FL^{2}RS: Forward-Secure Lattice-based Linkable Ring Signature for Vehicle Anonymity in VANETs

Junjie Bian, Dong Xie, Ningning Dong, Fulong Chen, Weixin Bian, and Taochun Wang

***Abstract*—Ring signatures have been extensively researched for Vehicular Ad Hoc Network communications, aiming to safeguard vehicles' identity privacy and the authenticity of traffic messages. While a variety of such schemes have emerged in recent years, the overwhelming majority still depend on classical hardness assumptions, such as the difficulty of discrete logarithms or integer factorization, which are precisely the problems that quantum computing threatens to resolve efficiently. Moreover, forward security, as a critical cryptographic property, has not been adequately addressed or implemented in existing quantum-resistant schemes. In this paper, we introduce FL^{2}RS, a novel linkable ring signature built on lattice assumptions with forward security for VANETs to ensure vehicle privacy through anonymity and enable linkability detection of multiple signatures from the same vehicle. Moreover, forward-secure unforgeability guarantees the security of previous ring signatures even if an adversary obtains the signer's current secret key. To achieve this, we adopt a ternary tree structure to divide epochs and leverage lattice basis algorithms to enable one-way key evolution, so that vehicles refresh their secret keys at regular intervals. We conduct a thorough security analysis and compare our scheme with prior works. Extensive evaluations demonstrate that FL^{2}RS offers the best verifying efficiency, competitive signing and communication performance.**

***Index Terms*—Linkable Ring Signature, lattice-based, Forward-Secure, VANETs.**

I. Introduction

VEHICULAR Ad Hoc Networks (VANETs) serve as the key enabler of the Internet of Things (IoT) ecosystem and form a foundational pillar of Intelligent Transportation Systems (ITS). Their significance lies in the substantial contributions they make toward enhancing road safety and optimizing the efficiency of traffic operations [1]. A typical deployment brings together three categories of participants: Trusted Authority (TA), Roadside Units (RSUs), and vehicles equipped with On-Board Units (OBUs), as illustrated in Fig. 1. In accordance with the Dedicated Short-Range Communication (DSRC) standard [2], vehicles share real-time traffic information such as location, speed, and incident reports with RSUs and other vehicles through wireless communications, (i.e., vehicles-to-vehicle (V2V) and vehicles-to-infrastructure (V2I)). The shared information enables vehicles to avoid traffic congestion and possible traffic accidents [3]. Meanwhile, TA could obtain real-time traffic information via wired communication with RSUs and respond in a timely manner to improve traffic conditions and, consequently, enhance driving safety, traffic flow management and navigation efficiency [4], [5].

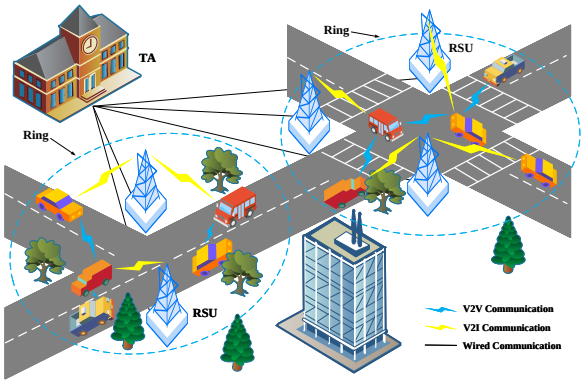


Fig. 1: A Typical Architecture of VANETs.

Although VANET offers significant advantages, it also raises privacy and security concerns for users, such as the exposure of vehicles' location and trajectory information [6], as well as the malicious forgery of traffic messages. To address these issues, a cryptographic primitive known as ring signature [7] has attracted significant attention due to its strong anonymity and unforgeability. Employing ring signatures in VANET communication can effectively protect the privacy of vehicles and ensure the unforgeability of transmitted signatures.

A. Motivations

Various schemes employed with ring signatures have been proposed, but most existing solutions fail to adequately address the following challenges:

- *Vulnerable to quantum attacks:* Most existing schemes [8]–[18] are built upon classical hardness assumptions that are no longer hold in the presence of the quantum adversary. A case in point is Shor's algorithm [19], which breaks both integer factorization and discrete-logarithm-based cryptography in polynomial time. Recently, Brenner et al. [20] demonstrated that integer factorization can be achieved in polynomial time within a continuous-variable framework using only three oscillators and a single qubit, challenging the traditional assumption that the number of qubits must scale with the problem size.

This paper is partially supported by the National Natural Science Foundation of China (Grant Nos. 61801004 and 61972438), the Natural Science Foundation of Anhui Province of China (Grant Nos. 2108085MF219 and 2108085MF206), the Key Research and Development Projects in Anhui Province (Grant Nos. 202004a05020002 and 2022a05020049), and the Wuhu Science and Technology Bureau Project (Grant No. 2023jc17).

Junjie Bian, Dong Xie, Ningning Dong, Fulong Chen, Weixin Bian and Taochun Wang are with the Anhui Provincial Key Laboratory of Industrial Intelligence Data Security, Anhui Engineering Research Centers of Medical Big Data Intelligent System, and the School of Computer and Information, Anhui Normal University, Wuhu 241002, China.

In addition, the U.S. National Institute of Standards and Technology (NIST) has recently published three official standards for post-quantum cryptography and set forth a transition roadmap that calls for completion by 2035 [21]. These actions not only signal the growing maturity of lattice-based cryptographic techniques but also reinforce the urgent need for quantum-resistant security in future communication infrastructures.

- *Vulnerable to forward security:* Although these schemes [9]–[16], [22]–[25] place great emphasis on the security of secret keys, adversaries can still exploit weaknesses in the protection of signers' secret keys to forge signatures. In VANETs, secret key exposure [26], [27] can occur due to vehicle capture, physical tampering [28], or cyberattacks [29], [30]. Once a vehicle's current key is compromised, an adversary can forge signatures on behalf of that vehicle, retroactively falsifying past traffic reports and authentication messages. Forward security [31] guarantees that signatures generated in past time periods remain unforgeable, even if the current secret key is compromised. This property is essential for preserving the integrity and non-repudiation of historical vehicular communications, especially for events like accident reports or liability evidence. Therefore, integrating forward security into ring signature schemes for VANETs emerges as a critical defense against key exposure risks.

B. Contributions

Our main contributions in this work fall into the following categories:

1) *A Ternary Tree-Based Key Update Mechanism.* We devise a ternary tree structure that partitions the time horizon into discrete epochs, with each node associated with a lattice basis. For each epoch, the secret key consists of a minimal cover set of node bases that collectively cover all future epochs. To update the key from epoch t to $t+1$, the user derives the necessary descendant bases from their ancestors and securely deletes the obsolete ones. This one-way evolution ensures that previous secret keys remain unrecoverable even in the event of current key compromise, thereby achieving forward security.
2) *Forward-Secure and Lattice-based Linkable Ring Signature.* We introduce FL^{2}RS, a lattice-based linkable ring signature scheme that achieves anonymity, unforgeability with forward security, and linkability within the standard model. The security of our scheme is built upon the hardness of the Short Integer Solution (SIS) problem. Its resistance to quantum adversaries follows from a fundamental worst-case to average-case reduction: breaking the SIS problem in the average case would imply an efficient solution to hard problems on ideal lattices in the worst case. This cryptographic connection provides the basis for our confidence in the scheme's post-quantum resilience. We formalize the security model and present rigorous proofs showing that FL^{2}RS attains the required security properties.
3) *Comprehensive Performance Evaluation.* We perform extensive experiments and comparative analyses against existing lattice-based ring signature schemes. The experimental results confirm that our FL^{2}RS scheme achieves the lowest verifying cost among all compared schemes. Its signing cost scales slowly with the ring size and becomes competitive in large-scale settings. Although the scheme incurs slightly higher communication overhead for small rings, it remains competitive to state-of-the-art solutions while providing forward security.

C. Organization

The rest of this paper is structured as follows. Sections II and III present the related work and preliminary knowledge, respectively. Section IV describes the method for updating the secret key. Section V provide a detailed description of the proposed scheme. A detailed performance evaluation is then presented in Section VI, followed by concluding remarks in Section VII.

II. Related Work

Since its inception by Rivest et al. [7], the ring signature has allowed a signer to sign a message on behalf of an arbitrary set of public keys, known as the ring, without revealing which particular key in the ring was used for signing. Subsequently, numerous researchers have focused on ring signatures and their applications in privacy-preserving systems. In addition, Table I provides a detailed description of whether each scheme satisfies the corresponding properties.

To address the challenge of generating authenticated information while preserving privacy in VANETs, Gamage et al. [8] introduced a ring signature with a designated verifier, which is secure against full key exposure attacks, i.e. forward-secure. Then, Xiong et al. [9] designed a spontaneous VANET privacy protection protocol leveraging ring signatures that achieves conditional privacy preservation. Büttner et al. [10] proposed an anonymous authenticated key agreement protocol that combines the Elliptic Curve Integrated Encryption Scheme (ECIES) with ring signatures, effectively reducing the number of required pseudonyms. Han et al. [11] developed a dual-protected ring signature for VANETs that requires message encryption prior to signing, with the TA facilitating key distribution between communicating parties. Liu et al. [12], meanwhile, introduced an identity-based variant that achieves conditional privacy and accommodates batch verification.

The absolute anonymity inherent in ring signatures, however, renders them ill-suited for environments where the presence of malicious actors must be taken into account. This limitation can be partially addressed by linkable ring signatures (LRS) [13], which enable the detection of two signatures generated by the same signer. To preserve the location privacy of vehicles, Chang et al. [14] designed a Sybil attack detection mechanism using LRS. Recently, Yadav et al. [15], [16] constructed two anonymous linkable location service schemes that preserve the location and identity privacy of vehicles. Boyen et al. [17] proposed a ring signature that provides both linkability and forward security. In 2023, Li et al. [18] proposed a forward-secure dualring signature.

TABLE I: Comparison of Privacy and Security in Related Work

Properties	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[22]	[23]	[24]	[25]	Ours
Anonymity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Unforgeability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Linkability	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
Quantum Resistance	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
Forward Security	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓

Note: "✓" indicates the presence of a property and "✗" denotes its absence.

However, all the aforementioned schemes rely on classical cryptography, which is vulnerable to quantum attacks.

Limited research has been conducted on the introduction of lattice-based ring signatures into VANET systems. Accordingly, we summarize several lattice-based ring signatures that can be utilized to protect vehicle privacy. Hu et al. [22] presented a lattice-based LRS scheme that is proven secure in the standard model rather than using random oracle heuristics. Zhang et al. [23] later presented a lattice-based LRS scheme that equips blockchain-enabled systems with precise signature traceability. In a related direction, Guo et al. [24] devised a new lattice-based LRS construction aimed at preserving privacy in blockchain-enabled IIoT environments. Recently in 2026, Sourav et al. [25] introduced a secure and efficient lattice-based signcryption scheme designed for blockchain-enabled IoT healthcare systems, which leverages the lattice-based hardness of the Learning With Errors (LWE) problem to ensure strong confidentiality, authentication, and data integrity. Nevertheless, none of the above schemes provides forward security.

III. Preliminaries

To avoid ambiguity, we summarize in Table II the main notations employed in this paper.

