Lether: Efficient Post-Quantum Account-Based Private Blockchain Payments

***Abstract*—We introduce Lether, the *first* efficient account-based private blockchain payment protocol based on *post-quantum* lattice assumptions, following the paradigm of Anonymous Zether (FC '19, IEEE S&P '21). The main challenge in building such a protocol from lattices lies in the absence of core building blocks: unbounded-level additively-homomorphic multi-message multi-recipient public key encryption (mmPKE), and event-oriented linkable ring signatures with support for multiple tags (i.e., events). To address these issues, we propose a verifiable refreshable additively-homomorphic mmPKE scheme and a plug-and-play event-oriented linkable tag scheme from lattices. We believe both to be of independent interest.**

Furthermore, to achieve unbounded-level homomorphic evaluation in the lattice-based setting *without* relying on heavy techniques such as bootstrapping or large moduli (e.g., over 60 bits) in fully homomorphic encryption (FHE), we introduce a simple yet blockchain-friendly mechanism called *refresh*. Namely, each user is required to verifiably refresh their account after a certain number of transactions. With our tailored parameter settings, the amortized per-refresh costs of communication and computation are only about 1.3% and 1.5%, respectively, of the cost of a transaction.

We also provide an implementation of Lether, demonstrating the practicality of our scheme. Specifically, for a typical transaction, the communication cost is approximately 68 KB, with the internal proof accounting for about 51 KB. Both the proof generation and verification take a fraction of a second each on a standard PC.

As an additional contribution, we formalize new definitions for Anonymous Zether-like protocols that more accurately capture real-world blockchain settings. These definitions are generic and are expected to benefit the broader development of account-based private blockchain payment protocols, beyond just the lattice-based setting.

1. Introduction

Rapid progress in quantum computing [1] has led to a global shift towards post-quantum cryptography. Consequently, cryptographic applications in blockchain systems are also being re-evaluated in the post-quantum setting. For example, the Ethereum Foundation [2] is exploring the integration of post-quantum signature schemes into the Ethereum platform.

Blockchain-based cryptocurrencies such as Bitcoin and Ethereum enable mutually distrustful users to reach consensus on the balances and the transactions that affect them. In general, there are two models of blockchain: the Unspent Transaction Output (UTXO) model and the account-based model. Both models have been extensively studied for their potential to support privacy-preserving payments, offering confidentiality and anonymity features.

In the UTXO model, many privacy-preserving blockchain payment protocols have been proposed, including ZCash [3], Monero [4] (with a series of RingCT protocols [5]–[11]), and Quisquis [12]. Among them, only MatRiCT [10] and MatRiCT$^+$ [11] operate in the post-quantum setting, where the communication cost per transaction is about 50–110 KB at an anonymity level of $1/11$. [1] On the other hand, the first privacy-preserving payment protocol for account-based blockchains was proposed in [13], named Zether, which provided a blueprint for achieving privacy and anonymity. Building on this, Anonymous Zether [14] refined the underlying zero-knowledge proof and identified potential insider attacks. To mitigate these, they introduced a "register" phase, requiring each user to prove the well-formedness of their public key. This line of work has since attracted significant attention and has been extended in various directions [15]–[17]. As discussed in [12], [14], [15], account-based protocols offer advantages in terms of wallet efficiency, as users only need to maintain their private key and account information to transact—unlike UTXO-based systems, which require scanning the entire transaction history. Moreover, account-based protocols support richer functionalities such as sealed-bid auctions and stake voting, as demonstrated in [13].

To the best of our knowledge, there is currently no private payment scheme for account-based blockchains in the post-quantum setting. The main reason lies in the absence of essential building blocks, as we discuss later.

Paradigm of Anonymous Zether. We first recall the paradigm of Anonymous Zether [13], [14]. At a high level, Anonymous Zether consists of three core components:

- *A multi-message multi-recipient public key encryption (mmPKE) scheme* [18], [19] that supports unbounded-level additive homomorphism, verifiable multi-encryption, and verifiable decryption. [2] In general, mmPKE enables the batch encryption of multiple messages for multiple recipients in a single operation, significantly reducing bandwidth compared to the trivial approach of encrypting each message individually. This efficiency also applies to verifiable multi-encryption in mmPKE. This capability is critical in Anonymous Zether, as it can significantly reduce the communication size of a transaction—especially

1. The anonymity level indicates that each real account is hidden within an anonymity set of size N.

2. We note that [13], [14] implicitly employ mmPKE, i.e., they directly utilize ElGamal-based mmPKE [18], [19] as a fundamental building block.

important in lattice-based post-quantum settings, where ciphertexts and proofs are typically large.

