTRUEncrypt\_KeyGen\ (or)\ NTRUSign\_KeyGen.$

$Master\_KeyGen$ is invoked only for the root node, and $Child\_KeyGen$ is invoked for the internal and leaf nodes. The chain code is used to break the link from the master key and child key. The polynomial private keys (f and g) that are generated from $Master\_KeyGen$ and $Child\_KeyGen$ are evaluated with the extended Euclidean algorithm for invertibility. The extended Euclidian algorithm determines the greatest common divisor (GCD) of two polynomials and returns the GCD value. For ternary polynomials, the input trits are converted into binary values before invoking the hash function. The SHA256 algorithm is used to find a hash value in the key generation process based on the EESS recommendation [57]. The lengths of the input random polynomials vary, and the output of SHA256 is always 256 bits. The binary value of each octet (8 bits) is converted into its equivalent ternary value with a length of 6 trits. Therefore, the hashed output length is 192 trits, and polynomials f and g are each 96 trits long. Similarly, the index value i is converted into a ternary value before invoking the key generation function.

The child index number is represented as $(r/1, r/2, r/3 \ldots r/k)$ when depth = 2 or level = 1, where r represents the root node and $1,2,3 \ldots k$ represent children of the root node (depth = 1 or level = 0), as shown in Fig. 3. Index numbers $(r/1/1)$ and $(r/1/2)$ represent the 1st and 2nd children, respectively, from a parent $(r/1)$. The depth of a tree can grow according to user requirements. The child index number is encoded into trinary values before it is fed into HashFunc. If the user requests the generation of a child key $(r/2/1)$, then the $Master\_KeyGen$ algorithm is invoked one time to derive (r) from random polynomials. $Child\_KeyGen$ is called twice: first, to derive $(r/2)$ from (r) and second, to derive $(r/2/1)$ from $(r/2)$. Out of these three derivations, the public key is derived only for a child (r/2/1) and not for r or r/2. By setting req_depth to 3 (req_depth = 3), the $Master\_KeyGen$ and $Child\_KeyGen$ algorithms ignore the public key generation process for (r) and $(r/2)$ nodes. Similarly, if the user requests $(r/1/1/1)$ node keys, req_depth is set to 4 (req_depth = 4), and the $Master\_KeyGen$ and $Child\_KeyGen$ algorithms ignore the public key generation process for nodes (r), $(r/1)$ and $(r/1/1)$ and generate a public key only for $(r/1/1/1)$.

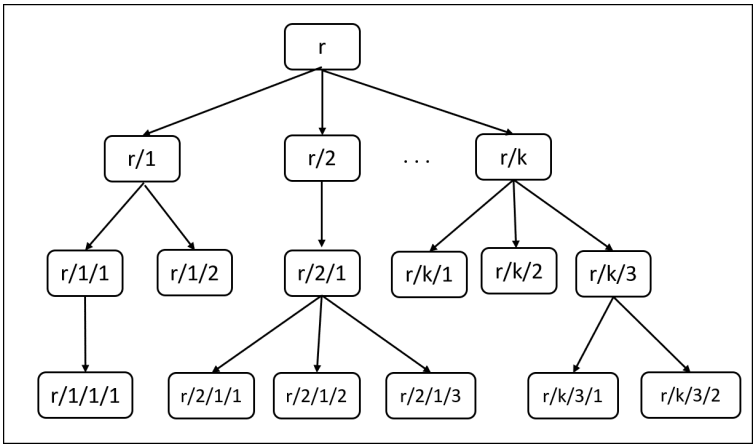

Fig. 3. Node index numbers

The advantages of the proposed LB-HDKG scheme are as follows: 1) User privacy is guaranteed by linking new public addresses in every incoming transaction. 2) Chain code is used to derive child keys and hide the properties of the private key from lower-level keys for increased security. 3) A seed phrase alone is required as input to derive the whole tree structure. 4) Users need not worry about key damage, loss, or storage management. 5) Lower-level public keys can be shared with trustless third-parties to maintain subtree accounts. 6) The scheme is secure against quantum computing. 7) It does not depend on any external system or user for key generation. 8) It is more efficient than the RSA and ECC algorithms.

V. SECURITY ANALYSIS

The security of the LB-HDKG system is evaluated based on the chosen N value. The recommended value of N is above 200 for moderate-level security and above 500 for high-level security. A system with a security level

of 80 requires approximately 10 to the power of 12 MIPS-years to break. Selecting large and relative prime values for $p\ and\ q$ increases the security level of the system [55], [56].

Using the combinatorial method, the attacker can recover private keys f and g from public value h or random value r. Similarly, the attacker attempts to recover message m from encrypted message e since the secret polynomials f and g fall under space $N * N$. The security against the combinatorial technique for binary polynomials is greater than or equal to $\frac{\binom{N/2}{d/2}}{\sqrt{N}}$. The security against the combinatorial technique for ternary polynomials is greater than equal to $\binom{N/2}{d/2} / \sqrt{N}$. Private values f or g can be recovered from public value h or e using the combinational technique when the correct parameters are not chosen.