TABLE II: Notations

Notation	Explanation
$[n]$	Set $\{1, 2, \ldots, n\}$
$\|\mathbf{A}\|, \|\mathbf{v}\|$	Euclidean norm of a matrix $\mathbf{A}$ and a vector $\mathbf{v}$
$[\mathbf{A}_1 \| \mathbf{A}_2]$	Concatenation of $\mathbf{A}_1$ and $\mathbf{A}_2$
$\tilde{\mathbf{A}}$	Gram-Schmidt orthogonalization of $\mathbf{A}$
$\|\tilde{\mathbf{A}}\|$	Gram-Schmidt norm of $\mathbf{A}$
s	Gaussian parameter
N	Ring size
T	Total number of epochs
id_i	Identity of V_i
sk_i	Secret key of V_i
pk_i	Public key of V_i

A. Lattice

Let $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n \in \mathbb{R}^m$ be linearly independent vectors. The *lattice* generated by these vectors is the set

$$\Lambda(\mathbf{B}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\},$$

where $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ is the *basis matrix*. The integer n is the *rank* of the lattice; when $n = m$, the lattice is called *full-rank*. And we primarily focus on integer lattices in this paper, where each point can be represented by an integer vector. Besides, there is a special category of integer lattices called q-ary lattices.

Definition 1 (q-ary lattices [32])**.** *Given a q, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the q-ary lattices can be defined as*

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} = \mathbf{0} \mod q\},$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} = \mathbf{u} \mod q\}.$$

Definition 2 (Small Integer Solution Problem [33])**.** *Given a large q, positive integers n, $m \geq 2n \log q$, a real number $\beta > 0$, and a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero integer vector $\mathbf{x} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0} \mod q$ and $\|\mathbf{x}\| \leq \beta$.*

B. Lattice Algorithms

Definition 3 (Gaussian Distribution)**.** *Given a vector $\mathbf{c} \in \mathbb{R}^m$ and $s > 0$, the m-dimensional Gaussian function $\rho_{s,\mathbf{c}}$ takes the form $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(-\frac{\pi \|\mathbf{x}-\mathbf{c}\|^2}{s^2}\right)$, for any $\mathbf{x} \in \mathbb{R}^m$.*

Consider a lattice $\Lambda \subset \mathbb{R}^m$. For $\forall \mathbf{x} \in \Lambda$, the discrete Gaussian distribution over Λ with center $\mathbf{c} \in \mathbb{R}^m$ and $s > 0$ is given by $D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}$, where $\rho_{s,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_{s,\mathbf{c}}(\mathbf{x})$ denotes the discrete Gaussian measure of Λ.

Lemma 1. *[34] Given positive integers n, q, m with $q \geq 2$ and $m \geq 2n \log q$, there exists a probabilistic polynomial time (PPT) algorithm, denoted **TrapGen**$(1^n, 1^m, q)$ that can output a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ along with a basis $\mathbf{B}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$ of $\Lambda_q^{\perp}(\mathbf{A})$, where the distribution of $\mathbf{A}$ is statistically close to uniform and $\|\tilde{\mathbf{B}}_{\mathbf{A}}\| \leq \mathbb{O}(\sqrt{n \log q})$.*

Lemma 2. *[33] Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a basis $\mathbf{B}_{\mathbf{A}}$ of its lattice $\Lambda_q^{\perp}(\mathbf{A})$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$, and a parameter $s > \|\tilde{\mathbf{B}}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log m})$ for $m \geq 2n \log q$, the PPT algorithm **SamplePre**$(\mathbf{A}, \mathbf{B}_{\mathbf{A}}, \mathbf{u}, s)$ produces a vector $\mathbf{e} \in \mathbb{Z}_q^m$ that is statistically close to $D_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),s}$ and satisfies $\mathbf{A}\mathbf{e} = \mathbf{u} \pmod q$.*

Lemma 3. *[35] Given positive integers n, q, m, k with $q \geq 2$ and $m \geq 2n \log q$, there exists a PPT algorithm **GenSamplePre**$(\mathbf{A}, \mathbf{B}_S, S, \mathbf{y}, r)$ that on input of $\mathbf{A} \in \mathbb{Z}_q^{n \times km}$, a set $S \subseteq [k]$, a basis $\mathbf{B}_S$ for $\Lambda_q^{\perp}(\mathbf{A}_S)$, a vector $\mathbf{y} \in \mathbb{Z}_q^n$, and an integer $r \geq \|\tilde{\mathbf{B}}_S\| \cdot \omega(\sqrt{\log km})$ outputs $\mathbf{e} =$*

$[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k] \in \mathbb{Z}_q^{km}$ *whose distribution is statistically indistinguishable from* $D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}$.

Without loss of generality, we may assume that $S = [s]$ *for some* $s \in [k]$. *Let* $S^c = [k] \setminus S$. *This algorithm proceeds as follows:*

1) *Sample* $\mathbf{e}_{S^c} \in \mathbb{Z}^{(k-s)\cdot m}$ *from the distribution* $D_{\mathbb{Z}^{(k-s)m},r}$, *and let* $\mathbf{z} = \mathbf{y} - \mathbf{A}_{S^c}\mathbf{e}_{S^c}$. *Parse* $\mathbf{e}_{S^c}$ *as* $[\mathbf{e}_{s+1}, \ldots, \mathbf{e}_k]$. *This defines* $\mathbf{e}_i$ *for each* $i \in S^c$.
2) *Run* $\mathbf{e}_S \leftarrow$ ***SamplePre***$(\mathbf{A}_S, \mathbf{B}_S, \mathbf{z}, r)$ *to sample a vector* $\mathbf{e}_S \in \mathbb{Z}^{sm}$ *from the distribution* $D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}$. *This vector is then decomposed as* $[\mathbf{e}_1, \ldots, \mathbf{e}_s] \in (\mathbb{Z}^m)^s$, *thereby defining* $\mathbf{e}_i$ *for each* $i \in S$.
3) *Output* $\mathbf{e} \in \mathbb{Z}^{km}$, *as* $\mathbf{e} = [\mathbf{e}_1, \ldots, \mathbf{e}_k]$.

Our construction relies on two lattice basis extension algorithms. The first, due to Agrawal, Boneh, and Boyen, is referred to as **BasisExtABB** algorithm [22], while the latter, presented by Cash, Hofheinz, Kiltz, and Peikert [36], is referred to as **BasisExt** algorithm.

Lemma 4. *[22] Given a* q, *positive integers* n, m *with* $m > n$, *the PPT algorithm* ***BasisExtABB***$(\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{T_B})$ *receives as input two matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$, *a matrix* $\mathbf{R} \in \mathbb{Z}^{m\times m}$, *and a basis* $\mathbf{T_B} \in \Lambda_q^{\perp}(\mathbf{B})$. *It then outputs a full-rank basis* $\mathbf{T_F}$ *in* $\Lambda_q^{\perp}(\mathbf{F})$, *where* $\|\tilde{\mathbf{T}}_\mathbf{F}\| < (\|\mathbf{R}\| + 1) \cdot \|\tilde{\mathbf{T}}_\mathbf{B}\|$, *subject to the norm bound* $\mathbf{F} = [\mathbf{A}\|\mathbf{AR} + \mathbf{B}] \in \mathbb{Z}_q^{n\times 2m}$.

Lemma 5. *[36] Given a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *equipped with its basis* $\mathbf{B_A} \in \mathbb{Z}_q^{m\times m}$ *of* $\Lambda^{\perp}(A)$, *and an arbitrary matrix* $\mathbf{A}' \in \mathbb{Z}_q^{n\times m'}$, *the PPT algorithm* ***BasisExt***$(\mathbf{A}'', \mathbf{B_A})$ *outputs a basis* $\mathbf{B}_{\mathbf{A}''}$ *of* $\Lambda^{\perp}(\mathbf{A}'') \in \mathbb{Z}_q^{m\times m''}$ *such that* $\|\tilde{\mathbf{B}}_{\mathbf{A}''}\| = \|\tilde{\mathbf{B}}_\mathbf{A}\|$, *where* $\mathbf{A}'' = \mathbf{A}\|\mathbf{A}'$ *and* $m'' = m + m'$.

IV. Method for Updating the Secret Key

To elaborate the secret key update, we begin by partitioning the epochs using a ternary tree to specify the lattice basis generation for each node. We then provide a rationale for employing the ternary tree.

A. Lattice Basis Generation

We begin by outlining the general method for epoch assignment and the corresponding lattice basis generation at each node of a ternary tree.

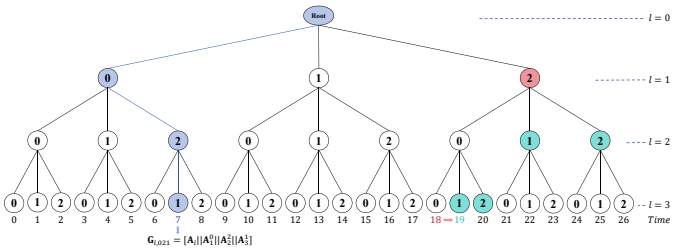


Fig. 2: Ternary tree with $l = 3$ for Time Assignment

The leaf nodes of a ternary tree of depth l are assigned to the epochs $t \in \{0, 1, \ldots, 3^l - 1\}$ in left-to-right order, as illustrated in Fig. 2. For a given epoch t, we define $Set(t)$ as the minimal cover set that contains the ancestors of all leaves in the suffix $\{t, t+1, \ldots, 3^l - 1\}$, while excluding any ancestor of leaves in the prefix $\{0, 1, \ldots, t-1\}$. Specifically, Table IV shows the minimal cover set for each epoch t when $l = 3$.

By the minimal cover set technique, we can present all nodes in the ternary tree (with depth l). On each epoch t, i.e. the leaf node in l-th level, there is an unique path $P(t) = (v_1 v_2 \cdots v_l)$ from the root to that leaf. At the j-th level ($1 \le j \le l$), the value $v_j \in \{0, 1, 2\}$ indicates the branch taken: $v_j = 0$ for a left child, $v_j = 1$ for a middle child, and $v_j = 2$ for a right child. Likewise, for any non-leaf node at level $k < l$, its path from the root is uniquely denoted by $P(k) = (v_1, v_2, \ldots, v_k)$, following the same encoding convention. Then, we define the corresponding matrix $\mathbf{G}_{i,P(t)} = [\mathbf{A}_i \| \mathbf{A}_1^{(v_1)} \| \cdots \| \mathbf{A}_l^{(v_l)}]$ for the leaf node and $\mathbf{G}_{i,P(k)} = [\mathbf{A}_i \| \mathbf{A}_1^{(v_1)} \| \cdots \| \mathbf{A}_l^{(v_k)}]$ for the non-leaf node, where $\mathbf{A}_j^{(b)}$ are matrices for $j \in \{1, 2, \ldots, l\}$ and $b \in \{0, 1, 2\}$. $\mathbf{A}_i$ is the public key generated by **KeyGen** for user i, and is taken as the corresponding matrix of root node. Thus, the initial secret key $\mathbf{S}_{\mathbf{A}_i}$ is the lattice basis for root node. For example, in Fig. 2, on epoch $t = 7$, the path is (021), thus the corresponding matrix $\mathbf{G}_{i,7} = \mathbf{G}_{i,021} = [\mathbf{A}_i \| \mathbf{A}_1^{(0)} \| \mathbf{A}_2^{(2)} \| \mathbf{A}_3^{(1)}]$.

To compute the lattice basis $\mathbf{T}_{P(k)}$ for node $P(k)$ of the ternary tree, the lattice basis extension algorithm **BasisExt** is employed. There are two different methods as below:

- Given the lattice basis of root node $\mathbf{S}_{\mathbf{A}_i}$, we can get the lattice basis $\mathbf{T}_{P(k)} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,P(k)}, \mathbf{S}_{\mathbf{A}_i})$.
- Given its any ancestor node's basis $\mathbf{T}_{P(j)}$, we can get the lattice basis $\mathbf{T}_{P(k)} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,P(k)}, \mathbf{T}_{P(j)})$, where $P(k) = (v_1 \cdots v_j v_{j+1} \cdots v_k)$ for $j < k$.

The same approach can be applied to compute the lattice basis for any epoch (i.e., a leaf node), provided that the basis of its ancestor node is already known.