- *An event-oriented linkable ring signature* [20], [21] (also known as a prefix/scoped linkable ring signature [22], [23]) that enables signers to anonymously produce at most one signature per event using their long-term private keys.
- *A highly modular non-interactive zero-knowledge (NIZK) proof system* that integrates the verifiable mmPKE, the event-oriented linkable ring signature, the balance proof, and the range proof.

In Anonymous Zether, the balance bal of each account is encrypted as a ciphertext $\mathsf{acc}[\mathsf{pk}] \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{bal})$ which can be indexed by the public key pk. We outline how the sender makes a transaction as follows:

- First, the sender selects a set of public keys $(\mathsf{pk}_i)_{i\in[N]}$ as the anonymity set, including her own public key $\mathsf{pk}_{\ell_\mathsf{s}}$ and the recipient's public key $\mathsf{pk}_{\ell_\mathsf{r}}$.
- Second, the sender performs a verifiable multi-encryption of the amount vector as $\mathbf{ct} := (\mathsf{ct}_i)_{i\in[N]} \leftarrow \mathsf{Enc}((\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]})$, where $\mathsf{m}_{\ell_\mathsf{s}} = -\mathsf{amt}$, $\mathsf{m}_{\ell_\mathsf{r}} = \mathsf{amt}$, $\mathsf{m}_i = 0$ for all $i \in [N] \setminus \{\ell_\mathsf{s}, \ell_\mathsf{r}\}$, and the amount $\mathsf{amt} \in [0, \mathsf{MAX}]$. The security of mmPKE ensures that no one—including the decoy users in the anonymity set—can determine which ciphertext encrypts a non-zero message, thereby preventing identification of the sender's or recipient's public key.
- Third, the sender proves via verifiable decryption that the updated balance of her account is non-negative: $\mathsf{bal}'_{\ell_\mathsf{s}} \geq 0 \leftarrow \mathsf{Dec}(\mathsf{acc}[\mathsf{pk}_{\ell_\mathsf{s}}] + \mathsf{ct}_{\ell_\mathsf{s}}, \mathsf{sk}_{\ell_\mathsf{s}})$. This ensures that the sender is not overdrawn, i.e., her initial balance satisfies $\mathsf{bal}_{\ell_\mathsf{s}} \geq \mathsf{amt}$.
- Fourth, the sender anonymously authorizes the transaction by generating an event-oriented linkable ring signature for the current epoch H, [3] which can be linked using a tag tag_H. Specifically, the sender must prove knowledge of her private key $\mathsf{sk}_{\ell_\mathsf{s}}$ in the anonymity set $(\mathsf{pk}_i)_{i\in[N]}$ and demonstrate that the tag tag_H is correctly formed.
- Finally, the sender outputs a proof Π attesting that all the above conditions hold, along with the multi-recipient ciphertext $\mathbf{ct}$ and the linkable tag tag_H.

Once the proof Π is verified, [4] the system updates all accounts in the anonymity set by homomorphically evaluating $\mathsf{acc}[\mathsf{pk}_i] := \mathsf{acc}[\mathsf{pk}_i] + \mathsf{ct}_i$ for all $i \in [N]$. [5]

Challenges. However, in attempting to build a post-quantum account-based private blockchain payment protocol following the paradigm of Anonymous Zether [13], [14], we identify several significant challenges, as summarized below.

First of all, the only existing mmPKE scheme based on lattices, proposed in [24], does not support additive homomorphism and lacks efficient mechanisms for verifiable multi-encryption and decryption. In general, lattice-based (mm)PKE schemes do not support unbounded-level additive homomorphic evaluation without resorting to expensive operations such as bootstrapping [25]. This limitation is inherent to the lattice-based setting, as noise accumulates with each homomorphic operation. Moreover, the message encoding techniques employed in lattice-based RingCT protocols [10], [11] to avoid homomorphic evaluation are not applicable in our setting, as the sender cannot compute the so-called "corrector" terms without knowing the recipient's balance. Therefore, designing a lattice-based mmPKE scheme that enables unbounded-level additive homomorphic evaluation in an efficient manner—while also supporting efficient range proofs, verifiable multi-encryption, and verifiable decryption—has been a significant open problem.