If the private key $= 1 + pF$ (F is a small polynomial) or $f = 1\ (mod\ p)$, then the lattice attack on f is equivalent to SVP. SVP is NP-hard under the randomized reduction hypothesis. If the private key satisfies $f\ != 1 + pF$, then a lattice attack on f is equivalent to CVP and is as hard as SVP. The security level of this $f = 1 + pF$ form can be represented with $a = N/q$ and c = $\sqrt{(4\pi e\ ||\ f\ ||\ ||\ g||\ q)}$. The breaking time increases when increasing the value of c with constant (a, N), and the breaking time increases exponentially when increasing the value of N with constant (a, c).

Secret message m is encrypted with random polynomial r and public polynomial h in Algorithm 2.

$$e(m) = r\ \circledast\ h\ + m\ (mod\ q)\ \ \ldots (3)$$

The attacker knows the values of h and $e(m)$ and tries to recover m with a different combination of random values r. The attack can be avoided by choosing a sufficiently large m by adding extra padding bits. For better security, the parameter d (as small as possible) is set to have a minimum number of 1's in trinary polynomials f and m.

A. Child private key recovery from the parent's private key

Loss of the parent's private key does not compromise the remainder of the tree. The private key is used for digital signatures and to transfer the ownership of financial holds in the blockchain. The private key can be revealed to the attacker by any means, but the chain code is never revealed to the outside for any purpose. The attacker cannot derive the child keys from the private key alone. The chain code helps regenerate the child's private key in the future whenever the user transfers ownership and helps avoid compromising the remainder of the tree or the child's private keys.

B. Private key recovery from a public key

The security of LB-HDKG depends on the recovery of shorter master private keys (f and g) from a public basis B of the master lattice:

$$B = \begin{bmatrix} qI_N & 0_N \\ H & I_N \end{bmatrix}$$

where I_N is an identity matrix with $N * N$ dimensions, 0_N is a zero matrix with $N * N$ dimensions and H is a circular matrix that corresponds to public key h. The attacker attempts to locate the shortest vector on a public basis B that corresponds to secret values $f\ and\ g$. The attacker chooses N' less than N and builds B' with N' and H'.

$$B' = \begin{bmatrix} qI_{N'} & 0_N \\ H' & I_{N'} \end{bmatrix}$$

where H' is a truncated matrix of H. The attacker removes matrix B' from the center of matrix B. The attacker repeats the process until all N'of q vectors are removed. Greater than k' bits of effort can be used to reduce the original lattice B'. Optimally choosing N' and k' for reduction, the problem is equivalent to a meet-in-the-middle search. Finding N' and k' is hard, and the master private key cannot be recovered from a public basis B . Choosing relatively small f and g compared to N is important to avoid recovery of the private key from the public key. In our scheme, the hashed output size is reduced to ½ for both polynomials f and g. For a security level of $k = 80$, the parameter N is chosen as a prime number that exceeds $3k(N > 3k)$. When $N = 251$, the sizes of f and g needed to be less than 251. The output of hash function SHA256 is split into 2 halves as the master private key (128 bits) and the chain code (128 bits). In the case of ternary polynomials, each octet (8 bits) is converted into 6 trits. The 256 bit (32 octets) output of SHA256 is reduced to 192 trits (32 octets * 6 trits). By splitting the 192 converted trits into 2 halves, we obtain 96 trits for private keys and 96 trits for the chain code. Therefore, the sizes

of f and g are always less than 251 or smaller than N . As a result, it is infeasible to recover smaller f and g from h_i using the reduction method.

C. Master/parent key recovery from the child keys

The security of the master key relies on the difficulty of learning short polynomials pri_{mf} and pri_{mg} from the child keys. The child node's private key is derived from a partial value of the master private key. The private keys are derived after hashing the master private key to avoid upstream key exposure or master key recovery. In the hierarchical derivation, upstream key exposure is a way of obtaining knowledge of parent key values from the known child key values. Two chain codes for each polynomial f and g are used to eliminate the relation between the master private keys and child keys. Child private key $\mathrm{pri_{cf}}$ is derived from the hashed value of master private key $\mathrm{pri_{mg}}$, and master chain code $\mathrm{CC_{mg}}$. Child private key $\mathrm{pri_{cg}}$ is derived from the hashed value of master private key $\mathrm{pri_{mf}}$ and master chain code CC_{mg}. The master keys are completely masked in child key derivation. Therefore, the attacker cannot obtain any knowledge from one secret value f or g Decryption failure can be avoided by adding extra bits with input for padding [58]. Various attacks of NTRU, such as hybrid attacks, ternary linear cryptanalysis, meet-in-the-middle attacks, and invariant attacks, are avoided by choosing the correct parameters [59-61]. Therefore, the security of the system depends solely on the hash function and parameter selection [62]. Table I presents the security level of NTRU with basic NTRU parameters [63] and the computation times for key generation, encryption, and decryption. Table II presents the security level of NTRU with basic NTRU parameters and the computation times of signature generation and verification. Few researchers have also proposed algorithms to restrict attacks in the physical layer [64].