B. Secret Key Update

Recall that each node in the ternary tree is assigned a lattice basis. Taking the node "10" at level $l = 2$ as an example, its basis $\mathbf{T}_{10}$ corresponds to the lattice $\Lambda_q^{\perp}(\mathbf{G}_{i,10})$, where $\mathbf{G}_{i,10} = [\mathbf{A}_i \mid \mathbf{A}_1^{(1)} \mid \mathbf{A}_2^{(0)}]$. With this node-to-basis mapping in place, the secret key sk_t for epoch t is simply the set of bases for all nodes in $Set(t)$.

To update from t to $t + 1$, the user i computes $Set(t + 1)$, adds the bases of newly included nodes (i.e., those in $Set(t + 1) \setminus Set(t)$), and securely deletes the bases of nodes no longer needed (i.e., those in $Set(t) \setminus Set(t + 1)$). The resulting collection of bases constitutes the new secret key $sk_{i,t+1}$, and the irreversible nature of this update provides forward security. For example, as shown in Fig. 2, for the update from $t = 18$ to $t = 19$, we have $Set(18) = \{2\}$ and $Set(19) = \{201, 202, 21, 22\}$, thus $Set(19) \setminus Set(18) = \{201, 202, 21, 22\}$ and $Set(18) \setminus Set(19) = \{2\}$. Then we should compute $\mathbf{T}_{201} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,201}, \mathbf{T}_2)$, $\mathbf{T}_{202} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,202}, \mathbf{T}_2)$, $\mathbf{T}_{21} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,21}, \mathbf{T}_2)$, $\mathbf{T}_{22} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,22}, \mathbf{T}_2)$, where $\mathbf{G}_{i,201} = [\mathbf{A}_i \| \mathbf{A}_1^{(2)} \| \mathbf{A}_2^{(0)} \| \mathbf{A}_3^{(1)}]$, $\mathbf{G}_{i,202} = [\mathbf{A}_i \| \mathbf{A}_1^{(2)} \| \mathbf{A}_2^{(0)} \| \mathbf{A}_3^{(2)}]$, $\mathbf{G}_{i,21} = [\mathbf{A}_i \| \mathbf{A}_1^{(2)} \| \mathbf{A}_2^{(1)}]$, $\mathbf{G}_{i,22} = [\mathbf{A}_i \| \mathbf{A}_1^{(2)} \| \mathbf{A}_2^{(2)}]$. Finally, we get $sk_{i,19} =$

TABLE III: Performance comparison of n-ary trees ($T = 2^{30}$, $N = 20$). All values normalized to the binary tree ($n = 2$) as baseline.

n	**Signature size**	**Secret key storage**	**BasisExt Count**	**Public Parameter**
2	1.000	1.000	1.000	1.000
3	**0.780** (22.0% ↓)	**0.551** (44.9% ↓)	1.267 (26.7% ↑)	**0.946** (5.4% ↓)
4	0.700 (30.0% ↓)	0.431 (56.9% ↓)	1.500 (50.0% ↑)	1.000 (0.0%)
5	0.660 (34.0% ↓)	0.389 (61.1% ↓)	1.733 (73.3% ↑)	1.083 (8.3% ↑)

$\{\mathbf{T}_{201}, \mathbf{T}_{202}, \mathbf{T}_{21}, \mathbf{T}_{22}\}$ and delete $\mathbf{T}_2$. The whole update process for user i is shown in Table IV.

C. Rationale for Employing the Ternary Tree

In our scheme, the tree-structured key evolution mechanism serves as the cornerstone for achieving both signature unforgeability against key exposure and efficient time management. This subsection provides a rigorous analysis to justify our design choice of a **ternary tree structure** ($n = 3$). We demonstrate that the ternary tree simultaneously outperforms the conventional binary tree in *all three* core efficiency metrics: signature size, secret key storage, and public parameter size.

Theorem 1 (Usefulness of the Ternary Tree)**.** *Let T be the total number of epochs and N the ring size. For an n-ary tree of depth $L_n = \lceil \log_n T \rceil$, the signature size factor is $S_n = N + L_n + 1$, the worst-case secret key storage factor is $K_n = (n-1)\sum_{j=1}^{L_n}(j+1)^2$, and the public parameter factor is $P_n = n\,L_n$.*

Proof. To intuitively illustrate the impact of different tree structures on system performance, we denote the total number of epochs by T. For an n-ary tree with branching factor n, the maximum tree depth is $L_n = \lceil \log_n T \rceil$. The choice of n governs the practical performance of the scheme across the following dimensions.

1) Signature Size: In our proposed scheme, the signature size factor (ignoring the common multiplier $m \log q$) is therefore

$$S_n = N + L_n + 1.$$

For a binary tree ($n = 2$), the depth is $L_2 = \lceil \log_2 T \rceil$ and for a ternary tree ($n = 3$), $L_3 = \lceil \log_3 T \rceil$. The relative signature size is

$$\frac{S_3}{S_2} = \frac{N + L_3 + 1}{N + L_2 + 1}.$$

For example, let $T = 2^{30} \approx 10^9$ (thus $L_2 = 30$, $L_3 = 19$).

- For $N = 5$: $S_3/S_2 \approx 0.686$, 31.4% reduction.
- For $N = 20$: $S_3/S_2 \approx 0.780$, 22.0% reduction.
- For $N = 100$: $S_3/S_2 \approx 0.915$, 8.5% reduction.

In most practical ring signature applications, N is comparable to or smaller than L_2, hence the ternary tree yields substantial bandwidth savings. Increasing n beyond 3 (e.g., $n = 4$ with $L_4 = 15$) gives an additional 8–10 percentage points of signature reduction, but comes at the expense of increased public parameters and higher update complexity, as shown below.

2) Secret Key Storage: To guarantee forward security, a user must maintain a minimal cover set $Set(t)$ of lattice bases for future epochs. In the worst case, $Set(t)$ contains $n-1$ nodes at each depth $j = 1, \ldots, L_n$. The basis of a node at depth j is a matrix of dimension $(j+1)m \times (j+1)m$ and its storage cost scales quadratically with j, i.e., proportional to $(j+1)^2$. Hence, the total secret key storage factor is

$$K_n = (n-1)\sum_{j=1}^{L_n}(j+1)^2, \quad L_n = \ln T / \ln n.$$

We evaluate this function as follows:

- $n = 2$: $K_2 = 10415$
- $n = 3$: $K_3 = 5738$ ($\downarrow 44.9\%$ relative to $n = 2$)
- $n = 4$: $K_4 = 4485$ ($\downarrow 56.9\%$)
- $n = 5$: $K_5 = 4056$ ($\downarrow 61.1\%$)

Thus, contrary to the naive node-count analysis, the ternary tree actually halves the secret key storage compared to the binary tree, due to the cubic dependence on tree depth. The storage continues to decrease for larger n, but the marginal gain from $n = 3$ to $n = 4$ is only 12 percentage points, while public parameter and update costs begin to grow.

*3) Number of **BasisExt** Operations:* Besides signature size and storage, the key update overhead is a critical metric. When the epoch advances from t to $t+1$, the user must compute the basis for each node in $Set(t+1) \setminus Set(t)$ by invoking one **BasisExt** operation per node.

The worst-case number of newly added nodes per update is $(n-1)L_n$, attained at the first transition $t = 0 \to 1$ where $Set(0) = \{\text{root}\}$ and the user must expand all future sibling branches. Hence the worst-case **BasisExt** operation count is

$$U_n = (n-1)\,L_n.$$

For $T = 2^{30}$: $U_2 = 30$, $U_3 = 38$, $U_4 = 45$, $U_5 = 52$. Thus the ternary tree performs 26.7% more **BasisExt** calls per update than the binary tree.

However, the computational cost of a single **BasisExt** at depth j scales with the output basis dimension, namely $\Theta((j+1)^2)$. Summing over all new nodes yields the total update computation cost

$$C_n = (n-1)\sum_{j=1}^{L_n}(j+1)^2 = K_n,$$

which coincides with the storage formula. Consequently, the ternary tree's total update computation cost is 44.9% lower than the binary tree's, despite requiring more individual operations. In other words, the ternary tree trades many lightweight operations at shallow depths for fewer but much heavier operations at deep depths, yielding a net computational saving.