Second, to the best of our knowledge, there exists no *practical* lattice-based event-oriented linkable ring or group signature that supports *multiple* events (i.e., tags). Existing constructions typically support only a single event or a very limited number of events, as in [26], where the resulting tag takes the form $\mathsf{tag} := \mathbf{A}\mathbf{s}$, with $\mathbf{A} \leftarrow \mathsf{hash}(\mathsf{event})$ and the secret $\mathbf{s}$ committed in advance. The main limitation arises from the contradiction between the need for fresh noise (to ensure indistinguishability from random values) and the determinism required for tag linkability. Specifically, to argue indistinguishability between ℓ tags $(\mathbf{A}_i \cdot \mathbf{s})_{i\in[\ell]}$ and random values, the secret $\mathbf{s}$ must include at least ℓ fresh noise components and be committed in advance—leading to a cost *linear* in the number of tags or events, which is significantly inefficient. Therefore, constructing a *practical* lattice-based event-oriented linkable ring or group signature that supports multiple events has been a significant challenge.

Finally, how to efficiently combine all of the above components—including a balance proof—within a unified lattice-based NIZK framework has remained unknown.

1.1. Contribution

In this work, we propose *Lether*, the *first* practical post-quantum account-based private blockchain payment system based on lattice assumptions. To build such a system, we develop two novel building blocks: a verifiable refreshable additively-homomorphic mmPKE (Ref-AH mmPKE) scheme and a plug-and-play event-oriented linkable tag scheme. Both tools may be of independent interest. We summarize our main contributions below. For technical overview, we refer the reader to Section 1.2.

Lether: a novel post-quantum account-based private blockchain payment system. Our primary contribution is the design of *Lether* using the proposed building blocks. To accommodate the constraints of the lattice-based setting, we introduce a lightweight *refresh* mechanism: each fresh or refreshed account supports up to t transactions involving additive homomorphic evaluation. Once this threshold is reached, the account can be refreshed, re-enabling support for another t transactions. This mechanism is particularly well-suited for blockchain environments, providing an efficient

3. In Anonymous Zether [13], [14], time is divided into epochs. A user is allowed to send at most one transaction per epoch. This "locking" mechanism mitigates double-spending attacks. As discussed in [13], [14], with a carefully chosen epoch length, usability is expected to remain largely unaffected. We refer readers to [13], [14] for further details.

4. In Anonymous Zether [13], [14], it is assumed that the proof Π is verified before the end of the current epoch.

5. As discussed in (Anonymous) Zether [13], [14], updating accounts at the end of the epoch is to prevent front-running issue in asynchronous environments such as blockchain systems.

alternative to bootstrapping or large-modulus schemes in FHE. Specifically, in our setting, the amortized communication cost per refresh, $|\mathsf{ref}|/t$, and the amortized proving and verification time per refresh, $(t_p + t_v)/t$, are only about 1.3% and 1.5%, respectively, of the cost of a transaction. The overall performance of *Lether* is summarized in Table 1.

Verifiable Ref-AH mmPKE. We construct the *first* verifiable Ref-AH mmPKE scheme from lattices, extending the basic mmPKE in [24]. Our scheme supports t-level additive homomorphism, meaning that a fresh ciphertext can undergo up to t homomorphic additions. We further introduce a *refresh* operation, which allows a user to convert a fully evaluated ciphertext into a fresh one using their private key—thereby enabling the refresh mechanism in *Lether*.

We formalize and realize three types of verifiability for mmPKE: *verifiable multi-encryption*, *verifiable decryption*, and *verifiable refresh*, where the sizes of the resulting proofs and ciphertexts are the primary contributors to the transaction communication overhead.

For verifiable multi-encryption, as shown in Table 2, our construction outperforms the state-of-the-art in terms of communication size for $N = 16$ recipients, achieving an order-of-magnitude reduction.[6] Moreover, our scheme uniquely supports both additive homomorphism and decryption-time independence, ensuring that ciphertexts—including those generated by adversaries—can be additively evaluated and subsequently decrypted efficiently by honest users.

For verifiable decryption, we introduce a new *unforgeability* notion, which strengthens the standard soundness definitions in prior work [30]–[32]. Informally, unforgeability ensures that no adversary can generate two valid proofs for different decryption outputs of the same ciphertext under a legitimate private key. We also present a generic transformation that upgrades existing schemes to satisfy this stronger notion by incorporating a proof of knowledge of the private key corresponding to the associated public key. As shown in Table 3, our construction achieves at least a 40% reduction in proof size while simultaneously supporting additive homomorphism and exact norm proofs for the decryption error—both crucial for maximizing the level of homomorphic evaluations.