Table I

Computation times for key generation, encryption, and decryption

N	p	q	Security level (bits)	Key generation (msec)	Encryption (msec)	Decryption (msec)
251	2	197	80	62.2	4.23	21.53
347	2	269	112	146.18	6.72	33.3
397	2	307	128	192	9.11	49.37
587	2	439	192	418.51	13.09	79.21
787	2	587	256	767	23.38	10.64

Table II

Computation times for signature generation and verification

N	q	Security level (bits)	Signature generation (msec)	Signature verification (msec)
157	256	80	3.17	1.567
197	256	112	4.56	2.78
223	256	128	5.78	3.52
313	512	192	14.98	7.91
349	512	256	26.8	10.786

VI. PERFORMANCE ANALYSIS

In this section, we present a detailed comparison of the computation times for the key generation, encryption/decryption, and signature/verification processes of ECC and NTRU. Then, we compare BIP32 and LB-HDKG and present the computation times of our proposed LB-HDKG master key generation and child key

generation processes. The performance of the system is examined on a 2.30 GHz Intel Core operating system under Windows. Unoptimized Python code is implemented for ECC and NTRU key generation, encryption/decryption, signature/verification, and LB-HDKG. The computation times include the initialization of parameters, hashing, primitive polynomial operations for the extended Euclidean algorithm, encoding, and key generation process.

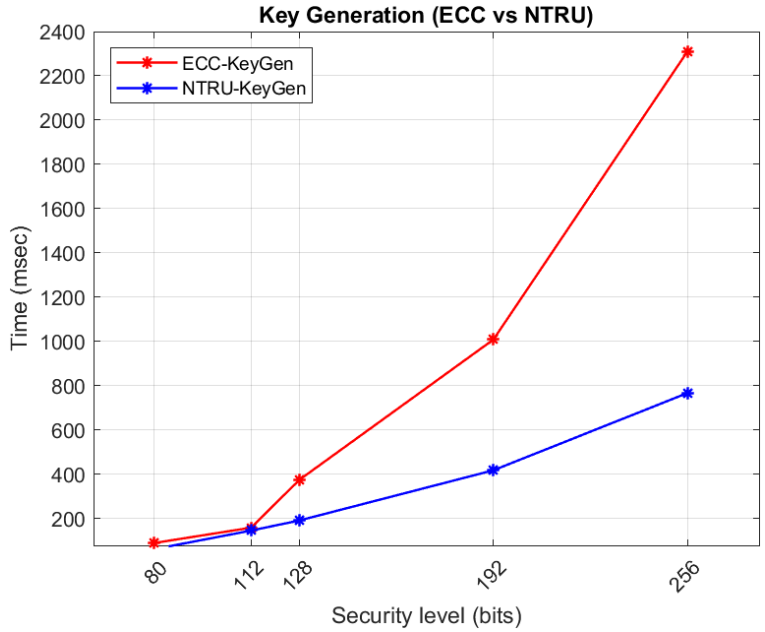

Fig. 4. Comparison of computation time for key generation between ECC and NTRU

The sizes of the private key and public keys are approximately equal in ECC and extremely different in NTRU. The value of N is chosen based on the requirement of security level k namely, N should exceed $3k$.