TABLE IV: The Whole Secret Key Update Process for User i When $l = 3$

Update	**Target** t	$Set(t)$	**New node(s)**	**Deleted node(s)**	**New secret key** $sk_{i,t}$
$0 \to 1$	1	$\{001, 002, 01, 02, 1, 2\}$	Calculate from $\mathbf{T}_{\text{root}} = \mathbf{S}_{\mathbf{A}_i}$: $\mathbf{T}_{001} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,001}, \mathbf{T}_{\text{root}})$ $\mathbf{T}_{002} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,002}, \mathbf{T}_{\text{root}})$ $\mathbf{T}_{01} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,01}, \mathbf{T}_{\text{root}})$ $\mathbf{T}_{02} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,02}, \mathbf{T}_{\text{root}})$ $\mathbf{T}_{1} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,1}, \mathbf{T}_{\text{root}})$ $\mathbf{T}_{2} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,2}, \mathbf{T}_{\text{root}})$	$\mathbf{T}_{\text{root}}$	$\{\mathbf{T}_{001}, \mathbf{T}_{002}, \mathbf{T}_{01}, \mathbf{T}_{02}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$1 \to 2$	2	$\{002, 01, 02, 1, 2\}$	None	$\mathbf{T}_{001}$	$\{\mathbf{T}_{002}, \mathbf{T}_{01}, \mathbf{T}_{02}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$2 \to 3$	3	$\{01, 02, 1, 2\}$	None	$\mathbf{T}_{002}$	$\{\mathbf{T}_{01}, \mathbf{T}_{02}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$3 \to 4$	4	$\{011, 012, 02, 1, 2\}$	Calculate from $\mathbf{T}_{01}$: $\mathbf{T}_{011} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,011}, \mathbf{T}_{01})$ $\mathbf{T}_{012} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,012}, \mathbf{T}_{01})$	$\mathbf{T}_{01}$	$\{\mathbf{T}_{011}, \mathbf{T}_{012}, \mathbf{T}_{02}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$4 \to 5$	5	$\{012, 02, 1, 2\}$	None	$\mathbf{T}_{011}$	$\{\mathbf{T}_{012}, \mathbf{T}_{02}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$5 \to 6$	6	$\{02, 1, 2\}$	None	$\mathbf{T}_{012}$	$\{\mathbf{T}_{02}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$6 \to 7$	7	$\{021, 022, 1, 2\}$	Calculate from $\mathbf{T}_{02}$: $\mathbf{T}_{021} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,021}, \mathbf{T}_{02})$ $\mathbf{T}_{022} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,022}, \mathbf{T}_{02})$	$\mathbf{T}_{02}$	$\{\mathbf{T}_{021}, \mathbf{T}_{022}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$7 \to 8$	8	$\{022, 1, 2\}$	None	$\mathbf{T}_{021}$	$\{\mathbf{T}_{022}, \mathbf{T}_{1}, \mathbf{T}_{2}\}$
$8 \to 9$	9	$\{1, 2\}$	None	$\mathbf{T}_{022}$	$\{\mathbf{T}_{1}, \mathbf{T}_{2}\}$
$9 \to 10$	10	$\{101, 102, 11, 12, 2\}$	Calculate from $\mathbf{T}_{1}$: $\mathbf{T}_{101} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,101}, \mathbf{T}_{1})$ $\mathbf{T}_{102} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,102}, \mathbf{T}_{1})$ $\mathbf{T}_{11} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,11}, \mathbf{T}_{1})$ $\mathbf{T}_{12} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,12}, \mathbf{T}_{1})$	$\mathbf{T}_{1}$	$\{\mathbf{T}_{101}, \mathbf{T}_{102}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{2}\}$
$10 \to 11$	11	$\{102, 11, 12, 2\}$	None	$\mathbf{T}_{101}$	$\{\mathbf{T}_{102}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{2}\}$
$11 \to 12$	12	$\{11, 12, 2\}$	None	$\mathbf{T}_{102}$	$\{\mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{2}\}$
$12 \to 13$	13	$\{111, 112, 12, 2\}$	Calculate from $\mathbf{T}_{11}$: $\mathbf{T}_{111} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,111}, \mathbf{T}_{11})$ $\mathbf{T}_{112} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,112}, \mathbf{T}_{11})$	$\mathbf{T}_{11}$	$\{\mathbf{T}_{111}, \mathbf{T}_{112}, \mathbf{T}_{12}, \mathbf{T}_{2}\}$
$13 \to 14$	14	$\{112, 12, 2\}$	None	$\mathbf{T}_{111}$	$\{\mathbf{T}_{112}, \mathbf{T}_{12}, \mathbf{T}_{2}\}$
$14 \to 15$	15	$\{12, 2\}$	None	$\mathbf{T}_{112}$	$\{\mathbf{T}_{12}, \mathbf{T}_{2}\}$
$15 \to 16$	16	$\{121, 122, 2\}$	Calculate from $\mathbf{T}_{12}$: $\mathbf{T}_{121} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,121}, \mathbf{T}_{12})$ $\mathbf{T}_{122} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,122}, \mathbf{T}_{12})$	$\mathbf{T}_{12}$	$\{\mathbf{T}_{121}, \mathbf{T}_{122}, \mathbf{T}_{2}\}$
$16 \to 17$	17	$\{122, 2\}$	None	$\mathbf{T}_{121}$	$\{\mathbf{T}_{122}, \mathbf{T}_{2}\}$
$17 \to 18$	18	$\{2\}$	None	$\mathbf{T}_{122}$	$\{\mathbf{T}_{2}\}$
$18 \to 19$	19	$\{201, 202, 21, 22\}$	Calculate from $\mathbf{T}_{2}$: $\mathbf{T}_{201} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,201}, \mathbf{T}_{2})$ $\mathbf{T}_{202} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,202}, \mathbf{T}_{2})$ $\mathbf{T}_{21} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,21}, \mathbf{T}_{2})$ $\mathbf{T}_{22} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,22}, \mathbf{T}_{2})$	$\mathbf{T}_{2}$	$\{\mathbf{T}_{201}, \mathbf{T}_{202}, \mathbf{T}_{21}, \mathbf{T}_{22}\}$
$19 \to 20$	20	$\{202, 21, 22\}$	None	$\mathbf{T}_{201}$	$\{\mathbf{T}_{202}, \mathbf{T}_{21}, \mathbf{T}_{22}\}$
$20 \to 21$	21	$\{21, 22\}$	None	$\mathbf{T}_{202}$	$\{\mathbf{T}_{21}, \mathbf{T}_{22}\}$
$21 \to 22$	22	$\{211, 212, 22\}$	Calculate from $\mathbf{T}_{21}$: $\mathbf{T}_{211} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,211}, \mathbf{T}_{21})$ $\mathbf{T}_{212} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,212}, \mathbf{T}_{21})$	$\mathbf{T}_{21}$	$\{\mathbf{T}_{211}, \mathbf{T}_{212}, \mathbf{T}_{22}\}$
$22 \to 23$	23	$\{212, 22\}$	None	$\mathbf{T}_{211}$	$\{\mathbf{T}_{212}, \mathbf{T}_{22}\}$
$23 \to 24$	24	$\{22\}$	None	$\mathbf{T}_{212}$	$\{\mathbf{T}_{22}\}$
$24 \to 25$	25	$\{221, 222\}$	Calculate from $\mathbf{T}_{22}$: $\mathbf{T}_{221} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,221}, \mathbf{T}_{22})$ $\mathbf{T}_{222} \leftarrow$ **BasisExt**$(\mathbf{G}_{i,222}, \mathbf{T}_{22})$	$\mathbf{T}_{22}$	$\{\mathbf{T}_{221}, \mathbf{T}_{222}\}$
$25 \to 26$	26	$\{222\}$	None	$\mathbf{T}_{221}$	$\{\mathbf{T}_{222}\}$

Note: All new basis are calculated via **BasisExt** from their ancestor node (already in the current secret key). Once the required descendant basis are calculated, the ancestor basis is immediately deleted (if the ancestor no longer belongs to the next $Set(t+1)$).

4) Public Parameter Size: The public parameters include n random matrices $\mathbf{A}_k^{(0)}, \ldots, \mathbf{A}_k^{(n-1)}$ for each tree level $k = 1, \ldots, L_n$, totalling nL_n matrices. Hence the public parameter factor is

$$P_n = nL_n \propto \frac{n}{\ln n}.$$

The function $n/\ln n$ attains its unique minimum at $n = e \approx 2.718$. Consequently, the ternary tree ($n = 3$) has smaller public parameters than the binary tree ($3/\ln 3 \approx 2.73$ and $2/\ln 2 \approx 2.89$, a 5.4% reduction), while $n = 4$ returns to the same level as $n = 2$ and larger n grows further.

Table III summarizes the normalized performance metrics of different n-ary tree structures, using concrete parameters $T = 2^{30}$ and $N = 20$.

As Table III clearly shows, the ternary tree ($n = 3$) is the unique configuration that simultaneously reduces signature size, secret key storage, and public parameter size over the binary tree. Its only penalty is a 26.7% increase in the number of **BasisExt** calls per update, yet the total update computation cost is 44.9% lower because each call operates on a shallower basis. Moving to $n = 4$ or beyond further increases both the public parameter size and the update operation count, with diminishing returns in signature and storage savings. Larger n continues to degrade public parameter size and update cost with rapidly diminishing signature and storage returns.

In conclusion, the ternary tree structure emerges as the optimal architecture for our forward-secure ring signature scheme, delivering substantial improvements in communication bandwidth, local storage, and system parameter size simultaneously. □

V. Proposed FL^{2}RS

A. System Model

In this subsection, we formalize the syntax of FL^{2}RS primitive based on the definition of linkable ring signatures in [37], [38], consisting of six algorithms, Π=(**Setup**,**KeyGen**,**Sign**,**Verify**,**Link**,**KeyUpdate**).

- $pp \leftarrow$ **Setup**(λ): On input the security parameter λ, the algorithm outputs the system public parameter pp.
- $(pk_i, sk_{i,0}) \leftarrow$ **KeyGen**(pp, id_i): On input the public parameters pp and an identity id_i, the algorithm outputs the corresponding public-secret key pair $(pk_i, sk_{i,0})$ for V_i at the starting time epoch $t = 0$.
- $\sigma \leftarrow$ **Sign**$(pp, sk_{i,t}, R_{pk}, M, t)$: On input the system public parameters pp, the secret key sk_i of V_i, a public-key ring R_{pk}, a message M, and the epoch t, the algorithm outputs a signature σ.
- '1'/ '0' $\leftarrow$ **Verify**$(pp, \sigma, R_{pk}, M, t)$: On input the system public parameters pp, the signature σ, the ring public-key set R_{pk}, the message M, and the epoch t, the algorithm returns '1' upon successful verification; otherwise, it outputs '0'.
- '1'/ '0' $\leftarrow$ **Link**$(pp, \sigma_1, R_{pk_1}, M_1, \sigma_2, R_{pk_2}, M_2, t)$: On input the public parameters pp, two signatures σ_1, σ_2 generated on epoch t, two public keys sets R_{pk_1}, R_{pk_2} and two messages M_1, M_2, this algorithm returns '1' if the signatures are linked; otherwise returns '0'.
- $sk_{i,t+1} \leftarrow$ **KeyUpdate**$(pp, pk_i, sk_{i,t}, t)$: On input the system public parameters pp, the public key pk_i, and the current epoch-t secret key $sk_{i,t}$, the algorithm produces the updated secret key $sk_{i,t+1}$ and subsequently discards the previous key $sk_{i,t}$.

B. Security Model

According to [37]–[39], we define the security model of the FL^{2}RS primitive, which encompasses anonymity, unforgeability with forward security, and linkability. The model is formalized as a game played between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.

We begin by defining the set of oracles that the adversary $\mathcal{A}$ may query throughout the security game.

- Registration Oracle $\mathcal{O}_{\mathcal{R}}$: On query, $\mathcal{C}$ adds a new user to the system by generating the public-secret key pair $(pk_i, sk_{i,0})$ and returning the public key pk_i at original epoch.
- Corruption Oracle $\mathcal{O}_{\mathcal{C}}$: On query a public key pk_i on time t, $\mathcal{C}$ returns the corresponding secret key $sk_{i,t}$.
- Signing Oracle $\mathcal{O}_{\mathcal{S}}$: On query the set of public keys R_{pk} with the public key $pk_\pi \in R_{pk}$ of user π on epoch t and the message M, $\mathcal{C}$ returns $\sigma_{\pi,t}$ by invoking the **Sign** algorithm.

a) Anonymity.: This property ensures that an adversary cannot determine which ring member actually generated the signature. The anonymity game $\mathsf{Game}^{\mathsf{Anon}}$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is defined as follows:

1) $\mathcal{C}$ initializes the system by executing $pp \leftarrow$ **Setup**(λ) and forwards pp to $\mathcal{A}$.
2) $\mathcal{A}$ selects a message M, a ring of public keys $R_{pk} = \{pk_i\}_{i\in[N]}$ (obtained via oracle queries $\mathcal{O}_{\mathcal{R}}$ at time t), and submits (R_{pk}, M, t) to $\mathcal{C}$.
3) $\mathcal{C}$ samples a random index $b \xleftarrow{\$} [N]$, computes the signature $\sigma_b \leftarrow$ **Sign**$(pp, sk_{b,t}, R_{pk}, M, t)$, and returns σ_b to $\mathcal{A}$.
4) $\mathcal{A}$ outputs a guess b' and succeeds if $b' = b$.

For a security parameter λ, the advantage of $\mathcal{A}$ in this game is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathsf{Anon}}]$.

Definition 4 (Anonymity)**.** *For any PPT adversary $\mathcal{A}$ and any ring size N, if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda)$ is negligibly close to $1/N$, we say our scheme is anonymous.*

b) Unforgeability with Forward Security.: This property guarantees that an adversary cannot forge a valid signature for any past epoch, even if the current secret keys of the ring members are exposed. The unforgeability game $\mathsf{Game}^{\mathsf{Forg}}$ is defined as follows:

1) $\mathcal{C}$ runs $pp \leftarrow$ **Setup**(λ) and gives pp to $\mathcal{A}$.
2) $\mathcal{A}$ is allowed to query the oracles $\mathcal{O}_{\mathcal{R}}$, $\mathcal{O}_{\mathcal{C}}$, and $\mathcal{O}_{\mathcal{S}}$, and eventually outputs a tuple (σ_t, R_{pk}, M).
3) $\mathcal{A}$ wins if the following conditions all hold:
   - **Verify**$(pp, \sigma_t, R_{pk}, M, t) = 1$;
   - each public key in R_{pk} has been generated by $\mathcal{O}_{\mathcal{R}}$;
   - none of public key in R_{pk} has been submitted to $\mathcal{O}_{\mathcal{C}}(pk_i, t')$ for any $t' \le t$;

- σ_t was not produced by $\mathcal{O}_{\mathcal{S}}$.

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Forg}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathsf{Forg}}]$.