For verifiable refresh, we construct the scheme by combining verifiable encryption and verifiable decryption in a way that achieves unforgeability. Furthermore, our parameter choices ensure that the evaluation level t is close to optimal, even for adversarially generated ciphertexts. This is reflected in the system performance shown in Table 1.

Plug-and-play event-oriented linkable tag scheme. We propose a plug-and-play, event-oriented linkable tag scheme from lattices that can be used to transform most existing lattice-based ring or group signature schemes including [10], [11], [28], [34]–[37] to support event-oriented linkability—the most general form of linkability—with *multiple* tags (i.e., events), and with *almost negligible* overhead. To the best of our knowledge, no existing lattice-based ring/group signature scheme supports such property.

6. Our scheme also demonstrates a similar advantage over related constructions such as [27].

We demonstrate the effectiveness of our technique by extending the state-of-the-art lattice-based group signature scheme from [28] to support event-oriented linkability. As shown in Table 4, our extended scheme increases the signature size by only about 1%, while enabling support for multiple tags.

New formal definitions for account-based private blockchain payments. As an additional contribution, we propose a new formal framework for account-based private blockchain payment protocols. Our definitions aim to balance the complexity of real-world blockchain systems with the abstraction required for rigorous security analysis, improving upon previous attempts [13], [14]. Due to space constraints, we present the syntax in Appendix B.

1.2. Technical Overview

In this subsection, we provide an overview of our techniques. We begin by showing how to extend the basic mmPKE scheme [24] to support additive homomorphism and refreshability, and how to realize its verifiability using the lattice-based NIZK [28], referred to as LNP22. We then present the construction of a plug-and-play event-oriented linkable tag scheme using the proof of rounding technique [38], instantiated via LNP22. Finally, we outline how these components are integrated to construct Lether.

Ref-AH mmPKE. We introduce a new property of (mm)PKE called *refreshability*, and propose a Ref-AH mmPKE from lattices, extending the scheme of [24]. We begin by recalling the basic mmPKE construction [24].

In the setup, the public matrix is sampled as $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m\times n})$ where $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ and $\mathcal{R} = \mathbb{Z}[X]/(X^d+1)$. Each public key $\mathbf{b}_i$ for $i \in [N]$ is generated by

$$\mathbf{b}_i := \mathbf{A}^\top \mathbf{s}_i + \mathbf{e}_i \tag{1.1}$$

where $(\mathbf{s}_i, \mathbf{e}_i) \leftarrow \mathcal{U}(\mathbb{S}_\nu^m) \times \mathcal{U}(\mathbb{S}_\nu^n)$ are uniformly sampled from $[-\nu, ..., \nu]$. To encrypt messages $(\hat{m}_i \in \mathcal{R}_2)_{i\in[N]}$ for multiple recipients $(\mathbf{b}_i)_{i\in[N]}$, the ciphertext $(\mathbf{c}, (c_i)_{i\in[N]})$ is computed as

$$\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u, \tag{1.2}$$

$$c_i = \langle \mathbf{b}_i, \mathbf{r}\rangle + y_i + \lfloor q/2 \rceil \cdot \hat{m}_i, \tag{1.3}$$

where $(\mathbf{r}, \mathbf{e}_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$ and $y_i \leftarrow \mathcal{D}_{\sigma_1}$. Decryption is computed as $\lfloor c_i - \langle \mathbf{c}, \mathbf{s}\rangle \rceil_2$. Using Equations (1.1) to (1.3), we obtain:

$$c_i - \langle \mathbf{c}, \mathbf{s}\rangle = \langle -\mathbf{s}_i || \mathbf{e}_i, \mathbf{e}_u || \mathbf{r}\rangle + y_i + \lfloor q/2 \rceil \cdot \hat{m}_i, \tag{1.4}$$

where $||$ denotes concatenation. Correctness of decryption holds when the decryption error $h_i := \langle -\mathbf{s}_i || \mathbf{e}_i, \mathbf{e}_u || \mathbf{r}\rangle + y_i$ satisfies $\|h_i\|_\infty \leq \lfloor q/4 \rceil$.