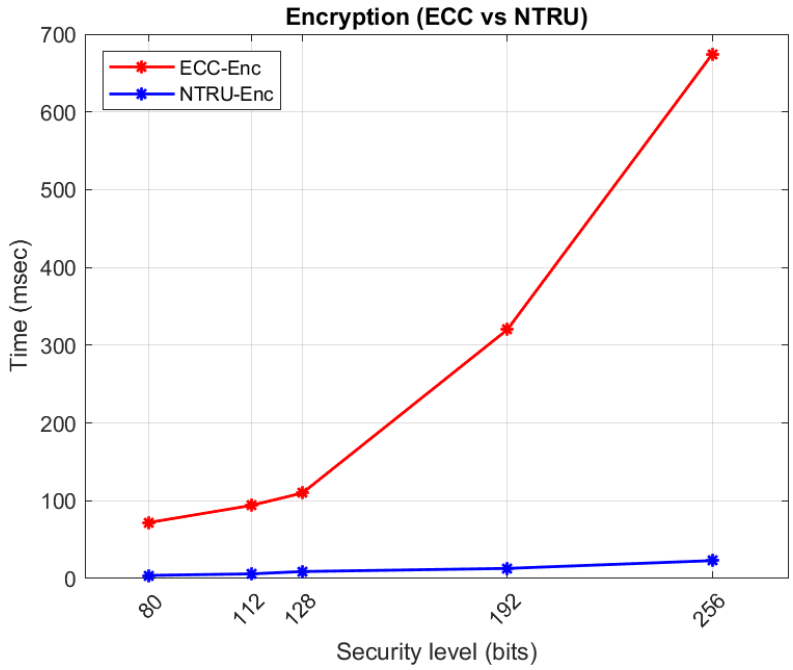

Fig. 5. Comparison of computation time for encryption between ECC and NTRU

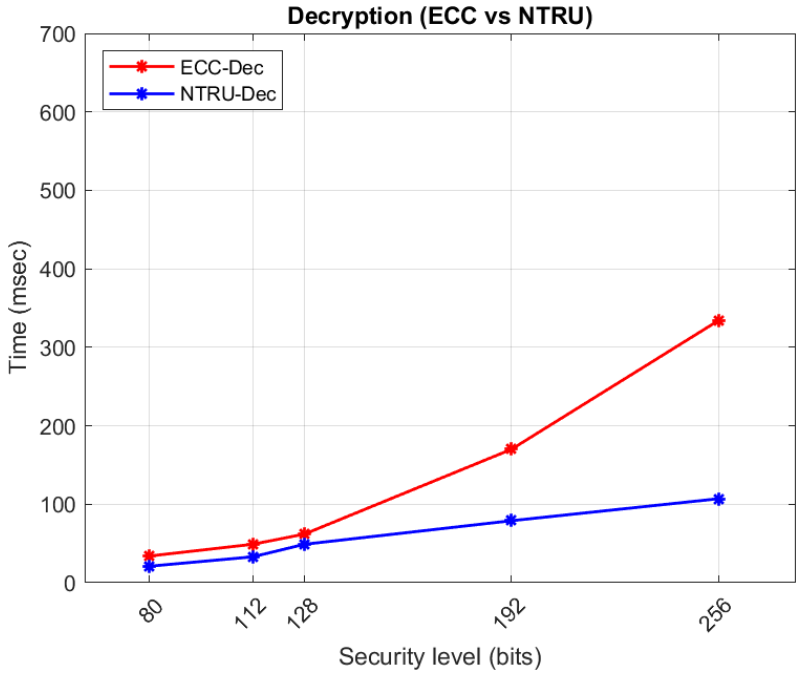

Fig. 6. Comparison of the computation time for decryption between ECC and NTRU

The public key size depends on the parameter N that is, the size of h is $N/(N-k)\ log_p q - to - 1$ [57]. Fig. 4. illustrates the comparison of computation time (msec) of ECC and NTRU key generation for security levels 8, 112, 128, 192 and 256. Fig. 5 illustrates the comparison of computation time (msec) of ECC and NTRU encryption, and Fig. 6 illustrates the comparison of computation time (msec) of ECC and NTRU decryption for various security levels 8, 112, 128, 192 and 256.

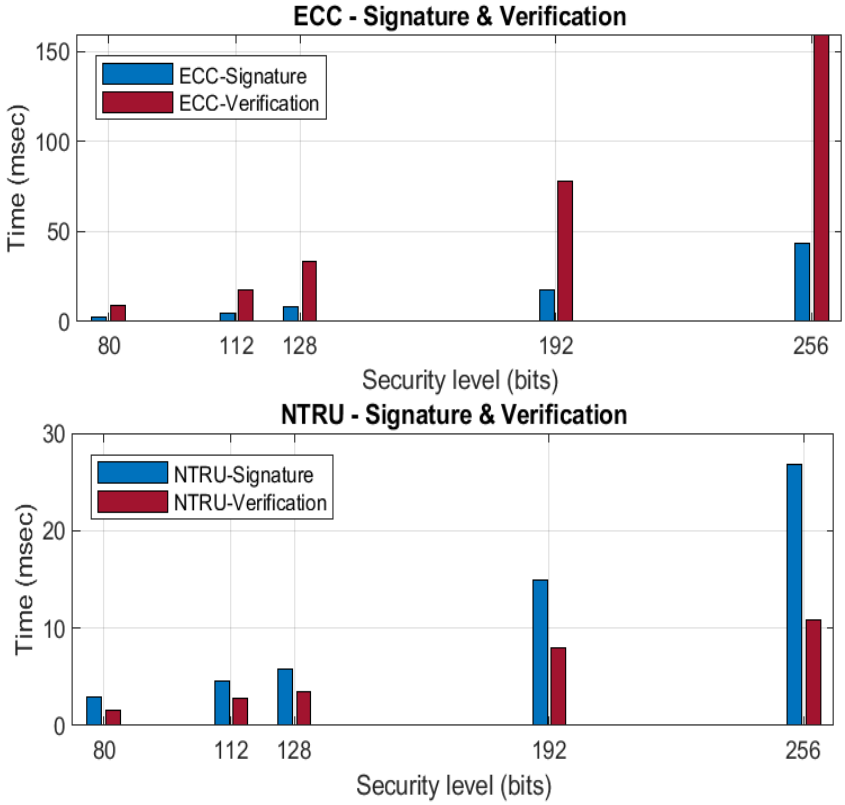

Fig. 7. Comparison of the computation times for ECC signature and verification with those for NTRU signature and verification

The computation times for ECC, NTRU signature, and verification are compared in Fig. 7. The figure presents the computation times for signature/verification for security levels 8, 112, 128, 192 and 256. The results

demonstrate that NTRU is faster than ECC for all security levels. Table III compares BIP32 with the proposed LB-HDKG scheme.