Definition 5 (Unforgeability with Forward Security)**.** *For any PPT adversary $\mathcal{A}$, if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Forg}}(\lambda)$ is negligible, we say our scheme achieves unforgeability with forward security.*

c) Linkability.: This property ensures that an adversary cannot produce two signatures that appear unlinked but were generated using the same secret key. The linkability game $\mathsf{Game}^{\mathsf{Link}}$ between $\mathcal{C}$ and $\mathcal{A}$ proceeds as follows:

1) $\mathcal{C}$ generates $pp \leftarrow \textbf{Setup}(\lambda)$ and submits pp to $\mathcal{A}$.
2) $\mathcal{A}$ makes queries to $\mathcal{O}_{\mathcal{R}}$, $\mathcal{O}_{\mathcal{C}}$, and $\mathcal{O}_{\mathcal{S}}$, and outputs N tuples $(\sigma_{i,t}, R_{pk_i}, M_i)$ for $i \in [N]$.
3) $\mathcal{A}$ wins if all of the following are satisfied:
   - $\textbf{Verify}(pp, \sigma_{i,t}, R_{pk_i}, M_i, t) = 1$ for all $i \in [N]$;
   - $\textbf{Link}(pp, \sigma_i, R_{pk_i}, M_i, \sigma_j, R_{pk_j}, M_j, t) = 0$ for all $i \neq j$;
   - none of the $\sigma_{i,t}$ is generated by $\mathcal{O}_{\mathcal{S}}$;
   - none of public key in R_{pk} has been queried to $\mathcal{O}_{\mathcal{C}}(pk_i, t')$ for any $t' \leq t$;
   - $\mathcal{A}$ is in possession of no more than one secret key that matches any public key in the ring R_{pk}..

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Link}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathsf{Link}}]$.

Definition 6 (Linkability)**.** *For any PPT adversary $\mathcal{A}$, if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\textbf{\textit{Link}}}(\lambda)$ is negligible, we say our scheme satisfies linkability.*

C. Algorithm Details

– **Setup**(λ): As shown in Alg. 1, given a security parameter λ, TA performs this algorithm as follows:
  1) Chooses a large q, determines the epoch count $T = 3^l$ with l being the ternary-tree depth, specifies the Gaussian sampling parameter s and the message length d, and selects integers n, m satisfying $m > 2n$, where $\frac{m}{2}$ is an integer.
  2) Runs **TrapGen** algorithm to get the master public key $\mathbf{A} \in \mathbb{Z}_q^{n \times \frac{m}{2}}$ and its basis $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2}}$.
  3) Chooses $3l$ matrices $\mathbf{A}_1^{(0)}, \mathbf{A}_1^{(1)}, \mathbf{A}_1^{(2)}, \cdots, \mathbf{A}_l^{(0)}, \mathbf{A}_l^{(1)}, \mathbf{A}_l^{(2)} \in \mathbb{Z}_q^{n \times m}$, together with $d+1$ uniformly random and mutually independent matrices $\mathbf{C}_0, \mathbf{C}_1, \cdots, \mathbf{C}_d \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{g} \in \mathbb{Z}_q^{(l+1)m}$.
  4) Announces the system public parameter $pp = \{q, n, m, l, d, s, \mathbf{g}, \mathbf{A}, \mathbf{A}_1^{(0)}, \mathbf{A}_1^{(1)}, \mathbf{A}_1^{(2)}, \cdots, \mathbf{A}_l^{(0)}, \mathbf{A}_l^{(1)}, \mathbf{A}_l^{(2)}, \mathbf{C}_0, \mathbf{C}_1, \cdots, \mathbf{C}_d\}$.

– **KeyGen**(pp, id_i): As shown in Alg. 2, given the public parameter pp and the identity id_i of V_i, TA performs this algorithm as follows:
  1) Selects randomly two matrix $\mathbf{R}_i \in \mathbb{Z}_q^{\frac{m}{2} \times \frac{m}{2}}$ and $\mathbf{B}_i \in \mathbb{Z}_q^{n \times \frac{m}{2}}$.
  2) Computes the public key $\mathbf{A}_i = [\mathbf{B}_i \| \mathbf{B}_i\mathbf{R}_i + \mathbf{A}] \in \mathbb{Z}_q^{n \times m}$.
  3) Generates the secret key $\mathbf{S}_{\mathbf{A}_i} \in \mathbb{Z}_q^{m \times m}$ by invoking the **BasisExtABB** algorithm.
  4) Returns the public-secret key pair $(\mathbf{A}_i, \mathbf{S}_{\mathbf{A}_i})$ for V_i.

– **Sign**$(pp, sk_{\pi,t}, R_{pk}, M, t)$: As shown in Alg. 3, given the public parameter pp and a ring with public keys set $R_{pk} = \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_N\}$, the signer V_π performs the algorithm to sign the message $M \in \{0\} \times \{0,1\}^d$ at the epoch t using its secret key $sk_{\pi,t}$ as follows:
  1) The procedure begins by checking whether the basis $\mathbf{T}_{P(t)}$ is already included in $sk_{\pi,t}$. In the negative case, he calls $\textbf{BasisExt}(\mathbf{G}_{\pi,P(t)}, \mathbf{T}_{P(j)})$ to generate $\mathbf{T}_{P(t)}$, where $\mathbf{T}_{P(j)}$ denotes an ancestor basis of $\mathbf{T}_{P(t)}$ that is stored in the current secret key $sk_{\pi,t}$.
  2) Computes $\mathbf{C}_M = \sum_{k=0}^{d} (-1)^{M[k]} \mathbf{C}_k \in \mathbb{Z}_q^{n \times m}$ and the link tag $tag = \mathbf{g}^{\mathrm{T}} \mathbf{T}_{\pi,P(t)}$.
  3) Generates $\mathbf{G}_{\pi,P(t)} = [\mathbf{A}_\pi \| \mathbf{A}_1^{(v_1)} \| \cdots \| \mathbf{A}_l^{(v_l)}]$ and form $\mathbf{A}_{R,t} = [\mathbf{A}_1 \| \cdots \| \mathbf{A}_N \| \mathbf{A}_1^{(v_1)} \| \cdots \| \mathbf{A}_l^{(v_l)} \| \mathbf{C}_M]$.
  4) A signature $\mathbf{z} \in \mathbb{Z}_q^{(N+l+1)m}$ is generated by running **GenSamplePre** with parameters $(\mathbf{A}_{R,t}, \mathbf{G}_{\pi,P(t)}, \mathbf{T}_{P(t)}, \mathbf{0}, s)$. The resulting vector lies within negligible statistical distance of the discrete Gaussian $D_{\Lambda_q^{\perp}(\mathbf{A}_{R,t}),s}$.
  5) Outputs the signature $\sigma_\pi = \{\mathbf{z}, tag\}$.

– **Verify**$(pp, \sigma, R_{pk}, M, t)$: As shown in Alg. 4, given the public parameter pp, the signature σ, the public keys set R_{pk}, the signed message M and the epoch t, TA or RSU performs the algorithm as follows:
  1) Checks if $\|\mathbf{z}\| \leq s\sqrt{(N+l+1)m}$, if it is, continues; otherwise, aborts.
  2) Computes $\mathbf{C}_M = \sum_{k=0}^{d} (-1)^{M[k]} \mathbf{C}_k$ and reconstruct $\mathbf{A}_{R,t} = [\mathbf{A}_1 \| \cdots \| \mathbf{A}_N \| \mathbf{A}_1^{(v_1)} \| \cdots \| \mathbf{A}_l^{(v_l)} \| \mathbf{C}_M]$.
  3) Returns '1' if $\mathbf{A}_{R,t} \cdot \mathbf{z} = \mathbf{0}$, otherwise returns '0'.

– **Link**$(pp, \sigma_1, R_{pk_1}, M_1, \sigma_2, R_{pk_2}, M_2, t)$: As shown in Alg. 5, given the public parameter pp, two signatures σ_1, σ_2, two public keys sets R_{pk_1}, R_{pk_2}, two signed messages M_1, M_2 and the epoch t, TA or RSU performs the algorithm as follows:
  1) Runs $b_1 \leftarrow \textbf{Verify}(pp, \sigma_1, R_{pk_1}, M_1, t)$ and $b_2 \leftarrow \textbf{Verify}(pp, \sigma_2, R_{pk_2}, M_2, t)$.
  2) Extracts tag_1 and tag_2.
  3) Returns '1' if $b_1 = b_2 = 1$ and $tag_1 = tag_2$, otherwise returns '0'.

– **KeyUpdate**$(pp, pk_i, sk_{i,t}, t)$: As shown in Alg. 6, given the public parameters pp, the public key pk_i, the secret key $sk_{i,t}$ with the epoch t, V_i performs the algorithm to update the secret key from $sk_{i,t}$ to $sk_{i,t+1}$ in the epoch $t+1$ as follows:
  1) The update from $sk_{i,t}$ to $sk_{i,t+1}$ is carried out by V_i following the rules in Section IV. Specifically, the vehicle first determines the minimal cover set $Set(t+1)$, then collects the lattice bases of nodes belonging exclusively to $Set(t+1)$, and finally erases the bases of nodes that are no longer needed (i.e., those in $Set(t) \setminus Set(t+1)$). The new secret key $sk_{i,t+1}$ is then obtained as the union of the retained bases.

Algorithm 1 Setup(λ)

Require: Security parameter λ
Ensure: Public parameters pp
1: Generate system parameters (q, n, m, l, d, s).
2: $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathbf{TrapGen}(1^n, 1^m, q)$.
3: Choose $\{\mathbf{A}_j^{(b)}\}_{j\in[l],\, b\in\{0,1,2\}}$.
4: Choose $\{\mathbf{C}_k\}_{k=0}^{d}$ and $\mathbf{g}$.
5: Announce $pp = (q, n, m, l, d, s, \mathbf{g}, \mathbf{A}, \{\mathbf{A}_j^{(b)}\}, \{\mathbf{C}_k\})$.
6: **return** pp.

Algorithm 2 KeyGen(pp, id_i)

Require: pp, identity id_i
Ensure: Public key pk_i, secret key sk_i
1: Select $\mathbf{R}_i$ and $\mathbf{B}_i$ uniformly.
2: Compute $\mathbf{A}_i = [\mathbf{B}_i \| \mathbf{B}_i\mathbf{R}_i + \mathbf{A}]$.
3: Generate $\mathbf{S}_{\mathbf{A}_i} \leftarrow \mathbf{BasisExtABB}(\mathbf{B}_i, \mathbf{A}_i, \mathbf{R}_i, \mathbf{T}_{\mathbf{A}})$.
4: **return** $pk_i = \mathbf{A}_i, sk_i = \mathbf{S}_{\mathbf{A}_i}$.

D. Correctness

(a) Correctness of verification

The **GenSamplePre** algorithm ensures that the output vector $\mathbf{z}$ fulfills the following properties with overwhelming probability: $\mathbf{A}_{R,t} \cdot \mathbf{z} = \mathbf{0} \pmod q$ and $\|\mathbf{z}\| \le s\sqrt{(N+l+1)m}$. Furthermore, $\mathbf{z}$ is sampled from a distribution whose statistical distance to $D_{\Lambda_q^{\perp}(\mathbf{A}_{R,t}),s}$ is negligible.

(b) Correctness of linkability

Suppose V_i signs two distinct messages M_1 and M_2 using the same secret key $\mathbf{T}_{i,P(t)}$ to generate two signatures σ_1 and σ_2 that contain link tags tag_1 and tag_2, respectively. The link tag of M_1 is computed as $tag_1 = \mathbf{g}^{\mathrm{T}}\mathbf{T}_{i,P(t)}$, and the link tag of M_2 is computed as $tag_2 = \mathbf{g}^{\mathrm{T}}\mathbf{T}_{i,P(t)}$. Because both tag_1 and tag_2 are generated using the same vector $\mathbf{g}$ and the same secret key $\mathbf{T}_{i,P(t)}$, it follows that $tag_1 = tag_2$.