The basic scheme [24] only supports binary messages in $\{0,1\}^d$, and therefore cannot handle additive homomorphism. To address this limitation, we assume without loss of generality that the integer message $m \in [0, \mathsf{MAX}]$, where $\mathsf{MAX} = 2^k - 1$ and $k \leq d$. We then encode the integer message m in binary form $\hat{m} \in \mathcal{R}_2$ satisfying $m = \langle \vec{0}^{d-k} \,\|\, \vec{2}^k, \vec{\hat{m}}\rangle$, where $\vec{2}^k = (2^0, \ldots, 2^{k-1})$ and $\vec{\hat{m}}$

Scheme	Anony.	Register				Transaction				Refresh				Eval. Level
	N	$\lvert\mathsf{reg}\rvert$	$\lvert\pi\rvert$	t_p	t_v	$\lvert\mathsf{tx}\rvert$	$\lvert\Pi\rvert$	t_p	t_v	$\lvert\mathsf{ref}\rvert$	$\lvert\pi'\rvert$	t_p	t_v	T
Lether	16	37.2	27.5	0.12	0.08	67.8	51.1	0.53	0.34	53.5	41.3	0.42	0.32	61

Table 1: Results in Lether scheme. Sizes and times are reported in kilobytes (KB) and seconds (s), respectively. For the **Register**, **Transaction**, and **Refresh** phases, we report their communication sizes, denoted as $|\mathsf{reg}|$, $|\mathsf{tx}|$, and $|\mathsf{ref}|$, respectively. We also report the sizes of the associated proofs ($|\pi|$, $|\Pi|$, and $|\pi'|$) that are already included in the communication sizes, as well as the corresponding proving time t_p and verification time t_v. The anonymity set size is set to $N := 16$ (**Anony.**), matching the setting used in Monero [4]. The evaluation level for additively homomorphic operations on fresh or refreshed accounts is set to $T := 61$ (**Eval. Level**).

Scheme	Recipients N	$\lvert\mathbf{ct}\rvert$ (KB)	$\lvert\pi\rvert$ (KB)	Add-Hom	Dec-Ind
Cons 2.8	16	**17**	**28**	✓	✓
[28]	16	16	304	✗	✓
[29]	16	144	144	✓	✗

Table 2: Comparison with the existing verifiable encryption schemes for recipients $N = 16$. We report the size of multi-recipient ciphertext $|\mathbf{ct}|$ and its proof size $|\pi|$. We also report whether the schemes support additive homomorphism (**Add-Hom**) and decryption time independence (**Dec-Ind**).

Scheme	$\lvert\pi\rvert$	Add-Hom	Exa-Pro
Cons 2.11	**31** KB	✓	✓
[30]	≈ 50 KB	✗	✓
[31]	≥ 1 MB*	✗	✗
[32]	≈ 50 KB†	✗	✗

* We choose the same security parameter, i.e. $\lambda = 128$
† This work is distributed verifiable decryption and the undistributed version is implied in [33]

Table 3: Comparison with the existing verifiable decryption schemes. For a fair comparison, we extend other works to unforgeability and estimate the proof size $|\pi|$. We also report whether the schemes support additive homomorphism (**Add-Hom**) and exact norm proof (**Exa-Pro**).

Scheme	Signature Size	Event-Link	Multi-Tag
Cons 2.20 + [28]	93 KB	✓	✓
[28]	92 KB	✗	✗
[10]	148 KB	✗	✗
[26]	386 KB	✓	✗

Table 4: Comparison with the existing group signature schemes for group size over 2^{20}. We report the signature size of each scheme and whether the schemes support event-oriented linkability (**Event-Link**) in the case of multiple tags (**Multi-Tag**).

is an integer vector consisting of the coefficients of $\hat{m}$. Next, we extend the message space to $\mathcal{M} := \{-t, \dots, t+1\}^d$ by modifying the ciphertexts as follows:

$$c_i = \langle \mathbf{b}_i, \mathbf{r} \rangle + y_i + \lfloor q/(2t+2) \rceil \cdot \hat{m}_i. \tag{1.5}$$

Homomorphic addition is then performed component-wise. For example,

$$m_2 = \langle \vec{0}^{d-k} \,\|\, \vec{2}^k, \vec{\hat{m}}_0 + \vec{\hat{m}}_1 \rangle = m_0 + m_1,$$

and the same principle applies to subtraction. Thus, the scheme supports up to t levels of additive homomorphism, as long as the accumulated noise remains within required bounds.