Table III
Comparison of hierarchical key generation schemes

Characteristics	**BIP32**	**LB-HDKG**
Approach	Asymmetric	Asymmetric
Based on	Elliptic curve key generation	Lattice-based NTRU
Public key size	Small	Large
Complexity of public key generation	$O(N^4)$	$O(Nlog\ N)$
Mathematical problem	Elliptic curve discrete logarithm	Short vector problem/closest vector problem
Operations	Scalar point addition and multiplication	Polynomial multiplication over a ring
Security	High	Very high
Quantum Resistance	No	Yes
Advantages	Short public key	High speed, high security, and quantum resistance
Disadvantages	Requires more computation time and breakable by quantum computers	Large public key size and security depends on parameter selection

The minimum computation time for the master key generation process is approximated using Eqn. (4). Equations (5), (6), and (7) are used to approximate the minimum computation time for child key generation, where $\mathrm{M_t}$ represents the minimum time for master key generation, $\mathrm{Ck_t}$ represents the minimum time for child key generation at level m, h_t represents the time for generating the hash, K_t represents the time for generating public key h from secret polynomials f and g, P_t represents the time for polynomial generation from the seed, and $\mathrm{E_t}$ represents the time for the encoding (ternary to binary) and decoding (binary to ternary) process. The approximate minimum time for the key generation process can be expressed as

$$M_t \simeq (\ 2\ *\ h_t\ ) + K_t + P_t \qquad \ldots (4)$$

$$C1_t \simeq (2\ *\ h_t) + (\ 2\ *\ h_t) + K_t\ +\ P_t +\ E_t \qquad \ldots (5)$$

$$C2_t \simeq\ 2\ *\ h_t + (\ 2\ *\ h_t) + (\ 2\ *\ h_t) +\ K_t + P_t +\ E_t\ \ldots (6)$$

From (5) and (6), the general form for child key generation can be expressed as

$$Cm_t \simeq\ \ (m + 1\ ) * (\ 2\ *\ h_t) + K_t + P_t +\ E_t \qquad \ldots (7)$$

For the same set of parameters $N, p,$ and q, the computation time for master key generation differs based on the seed value. Similarly, time variance may be possible for child nodes at the same level since the polynomials f and g are obtained from the hashed output. If the polynomial f is not invertible, the private key is recomputed by adding extra padding bits to the input of the hashing function. The complexity of public key generation from the private key and chain code is $O(N\ log\ N)$ for N-degree polynomials without consideration of the padding complexity.

Table IV

LB-HDKG master key generation and child key generation

N	Master key generation (msec)	Child key generation (Level 1) (msec)	Child key generation (Level 2) (msec)	Child key generation (Level 3) (msec)
11	98.2	101.53	120.62	129.31
53	104.62	111.31	124.61	136.32
251	159.27	178.98	199.22	203.84
347	281.61	293.81	305.42	306.71
449	352.22	378.44	395.43	410.82

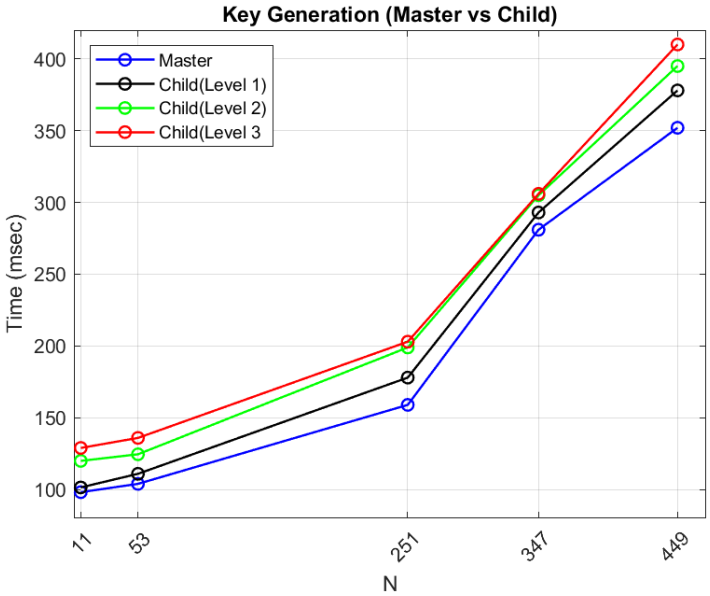

Fig. 8. Comparison of LB-HDKG master key generation and LB-HDKG child key generation