E. Security Analysis

Theorem 2. *(Anonymity) The proposed FL^{2}RS scheme satisfies anonymity in the standard model. More precisely, for any PPT adversary $\mathcal{A}$ and any ring size $N \ge 2$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda)$ is within negligible statistical distance of $1/N$.*

Proof. Suppose the challenger $\mathcal{C}$ interacts with an adversary $\mathcal{A}$ in this game defined in Definition 4. The game proceeds as follows:

1) $\mathcal{C}$ initializes the system by executing $pp \leftarrow \mathbf{Setup}(\lambda)$ and forwards them to $\mathcal{A}$.
2) When $\mathcal{A}$ queries the registration oracle $\mathcal{O}_{\mathcal{R}}$, $\mathcal{C}$ samples a uniformly random public key pk and sends it to $\mathcal{A}$.
3) In the challenge phase, $\mathcal{A}$ selects a ring $R_{pk} = \{pk_i\}_{i\in[N]}$, a message M, and an epoch t, and submits (R_{pk}, M, t) to $\mathcal{C}$. $\mathcal{C}$ then chooses a random index $b \xleftarrow{\$} [N]$, computes $\sigma_{b,t} \leftarrow \mathbf{Sign}(pp, sk_{b,t}, R_{pk}, M, t)$, and sends $\sigma_{b,t}$ to $\mathcal{A}$.
4) Finally, $\mathcal{A}$ announces a guess value $b' \in [N]$.

Because the distribution of the simulated signature is statistically close to $D_{\Lambda_q^{\perp}(\mathbf{A}_{R,t}),s}$ up to a negligible factor, the view of $\mathcal{A}$ in the simulation is computationally indistinguishable from that in the real scheme. Consequently, $\mathcal{A}$ cannot achieve a success probability better than random guessing, i.e., $\Pr[b' = b] = 1/N$. Therefore, the advantage of $\mathcal{A}$ is negligibly close to $1/N$, and our FL^{2}RS scheme satisfies anonymity. □

Algorithm 3 Sign$(pp, sk_{\pi,t}, R_{pk}, M, t)$

Require: pp, $sk_{\pi,t}$, ring R_{pk}, message M, period t
Ensure: Signature σ_{π}
1: Recover the basis $\mathbf{T}_{P(t)}$ if necessary.
2: Compute $\mathbf{C}_M = \sum_{k=0}^{d}(-1)^{M[k]}\mathbf{C}_k$.
3: Compute link tag $tag = \mathbf{g}^{\mathrm{T}}\mathbf{T}_{\pi,P(t)}$.
4: Generate $\mathbf{G}_{\pi,P(t)}$ and form $\mathbf{A}_{R,t}$.
5: Generate $\mathbf{z} \leftarrow \mathbf{GenSamplePre}(\mathbf{A}_{R,t}, \mathbf{G}_{\pi,P(t)}, \mathbf{T}_{P(t)}, \mathbf{0}, s)$.
6: **return** $\sigma_{\pi} = \{\mathbf{z}, tag\}$.

Algorithm 4 Verify$(pp, \sigma, R_{pk}, M, t)$

Require: pp, $\sigma = (\mathbf{z}, tag)$, R_{pk}, M, t
Ensure: $b \in \{0, 1\}$
1: **if** $\|\mathbf{z}\| > s\sqrt{(N+l+1)m}$ **then**
2: **return** 0
3: **end if**
4: Compute $\mathbf{C}_M$ from M and reconstruct $\mathbf{A}_{R,t}$.
5: **if** $\mathbf{A}_{R,t}\mathbf{z} = \mathbf{0}$ **then**
6: **return** 1
7: **else**
8: **return** 0
9: **end if**

Theorem 3. *(Unforgeability with Forward Security) The proposed FL^{2}RS scheme satisfies the unforgeability with forward security in the standard model under the $SIS_{q,n,(N+3l)m,s}$ hardness assumption.*

Proof. **Setup.** $\mathcal{C}$ prepares the public parameters according to the following procedure:

- Let max be the total number of users, l the ternary-tree depth, d the bit-length of messages, and s the Gaussian sampling parameter. The algorithm also selects an integer $N \le \max$ as the ring size for the challenge, and chooses positive integers q, n, m satisfying the usual lattice dimension constraints.
- Randomly selects $3l$ matrices $\mathbf{X}_1^{(0)}, \mathbf{X}_1^{(1)}, \mathbf{X}_1^{(2)}, \cdots, \mathbf{X}_l^{(0)}, \mathbf{X}_l^{(1)}, \mathbf{X}_l^{(2)} \in \mathbb{Z}_q^{n\times m}$ and sets $\mathbf{G} = [\mathbf{A}_1 \| \cdots \| \mathbf{A}_N \| \mathbf{X}_1^{(0)} \| \mathbf{X}_1^{(1)} \| \mathbf{X}_1^{(2)} \| \cdots \| \mathbf{X}_l^{(0)} \| \mathbf{X}_l^{(1)} \| \mathbf{X}_l^{(2)}] \in \mathbb{Z}_q^{n\times(N+3l)m}$ as SIS matrix, i.e. outputs a non-zero vector $\mathbf{v} \in \mathbb{Z}_q^{(N+3l)m}$ which satisfies $\mathbf{G}\cdot\mathbf{v} = \mathbf{0} \bmod q$.
- Samples a set $R_{\mathbf{u}} = \{u_1, \ldots, u_N\} \in \{1, 2, \ldots, \max\}$ uniformly at random as the challenge ring, and picks $P(t^*) = (v_1^* v_2^* \ldots v_l^*) \in \{0, 1, 2\}^l$ uniformly at random as the target epoch. If $\mathcal{A}$'s forgery does not satisfy $R^* = R_{\mathbf{u}}$ and t^* as the epoch, $\mathcal{C}$ aborts.
- A list $\mathcal{L}$ is created and set to be initially empty. For $i \in [\max]$,
  - if $i \notin R_{\mathbf{u}}$, runs $(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}_i}) \leftarrow \mathbf{TrapGen}(1^n, 1^m, q)$ and stores $(i, \mathbf{A}_i, \mathbf{T}_{\mathbf{A}_i})$ in the list $\mathcal{L}$;

Algorithm 5 Link$(pp, \sigma_1, R_{pk_1}, M_1, \sigma_2, R_{pk_2}, M_2, t)$

Require: pp, two signatures σ_1, σ_2, rings R_{pk_1}, R_{pk_2}, messages M_1, M_2, period t
Ensure: $b \in \{0, 1\}$
$b_1 \leftarrow$ **Verify**$(pp, \sigma_1, R_{pk_1}, M_1, t)$.
$b_2 \leftarrow$ **Verify**$(pp, \sigma_2, R_{pk_2}, M_2, t)$.
if $b_1 = b_2 = 1$ and $tag_1 = tag_2$ **then**
  return 1
else
  return 0
end if

Algorithm 6 KeyUpdate$(pp, pk_i, sk_{i,t}, t)$

Require: pp, pk_i, current secret key $sk_{i,t}$, period t
Ensure: Updated secret key $sk_{i,t+1}$
Compute the minimal cover set $Set(t+1)$.
Determine $\Delta^+ = Set(t+1) \setminus Set(t)$, $\Delta^- = Set(t) \setminus Set(t+1)$.
for all $v \in \Delta^+$ **do**
  Derive basis $\mathbf{T}_v \leftarrow$ **BasisExt**$(\mathbf{G}_{i,v}, \mathbf{S}_{\mathbf{A}_i})$, i.e. when the epoch $t = 0$.
  Derive basis $\mathbf{T}_v \leftarrow$ **BasisExt**$(\mathbf{G}_{i,v}, \mathbf{T}_{v'})$, i.e. when the epoch $t \geq 1$, where v' is the ancestor of v.
end for
for all $v \in \Delta^-$ **do**
  Securely erase $\mathbf{T}_v$.
end for
Update $sk_{i,t+1} = \big(sk_{i,t} \cup \{\mathbf{T}_v : v \in \Delta^+\}\big) \setminus \{\mathbf{T}_v : v \in \Delta^-\}$.
return $sk_{i,t+1}$.

  - if $i = u_{i^*} \in R_{\mathbf{u}}$, sets $\mathbf{A}_i = \mathbf{A}_{i^*}$.
- For each level $k \in [l]$ and bit $b \in \{0, 1, 2\}$,
  - if $b = v_k^*$, sets $\mathbf{A}_k^{(b)} = \mathbf{X}_k^{(b)}$;
  - if $b \neq v_k^*$, runs $(\mathbf{A}_k^{(b)}, \mathbf{T}_{\mathbf{A}_k^{(b)}}) \leftarrow$ **TrapGen**$(1^n, 1^m, q)$ and keeps $\mathbf{T}_{\mathbf{A}_k^{(b)}}$.
- Chooses $d+1$ random short matrices $\mathbf{D}_0, \ldots, \mathbf{D}_d \in \mathbb{Z}_q^{(N+l)m \times m}$ and scalars $h_0, h_1, \ldots, h_d \in \mathbb{Z}_q$, where $h_0 = 1$. Runs $(\mathbf{E}, \mathbf{T}_{\mathbf{E}}) \leftarrow$ **TrapGen**$(1^n, 1^m, q)$ and computes $\mathbf{C}_j = \mathbf{A}_{R_{\mathbf{u}},t^*}\mathbf{D}_j + h_j\mathbf{E} \bmod q$ for $j = 0, \ldots, d$, where $\mathbf{A}_{R_{\mathbf{u}},t^*} = [\mathbf{A}_1 \| \cdots \| \mathbf{A}_N \| \mathbf{A}_1^{(v_1^*)} \| \cdots \| \mathbf{A}_l^{(v_l^*)}]$.
- Sends the system parameters $\{q, n, m, l, d, s, \max, \{\mathbf{A}_k^{(0)}, \mathbf{A}_k^{(1)}, \mathbf{A}_k^{(2)}\}_{k=1}^l, \{\mathbf{C}_j\}_{j=0}^d\}$ and the public key set $\{\mathbf{A}_1, \ldots, \mathbf{A}_{\max}\}$ to $\mathcal{A}$.

Queries. $\mathcal{C}$ maintains the list $\mathcal{L}$ and answers $\mathcal{A}$'s queries as follows:

- *Corruption query* $\mathcal{O}_{\mathcal{C}}$*:* For a epoch $P(t) = (v_1 v_2 \ldots v_l)$ and the signer i. If the condition $pk_i \notin R_{\mathbf{u}}$ holds, or if the public key pk_i belongs to the challenged signer, then $\mathcal{C}$ terminates the execution. Else if $pk_i \in R_{\mathbf{u}}$,
  - if $t \leq t^*$, $\mathcal{C}$ aborts;
  - if $t > t^*$, we assume that k be the first index where $v_k \neq v_k^*$. By construction, $\mathcal{C}$ possesses the trapdoor $\mathbf{T}_{\mathbf{A}_k^{(v_k)}}$. Then, $\mathcal{C}$ computes the basis for v_k as $\mathbf{T}_{v_k} \leftarrow$ **BasisExt**$\big([\mathbf{A}_i \| \mathbf{A}_1^{(v_1)} \| \cdots \| \mathbf{A}_k^{(v_k)}], \mathbf{T}_{\mathbf{A}_k^{(v_k)}}\big)$ and expands it down to the leaf t to obtain $sk_{i,t}$.
- *Signing query* $\mathcal{O}_{\mathcal{S}}$*:* For a epoch $P(t) = (v_1 v_2 \ldots v_l)$, the ring R, the message M and the signer i. If $R \neq R_{\mathbf{u}}$ or $t \neq t^*$, $\mathcal{C}$ is able to generate a valid signature $\mathbf{e}$ through the basis of some user in R that is not in $R_{\mathbf{u}}$, or by using the basis of a time-node matrix as in the corruption case. In all these cases $\mathcal{C}$ knows a suitable short basis and invokes **GenSamplePre** honestly. If $R = R_{\mathbf{u}}$ and $t = t^*$, $\mathcal{C}$ first computes $h_M = \sum_{j=0}^d (-1)^{M[j]} h_j \in \mathbb{Z}_q$, $\mathbf{D}_M = \sum_{j=0}^d (-1)^{M[j]} \mathbf{D}_j \in \mathbb{Z}_q^{(N+l)m \times m}$.
  - If $h_M = 0$, $\mathcal{C}$ aborts.
  - If $h_M \neq 0$, then $\mathbf{C}_M = \mathbf{A}_{R_{\mathbf{u}},t^*}\mathbf{D}_M + h_M\mathbf{E}$. Using the basis $\mathbf{T}_{\mathbf{E}}$ of $\mathbf{E}$ and the non-zero scalar h_M, $\mathcal{C}$ can sample a short $\mathbf{e}$ satisfying $[\mathbf{A}_{R_{\mathbf{u}},t^*} \| \mathbf{C}_M] \cdot \mathbf{e} = \mathbf{0} \bmod q$, exactly as required by the signature.