Table IV presents the computation times of the LB-HDKG master key generation and child key generation algorithms. Fig. 8 presents the measured execution times for master key generation and child key generation for various values of N. The child key generation process includes the time that is needed to find a child position based on the index and path for the child key. The times that is required is approximately equal for all children at the same level. A smaller variance of time at the same level may occur due to the noninvertibility of f from the hashing output. The results demonstrate that increasing the tree level will not increase the key generation time. The public key is not derived for all parent/master nodes; it is derived only for the child node with the index that is passed in the input. We can omit the public key derivation process from the root to the parent of the child node. Additionally, if the user provides the original parent's private keys along with the new child index, the keys are derived from the original parent instead of the root. The presented result includes the time that is required for deriving keys from the root node to the child node.

VII. CONCLUSIONS

Blockchain has been considered a promising technology for adaption from a centralized system to an immutable decentralized system. To increase user privacy and reduce energy consumption, we proposed an efficient, privacy-preserving, and quantum-resistant hierarchical deterministic key generation algorithm. The

proposed hierarchical key generation algorithm uses a lattice-based NTRU cryptosystem and efficiently generates numerous keys to avoid user privacy leaks without key management overhead. The lattice-based key generation algorithm is efficient, quantum-safe, and highly suitable for distributed systems. The security analysis of our scheme shows that upstream key exposure is not possible, and the performance analysis shows that the computation time for the proposed scheme is less than those of RSA and ECC. The proposed model can be applied to distributed systems, public storage systems, and cloud storage systems, among other systems. However, parameter selection is considered to be a crucial factor when the NTRU cryptosystem is implemented in real time.

REFERENCES

[1] Nakamoto, S., 2019. Bitcoin: A peer-to-peer electronic cash system. Manubot.

[2] Pilkington, M., 2016. Blockchain technology: principles and applications. *In Research handbook on digital transformations*. Edward Elgar Publishing.

[3] Underwood, S., 2016. Blockchain beyond bitcoin.

[4] Saberi, S., Kouhizadeh, M., Sarkis, J. and Shen, L., 2019. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57(7), pp.2117-2135.

[5] Karamitsos, I., Papadaki, M. and Al Barghuthi, N.B., 2018. Design of the blockchain smart contract: A use case for real estate. *Journal of Information Security*, 9(3), pp.177-190.

[6] Mengelkamp, E., Notheisen, B., Beer, C., Dauer, D. and Weinhardt, C., 2018. A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science-Research and Development*, 33(1-2), pp.207-214.

[7] Lin, X., Wu, J., Bashir, A.K., Li, J., Yang, W. and Piran, J., 2020. Blockchain-Based Incentive Energy-Knowledge Trading in IoT: Joint Power Transfer and AI Design. IEEE Internet of Things Journal.

[8] Ølnes, S., Ubacht, J. and Janssen, M., 2017. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing.

[9] Yli-Huumo, J., Ko, D., Choi, S., Park, S. and Smolander, K., 2016. Where is current research on blockchain technology?—a systematic review. PloS one, 11(10), p.e0163477.

[10] Wuille, P., 2012. Bip32: Hierarchical deterministic wallets. h ttps://github. com/genjix/bips/blob/master/bip-0032. md.

[11] Eskandari, S., Clark, J., Barrera, D. and Stobert, E., 2018. A first look at the usability of bitcoin key management. arXiv preprint arXiv:1802.04351.

[12] Zhang, R., Xue, R. and Liu, L., 2019. Security and privacy on blockchain. *ACM Computing Surveys* (CSUR), 52(3), pp.1-34.

[13] Shi, N., Tan, L., Li, W., Qi, X. and Yu, K., 2020. A blockchain-empowered AAA scheme in the large-scale HetNet. *Digital Communications and Networks*.

[14] Yu, K., Tan, L., Shang, X., Huang, J., Srivastava, G. and Chatterjee, P., 2020. Efficient and Privacy-Preserving Medical Research Support Platform Against COVID-19: A Blockchain-Based Approach. *IEEE Consumer Electronics Magazine*.

[15] J. Zhang, K. Yu, Z. Wen, X. Qi and A. K. Paul, 2021. 3d reconstruction for motion blurred images using deep learning-based intelligent systems. *Computers, Materials & Continua*, 66(2), pp. 2087–2104.

[16] Yu, K.P., Tan, L., Aloqaily, M., Yang, H. and Jararweh, Y., 2021. Blockchain-enhanced data sharing with traceable and direct revocation in IIoT. *IEEE Transactions on Industrial Informatics*.

[17] Deutsch, D., 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. Proceedings of the Royal Society of London. *A. Mathematical and Physical Sciences*, 400(1818), pp.97-117.

[18] Bernstein, D.J., 2009. Introduction to post-quantum cryptography. *In Post-quantum cryptography* (pp. 1-14). Springer, Berlin, Heidelberg.

[19] Buchmann, Johannes; Dahmen, Erik; Hülsing, Andreas (2011). "XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions". *Lecture Notes in Computer Science*. 7071 (Post-Quantum Cryptography. PQCrypto 2011): 117–129.

[20] Overbeck, R. and Sendrier, N., 2009. Code-based cryptography. *In Post-quantum cryptography* (pp. 95-145). Springer, Berlin, Heidelberg.

[21] Kottursamy, K., Raja, G., Padmanabhan, J. and Srinivasan, V., 2017. An improved database synchronization mechanism for mobile data using software-defined networking control. Computers & Electrical Engineering, 57, pp.93-103.

[22] Ding, J. and Schmidt, D., 2005, June. Rainbow, a new multivariable polynomial signature scheme. *In International Conference on Applied Cryptography and Network Security* (pp. 164-175). Springer, Berlin, Heidelberg.

[23] Arul, R., Raja, G., Kottursamy, K., Sathiyanarayanan, P. and Venkatraman, S., 2017. User path prediction based key caching and authentication mechanism for broadband wireless networks. Wireless Personal Communications, 94(4), pp.2645-2664

[24] MacKinnon, S.J., Taylor, P.D., Meijer, H. and Akl, S.G., 1985. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Transactions on Computers*, (9), pp.797-802.

[25] Lin, J.C., Huang, K.H., Lai, F. and Lee, H.C., 2009. Secure and efficient group key management with shared key derivation. *Computer Standards & Interfaces*, 31(1), pp.192-208.

[26] Arul, R., Raja, G., Almagrabi, A.O., Alkatheiri, M.S., Chauhdary, S.H. and Bashir, A.K., 2019. A Quantum-Safe Key Hierarchy and Dynamic Security Association for LTE/SAE in 5G Scenario. IEEE Transactions on Industrial Informatics, 16(1), pp.681-690.

[27] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. *In ASIACRYPT*, pages 548–566. 2002.

[28] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. *In EUROCRYPT*, pages 466–481. 2002.

[29] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). *In CRYPTO*, pages 290–307. 2006.

[30] Katsumata, S., Matsuda, T. and Takayasu, A., 2020. Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance. *Theoretical Computer Science*, 809, pp.103-136.

[31] Shen, V.R. and Chen, T.S., 2002. A novel key management scheme based on discrete logarithms and polynomial interpolations. Computers & Security, 21(2), pp.164-171.

[32] Das, M.L., Saxena, A., Gulati, V.P. and Phatak, D.B., 2005. Hierarchical key management scheme using polynomial interpolation. ACM SIGOPS Operating Systems Review, 39(1), pp.40-47.

[33] Wang, T., Zheng, Z., Bashir, A.K., Jolfaei, A. and Xu, Y., 2020. FinPrivacy: A Privacy-Preserving Mechanismfor Fingerprint Identification. ACM Transactions on Internet Technology (TOIT).

[34] Gutoski, G. and Stebila, D., 2015, January. Hierarchical deterministic bitcoin wallets that tolerate key leakage. *In International Conference on Financial Cryptography and Data Security* (pp. 497-504). Springer, Berlin, Heidelberg.

[35] Khovratovich, D. and Law, J., 2017, April. BIP32-Ed25519: Hierarchical Deterministic Keys over a Non-linear Keyspace. *In 2017 IEEE European Symposium on Security and Privacy Workshops* (EuroS&PW) (pp. 27-31). IEEE.

[36] Heilman, E., Baldimtsi, F. and Goldberg, S., 2016, February. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In *International conference on financial cryptography and data security* (pp. 43-60). Springer, Berlin, Heidelberg.

[37] Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A. and Goldberg, S., 2017. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *Network and Distributed System Security Symposium*.

[38] Knirsch, F., Unterweger, A. and Engel, D., 2018. Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions. *Computer Science-Research and Development*, *33*(1-2), pp.71-79.

[39] Maxwell, G. 2013. Coinjoin: Bitcoin privacy for the real world. In Post on Bitcoin Forum.

[40] Bissias, G., Ozisik, A.P., Levine, B.N. and Liberatore, M.2014.Sybilresistant mixing for bitcoin. In *The Workshop on Privacy in the Electronic Society*, pp. 149–158.

[41] Ruffing T., Moreno-Sanchez P., Kate A. (2014) CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. *In: Kutyłowski M., Vaidya J. (eds) Computer Security - ESORICS 2014. ESORICS 2014. Lecture Notes in Computer Science*, vol 8713. Springer, Cham. https://doi.org/10.1007/978-3-319-11212-1_20

[42] Rivest R.L., Shamir A., Tauman Y. (2001) How to Leak a Secret. In: Boyd C. (eds*) Advances in Cryptology — ASIACRYPT 2001. ASIACRYPT 2001. Lecture Notes in Computer Science*, vol 2248. Springer, Berlin, Heidelberg. Courtois, N.T. and Mercer, R., 2017. Stealth Address and Key Management Techniques in Blockchain Systems. ICISSP, 2017, pp.559-566.