Forgery. Eventually, the adversary $\mathcal{A}$ submits a forgery tuple $(M^*, R^*, \sigma^*, t^*)$. However, $\mathcal{C}$ terminates the simulation unless $R^* = R_{\mathbf{u}}$ and the epoch matches t^*; otherwise, the challenger aborts. Otherwise $\mathcal{C}$ computes h_{M^*} and $\mathbf{D}_{M^*}$ analogously. If $h_{M^*} \neq 0$, $\mathcal{C}$ aborts.

When $\mathcal{C}$ does not abort and $h_{M^*} = 0$, the forgery satisfies

$$[\mathbf{A}_{R_{\mathbf{u}},t^*} \| \mathbf{C}_{M^*}] \cdot \sigma^* = \mathbf{0} \bmod q.$$

Because $h_{M^*} = 0$, we have $\mathbf{C}_{M^*} = \mathbf{A}_{R_{\mathbf{u}},t^*}\mathbf{D}_{M^*}$. Write $\sigma^* = \begin{pmatrix} \sigma_1^* \\ \sigma_2^* \end{pmatrix}$ with $\sigma_1^* \in \mathbb{Z}^{(N+l)m}$ and $\sigma_2^* \in \mathbb{Z}^m$. Then

$$\mathbf{A}_{R_{\mathbf{u}},t^*} \cdot (\sigma_1^* + \mathbf{D}_{M^*}\sigma_2^*) = \mathbf{0} \bmod q.$$

Define $\mathbf{e}^* = \sigma_1^* + \mathbf{D}_{M^*}\sigma_2^*$, which satisfies $\mathbf{A}_{R_{\mathbf{u}},t^*}\mathbf{e}^* = \mathbf{0}$.

Now $\mathcal{C}$ expands $\mathbf{e}^*$ into a vector $\mathbf{v} \in \mathbb{Z}^{(N+3l)m}$ as follows:

- The first $N \cdot m$ entries correspond to $\mathbf{A}_1, \ldots, \mathbf{A}_N$, thus copy the first N blocks of $\mathbf{e}^*$ directly.
- For each level $k = [l]$:
  - at the position of $\mathbf{X}_k^{(v_k^*)}$ (which equals $\mathbf{A}_k^{(v_k^*)}$), copy the $(N+k)$-th block of $\mathbf{e}^*$;
  - at the positions of $\mathbf{X}_k^{(b)}$ for $b \neq v_k^*$, insert $\mathbf{0}$ vectors of length m.

Because $\mathbf{A}_{R_{\mathbf{u}},t^*}$ contains exactly $\mathbf{A}_i$ (for $i \in R_{\mathbf{u}}$) and $\mathbf{A}_k^{(v_k^*)}$ for $k = [l]$, the product $\mathbf{G} \cdot \mathbf{v} = \mathbf{0}$ collapses to $\mathbf{A}_{R_{\mathbf{u}},t^*} \cdot \mathbf{e}^* = \mathbf{0}$. Moreover, $\|\mathbf{v}\| = \|\mathbf{e}^*\|$ and $\mathbf{v} \neq \mathbf{0}$ with overwhelming probability (otherwise the forgery would be trivial). Finally, $\mathcal{C}$ outputs $\mathbf{v}$ as the SIS solution. But as we all know, the SIS problem has been widely recognized as a hard problem, which indicates that our FL^{2}RS scheme is unforgeability with forward security within the standard model. □

Theorem 4. *(Linkability) Our FL^{2}RS scheme satisfies the linkability in the standard model if the FL^{2}RS scheme satisfies the unforgeability with forward security.*

Proof. Suppose there exists an adaptive adversary $\mathcal{A}$ whose goal is to compromise the linkability property of FL^{2}RS. In response, a challenger $\mathcal{C}$ interacts with $\mathcal{A}$ by responding to its oracle queries throughout the execution of the security game.

For the oracle simulation, $\mathcal{A}$ begins its queries, following the procedures outlined in V-B.

For the unlinking phrase, $\mathcal{A}$ outputs $(\sigma_{i,t}, R_{pk_i}, M_i, t)$ and $(\sigma_{j,t}, R_{pk_j}, M_j, t)$, where $i, j \in [N], i \neq j$.

Assume that $\sigma_{i,t}$ and $\sigma_{j,t}$ are both valid signatures for previously unsigned message-ring pairs, and since the FL^{2}RS scheme satisfies the unforgeability with forward security, it must be the case that $\mathcal{A}$ generated these signatures honestly using the legitimate signing algorithm **Sign** with actual secret keys. In particular, there exist secret keys $sk_{i,t}$ and $sk_{j,t}$ such that ($\mathbf{T}_{i,P(t)}$ in $sk_{i,t}$ and $\mathbf{T}_{j,P(t)}$ in $sk_{j,t}$ are used):

$$\sigma_{i,t} \leftarrow \textbf{Sign}(pp, sk_{i,t}, R_{pk_i}, M_i, t),$$

$$\sigma_{j,t} \leftarrow \textbf{Sign}(pp, sk_{j,t}, R_{pk_j}, M_j, t).$$

If $\mathcal{A}$ did not possess the secret key used to create either signature, the adversary would have successfully forged a valid signature, contradicting the unforgeability property. Therefore, the secret keys used to generate $\sigma_{i,t}$ and $\sigma_{j,t}$ must be known to $\mathcal{A}$. Moreover, $\mathcal{A}$ can corrupt at most one secret key among the users in the union of the two rings. Consequently, $\mathcal{A}$ knows at most one distinct secret key. Hence, the keys used to produce the two signatures must be identical.

In the **Sign** algorithm, the link tag is computed deterministically from the signer's secret key and the public vector $\mathbf{g}$. Thus, the link tags contained in $\sigma_{i,t}$ and $\sigma_{j,t}$ are respectively $tag_{i,t} = \mathbf{g}^{\mathrm{T}}\mathbf{T}_{i,P(t)}, tag_{j,t} = \mathbf{g}^{\mathrm{T}}\mathbf{T}_{j,P(t)}$. Since both tags are generated using the identical secret key (basis) and the same public parameter, we have $tag_{i,t} = tag_{j,t}$, which implies that the **Link** algorithm, upon checking whether the two link tags are equal, will return '1'. This directly contradicts the assumption of the linkability game. Therefore, the adversary's success probability in the linkability game is negligible and our FL^{2}RS scheme satisfies linkability. □

VI. Performance Analysis

This section presents a comparative performance analysis between our FL^{2}RS scheme and a number of representative lattice-based ring signature constructions [22]–[25], [40] in computational cost and communication overhead. Specifically, all simulation experiments were conducted in a system environment with an Intel(R) Core(TM) i7-10750H processor and a memory of 16.0 GB in Python. In general, we evaluate the cost and overhead in the parameter setting environments where $n = 16$, $m = 800$, $q = 2^{25} - 1$ and $l = 2$.

A. Comparison of Computational Cost

In this subsection, we analyze the computational cost involved in each scheme in terms of the **Sign** algorithm and **Verify** algorithm. Additionally, N represents the number of vehicles, i.e., the ring size.

Fig. 3 illustrates the computational costs of signing and verifying under different numbers of vehicles. As shown in Fig. 3(a), the proposed scheme does not achieve the lowest signing cost. Specifically, scheme [25] consistently outperforms the proposed scheme in terms of signing efficiency, with their signing costs remaining below 6 *ms*. However, it is evident that our scheme exhibits the slowest growth in signing time as the ring size increases, and at $N = 100$, it becomes comparable to that of scheme [25]. Moreover, the signing cost of our scheme remains significantly lower than those of schemes [22], [23] and [24], whose costs exceed 64 *ms* in large-scale settings.

As can be observed from Fig. 3(b), our scheme achieves the lowest verification overhead among the compared schemes. The verifying cost is approximately 1 *ms* when the ring size is 100. In contrast, scheme [23] suffers from a substantial increase in verifying cost, reaching nearly 38 *ms* when 100 vehicles participate in the ring.

In addition, we analyze the impact of the ternary tree depth l on the computational cost. Fig. 4(a) and 4(b) illustrate the effect of depth l on the computational cost when the ring size is set to 20. The cost of both **Sign** and **Verify** algorithms increase as depth l increases.

B. Comparison of Communication Overhead

Since a large number of signatures are transmitted over the channel, we focus primarily on comparing their sizes. Table V presents a theoretical comparative analysis of the public key, secret key, signature size, and link tag, where N is the size of the ring.

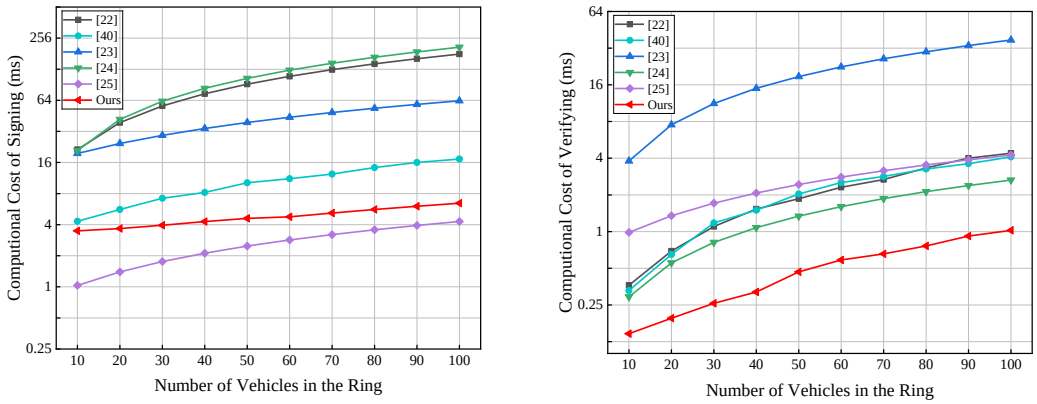
In our scheme, the signature is $\mathbf{z} \in \mathbb{Z}_q^{(N+l+1)m}$. Thus, the maximal size of signature is $(N + l + 1)m \log q = ((N + 3) * 800) * 25 = 20000N + 60000$ bits, i.e., $2500N + 7500$ bytes. The maximal size of signature in [22]–[25], [40] can be calculated by the same way.

Fig. 5(a) presents the sizes of the public and secret keys for each scheme (The initial secret key is used in our scheme). Since all public keys have the same size, the communication overhead for broadcasting them is identical across schemes. Therefore, the dominant factor in communication overhead is the signature size. It is easy to see from Fig. 5(b) that our scheme incurs slightly higher overhead than schemes [23]–[25] for small ring size (i.e., below 50). However, as the ring size grows larger, the overhead of our scheme becomes comparable to that of schemes [23]–[25]. Furthermore, only our scheme and those in [22]–[24] achieve linkability. In addition, while both our scheme and that of [40] guarantee forward security in a quantum setting, the latter incurs significantly higher communication overhead.

Fig. 4(c) illustrates the effect of the ternary tree depth l on the signature size of our scheme when the ring size is fixed at 20. Specifically, the signature size of our scheme increases as l grows.