[43] Garcia, F.D. and Jacobs, B., 2010, September. Privacy-friendly energy-metering via homomorphic encryption. In International Workshop on Security and Trust Management (pp. 226-238). Springer, Berlin, Heidelberg.

[44] Zhou, L., Wang, L., Sun, Y. and Lv, P., 2018. Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation. IEEE Access, 6, pp.43472-43488.

[45] Maxwell, G., 2015. Confidential transactions. URL: https://people. xiph. org/greg/confidential values. txt (Accessed 09/05/2016).
[46] Fujisaki, E., Suzuki, K., "Traceable ring signature," in *Public Key Cryptography*, vol. 4450, pp. 181–200, Springer, 2007.
[47] Fujisaki E. (2011) Sub-linear Size Traceable Ring Signatures without Random Oracles. In: *Kiayias A. (eds) Topics in Cryptology – CT-RSA 2011. CT-RSA 2011. Lecture Notes in Computer Science*, vol 6558. Springer, Berlin, Heidelberg
[48] Van Saberhagen.N.2013. Cryptonote v 2. 0.
[49] Noether, S. and Mackenzie, A., 2016. Ring confidential transactions. *Ledger*, *1*, pp.1-18.
[50] Blum, M., Feldman, P. and Micali, S., 2019. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali* (pp. 329-349).
[51] Shor, P.W., 1994, November. Algorithms for quantum computation: discrete logarithms and factoring. *In Proceedings 35th annual symposium on foundations of computer science* (pp. 124-134). Ieee.
[52] Liu, Z., Choo, K.K.R. and Grossschadl, J., 2018. Securing edge devices in the post-quantum internet of things using lattice-based cryptography. *IEEE Communications Magazine*, 56(2), pp.158-162.
[53] Nejatollahi, H., Dutt, N., Ray, S., Regazzoni, F., Banerjee, I. and Cammarota, R., 2019. Post-quantum lattice-based cryptography implementations: A survey. *ACM Computing Surveys* (CSUR), 51(6), pp.1-41.
[54] Dharminder, D. and Mishra, D., 2020. LCPPA: Lattice-based conditional privacy preserving authentication in vehicular communication. *Transactions on Emerging Telecommunications Technologies*, 31(2), p.e3810.
[55] Hoffstein, J., Howgrave-Graham, N., Pipher, J. and Whyte, W., 2009. Practical lattice-based cryptography: NTRUEncrypt and NTRUSign. In *The LLL Algorithm* (pp. 349-390). Springer, Berlin, Heidelberg.
[56] Karu, P. and Loikkanen, J., 2001. Practical comparison of fast public-key cryptosystems. In *Telecommunications Software and Multimedia Lab. at Helsinki Univ. of Technology, Seminar on Network Security* (pp. 1-18). Citeseer.
[57] EESS, "Efficient embedded security standards (eess)," 2003
[58] Howgrave-Graham, N., Nguyen, P.Q., Pointcheval, D., Proos, J., Silverman, J.H., Singer, A. and Whyte, W., 2003, August. The impact of decryption failures on the security of NTRU encryption. In *Annual International Cryptology Conference* (pp. 226-246). Springer, Berlin, Heidelberg.
[59] Hoffstein, J., Pipher, J., Schanck, J.M., Silverman, J.H., Whyte, W. and Zhang, Z., 2017, February. Choosing parameters for NTRUEncrypt. In *Cryptographers' Track at the RSA Conference* (pp. 3-18). Springer, Cham
[60] N. Howgrave-Graham, J. H. Silverman, W. Whyte Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3, *Topics in cryptology*|CT-RSA 2005, 118{135, Lecture Notes in Comput. Sci., 3376, Springer, Berlin, 2005. http://www.ntru.com/cryptolab/articles.htm\#2005_1
[61] P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, W. Whyte, Choosing NTRUEncrypt Parameters in Light of Combined Lattice Reduction and MITM Approaches, ACNS 2009. 437-455
[62] Howgrave-Graham, N., 2007, August. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Annual International Cryptology Conference* (pp. 150-169). Springer, Berlin, Heidelberg.
[63] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H. and Whyte, W., 2003, April. NTRUSIGN: Digital signatures using the NTRU lattice. In *Cryptographers' track at the RSA conference* (pp. 122-140). Springer, Berlin, Heidelberg.
[64] Ponnusamy, V., Kottursamy, K., Karthick, T., Mukeshkrishnan, M.B., Malathi, D. and Ahanger, T.A., 2020. Primary user emulation attack mitigation using neural network. *Computers & Electrical Engineering*, *88*, p.106849.