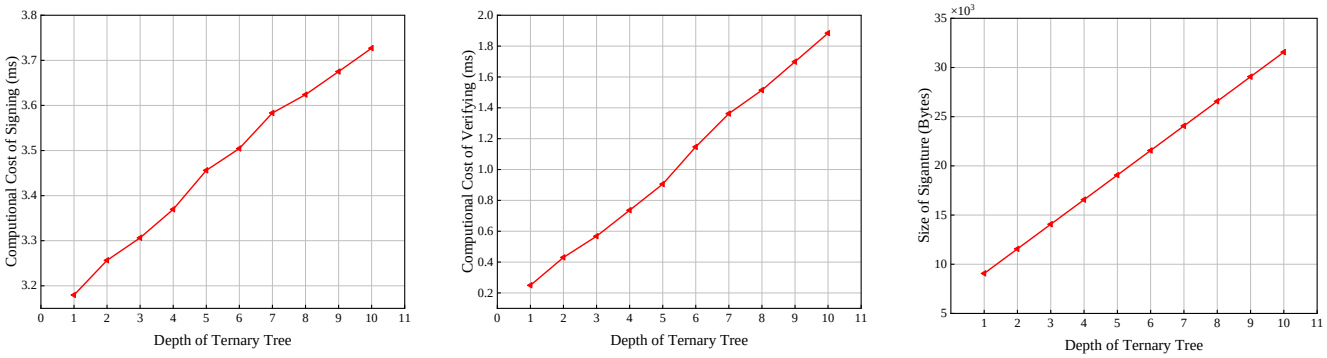
VII. Conclusion

In this work, we present FL^{2}RS, a novel lattice-based linkable ring signature scheme that mitigates both quantum threats and key exposure risks in VANETs. By integrating a ternary tree structure for time partitioning with one-way secret key evolution, FL^{2}RS successfully guarantees anonymity, linkability, and forward-secure unforgeability. Formal security analysis proves that FL^{2}RS reduces to the hardness of the standard SIS problem. Remarkably, the computational cost for signing is lower than that of the vast majority of existing schemes, the verifying cost is the lowest among all compared



(a) Cost of Signing (b) Cost of Verifying

Fig. 3: Comparison of computational cost between our FL^{2}RS scheme and [22]–[25], [40].

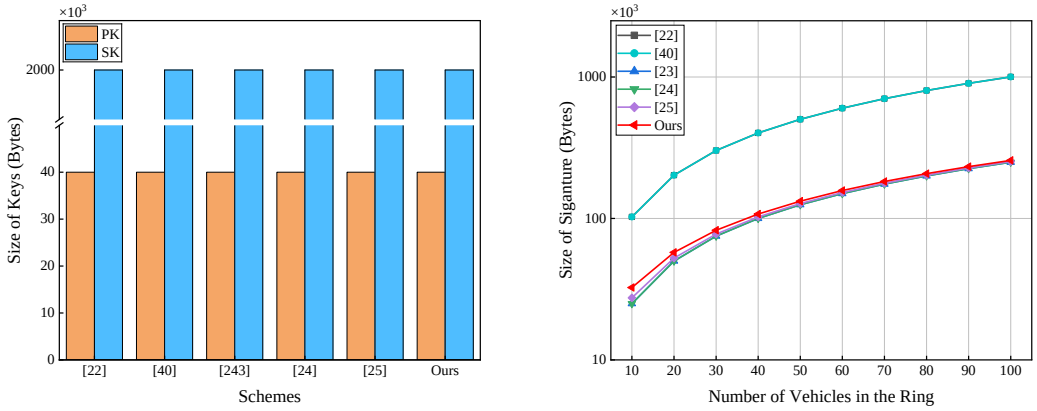


(a) Cost of Signing (b) Cost of Verifying (c) Size of Signature

Fig. 4: Computational cost and communication overhead associated with l for our FL^{2}RS scheme.

TABLE V: Comparison of Storage and Communication Overhead

Schemes	Year	Size of Keys		Size of Signature and Link Tag	
		Public Key	Secret Key	Signature	Link Tag
Hu et al. [22]	2022	$mn\log q$	$m^2\log q$	$((4N+1)m)\log q$	$mn\log q$
Yu et al. [40]	2023	$mn\log q$	$m^2\log q$	$((N(l+1)+1)m)\log q$	–
Zhang et al. [23]	2024	$mn\log q$	$m^2\log q$	$Nm\log q$	$mn\log q$
Guo et al. [24]	2025	$mn\log q$	$m^2\log q$	$Nm\log q$	$m\log q$
Sourav et al. [25]	2026	$mn\log q$	$m^2\log q$	$(N+1)m\log q$	–
Ours	-	$mn\log q$	$m^2\log q$	$(N+l+1)m\log q$	$(l+1)m\log q$



(a) Size of Keys (b) Size of Signature

Fig. 5: Comparison of communication overhead between our FL^{2}RS scheme and [22]–[25], [40].

schemes, and the communication overhead also compares favorably. Notably, although our scheme currently incurs higher overhead than conventional cryptographic techniques, we believe that practical forward-secure lattice-based ring signatures are of broad interest given the rapid development of quantum computing. Our work constitutes an important step in this direction and may stimulate interest in new quantum-resistant research. In future work, we will consider using Ring-LWE based cryptosystems to optimize the scheme and reduce its overhead.

REFERENCES

[1] R. Hussain, J. Lee, and S. Zeadally, "Trust in VANET: A survey of current solutions and future research opportunities," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2553–2571, 2021.

[2] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[3] C. Wang, Y. Ming, H. Liu, J. Feng, and N. Zhang, "Secure and flexible data sharing with dual privacy protection in vehicular digital twin networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 12 407–12 420, 2024.

[4] C. Yao, X. Liu, J. Wang, and Y. Cheng, "Optimized design of edgeboard intelligent vehicle based on PP-YOLOE+," *Sensors*, vol. 24, no. 10, p. 3180, 2024.

[5] Y. Chen, J. Zhang, Z. Xie, W. Li, F. Zhang, J. Lu, and L. Zhang, "S-NeRF++: Autonomous driving simulation via neural reconstruction and generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 6, pp. 4358–4376, 2025.

[6] S. Li and H.-S. Yoon, "Sensor fusion-based vehicle detection and tracking using a single camera and radar at a traffic intersection," *Sensors*, vol. 23, no. 10, p. 4888, 2023.

[7] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *International conference on the theory and application of cryptology and information security*. Springer, 2001, pp. 552–565.

[8] C. Gamage, B. Gras, B. Crispo, and A. S. Tanenbaum, "An identity-based ring signature scheme with enhanced privacy," in *2006 Securecomm and Workshops*. IEEE, 2006, pp. 1–5.

[9] H. Xiong, K. Beznosov, Z. Qin, and M. Ripeanu, "Efficient and spontaneous privacy-preserving protocol for secure vehicular communication," in *2010 IEEE International Conference on Communications*. IEEE, 2010, pp. 1–6.

[10] C. Büttner and S. A. Huss, "A novel anonymous authenticated key agreement protocol for vehicular ad hoc networks," in *2015 International Conference on Information Systems Security and Privacy (ICISSP)*. IEEE, 2015, pp. 259–269.

[11] Y. Han, N.-N. Xue, B.-Y. Wang, Q. Zhang, C.-L. Liu, and W.-S. Zhang, "Improved dual-protected ring signature for security and privacy of vehicular communications in vehicular ad-hoc networks," *IEEE Access*, vol. 6, pp. 20 209–20 220, 2018.

[12] F. Liu and Q. Wang, "IBRS: An efficient identity-based batch verification scheme for VANETs based on ring signature," in *2019 IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–8.

[13] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Australasian conference on information security and privacy*. Springer, 2004, pp. 325–335.

[14] S. Chang, Y. Qi, H. Zhu, J. Zhao, and X. Shen, "Footprint: Detecting sybil attacks in urban vehicular networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 6, pp. 1103–1114, 2012.

[15] V. K. Yadav, S. Verma, and S. Venkatesan, "Linkable privacy-preserving scheme for location-based services," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7998–8012, 2022.

[16] V. K. Yadav, N. Andola, S. Verma, and S. Venkatesan, "P2LBS: Privacy provisioning in location-based services," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 466–477, 2023.

[17] X. Boyen and T. Haines, "Forward-secure linkable ring signatures," in *Australasian Conference on Information Security and Privacy*. Springer, 2018, pp. 245–264.

[18] N. Li, Y. Li, A. Miyaji, Y. Tian, and T. H. Yuen, "A practical forward-secure dualring," in *International Conference on Cryptology and Network Security*. Springer, 2023, pp. 516–537.

[19] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[20] L. Brenner, L. Caha, X. Coiteux-Roy, and R. Koenig, "Factoring an integer with three oscillators and a qubit," *Nature Communications*, 2025.

[21] D. Moody, R. Perlner, A. Regenscheid, A. Robinson, and D. Cooper, "Transition to post-quantum cryptography standards," National Institute of Standards and Technology, Tech. Rep., 2024.

[22] M. Hu and Z. Liu, "Lattice-based linkable ring signature in the standard model," *Cryptology ePrint Archive*, 2022.

[23] Y. Zhang, Y. Tang, C. Li, M. Dong, M. Huang, H. Zhang, and K. Ota, "Privacy-preserving for blockchain-enabled cold-chain logistics system with IoV and linkable ring signature," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 9, pp. 12 585–12 596, 2024.

[24] F. Guo, Y. Gao, J. Jiang, X. Chen, X. Chen, and Z. Jiang, "Linkable ring signature for privacy protection in blockchain-enabled IIoT," *Sensors*, vol. 25, no. 12, 2025.

[25] Sourav, R. Ali, and M. S. Obaidat, "Secure and efficient lattice-based signcryption for blockchain-enabled IoT healthcare," *IEEE Transactions on Dependable and Secure Computing*, vol. 23, no. 1, pp. 356–367, 2026.

[26] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1167–1179, 2015.

[27] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Information Sciences*, vol. 472, pp. 223–234, 2019.

[28] T. Park and K. Shin, "Soft tamper-proofing via program integrity verification in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 297–309, 2005.

[29] S. H. Kim, Q.-H. Wang, and J. B. Ullrich, "A comparative study of cyberattacks," *Communications of the ACM*, vol. 55, no. 3, pp. 66–73, 2012.

[30] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and countermeasures for in-vehicle networks," *ACM computing surveys (CSUR)*, vol. 54, no. 1, pp. 1–37, 2021.

[31] X. Huang, J. K. Liu, S. Tang, Y. Xiang, K. Liang, L. Xu, and J. Zhou, "Cost-effective authentic and anonymous data sharing with forward security," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 971–983, 2015.

[32] C. Peikert, "An efficient and parallel gaussian sampler for lattices," in *Annual Cryptology Conference*. Springer, 2010, pp. 80–97.

[33] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 197–206.

[34] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2012, pp. 700–718.

[35] D. Cash, D. Hofheinz, and E. Kiltz, "How to delegate a lattice basis," *Cryptology ePrint Archive*, 2009.

[36] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," *Journal of cryptology*, vol. 25, no. 4, pp. 601–639, 2012.

[37] J. K. Liu, M. H. Au, W. Susilo, and J. Zhou, "Linkable ring signature with unconditional anonymity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 157–165, 2014.

[38] X. Lu, M. H. Au, and Z. Zhang, "Raptor: a practical lattice-based (linkable) ring signature," in *International Conference on Applied Cryptography and Network Security*. Springer, 2019, pp. 110–130.

[39] J. K. Liu, T. H. Yuen, and J. Zhou, "Forward secure ring signature without random oracles," in *International Conference on Information and Communications Security*. Springer, 2011, pp. 1–14.

[40] X. Yu and Y. Wang, "Forward secure lattice-based ring signature scheme in the standard model," in *International Conference on Information and Communications Security*. Springer, 2023, pp. 146–158.