However, unbounded-level homomorphism cannot be achieved due to possible message overflow and the accumulation of noise beyond the correctness bound $\|h_i\|_\infty \geq \lfloor q/(4t+4) \rceil$. To address this, we introduce a *refresh* mechanism. Roughly speaking, each recipient decrypts the evaluated ciphertext to obtain $\hat{m} \in \mathcal{R}_{2t+2}$, then decodes the message to $m := \langle \vec{0}^{d-k} \| \vec{2}^k, \vec{\hat{m}} \rangle$. The message is re-encoded into binary form and re-encrypted with fresh randomness.

To ensure the refreshed ciphertext is honestly generated—i.e., the message is consistent and randomness is fresh—we define a new primitive called *verifiable refresh*, which we describe later.

Verifiability in Ref-AH mmPKE. We introduce verifiability for our Ref-AH mmPKE, including verifiable multi-encryption, verifiable decryption, and verifiable refresh.

In our work, we employ LNP22 [28] as a black-box to achieve these verifiability properties. At a high level, LNP22 supports proving linear and quadratic relations over both $\mathcal{R}_q$ and $\mathbb{Z}_q$ with respect to the witness. It also enables both exact and approximate range proofs (ARP) for the ℓ_2 norm of linear combinations of the witness. As shown in [28], [39] and summarized in Section A.3, these capabilities can be extended to prove binary bits, polynomials with binary coefficients, and ARPs for ℓ_∞-norms.

To construct a verifiable multi-encryption, we must prove that Equations (1.2) and (1.5) hold with $\|(\mathbf{r}, \mathbf{e})\|_\infty \leq \beta_0$, $\|(y_1\| \cdots \|y_N)\|_\infty \leq \beta_1$, and $m_i \in \mathcal{R}_2$. We define the witness as $\mathsf{wit} := (\mathbf{r}, (\hat{m}_i)_{i\in[N]})$ and the statement as $\mathsf{stat} := ((\mathbf{b}_i)_{i\in[N]}, (\mathbf{c}, (c_i)_{i\in[N]}))$. This yields the relation $\|(\mathbf{r}, \mathbf{c} - \mathbf{A}\mathbf{r})\|_\infty \leq \beta_0$ and

$$\left\| \begin{matrix} c_1 - \langle \mathbf{b}_1, \mathbf{r} \rangle - \lfloor q/(2t+2) \rceil \cdot \hat{m}_1 \\ \vdots \\ c_N - \langle \mathbf{b}_N, \mathbf{r} \rangle - \lfloor q/(2t+2) \rceil \cdot \hat{m}_N \end{matrix} \right\|_\infty \leq \beta_1$$

assuming that the modulus of the proof system equals the mmPKE modulus.

Notably, the ℓ_∞-norms of the randomness values—particularly y_i—can be quite large. Proving the exact ℓ_∞-norms of these values via binary decomposition would be costly. Moreover, we cannot prove their exact ℓ_2-norms either, since the magnitude of $\|y_i\|_2$ could violate the soundness requirements of LNP22 under the same modulus. To resolve this, we rely on ARP and ensure compliance with the soundness conditions in [28, Lemma 2.9]. Thus, we define $\beta_0 := \psi \cdot \sqrt{(m+n)d} \cdot \tau\sigma_0$ and $\beta_1 := \psi \cdot \sqrt{Nd} \cdot \tau\sigma_1$, where ψ is the relaxation factor and τ is the Gaussian tail bound for ℓ_2-norm in [40, Lemma 4.4]. These bounds will also be used to derive the decryption error bound and the supported level of homomorphic evaluation.

Next, we turn to verifiable decryption. We introduce a stronger security notion, called *unforgeability*, which guarantees the uniqueness of the decrypted message under the correct private key. This property is absent in prior works but is crucial in real-world settings, particularly under lattice assumptions. To achieve this, we add a proof of knowledge of the private key corresponding to the public key, i.e., $\|(\mathbf{s}, \mathbf{b}-\mathbf{A}^\top\mathbf{s})\|_\infty \le \nu$. This approach can also be applied to existing schemes [30]–[32] to enhance them with unforgeability.

To construct the verifiable decryption, we prove Equation (1.4), i.e.,

$$c - \langle \mathbf{c}, \mathbf{s}\rangle = h + \lfloor q/(2t+2)\rceil \cdot \hat{m}, \tag{1.6}$$

with $\|h\|_\infty \le \lfloor q/(4t+4)\rceil$. Since the ℓ_∞ bound on h is large—close to the mmPKE modulus—we cannot use ARP when the proof system shares the same modulus. Instead, we employ bit decomposition to prove the exact ℓ_∞-norm of h. Specifically, let $\beta := \lfloor q/(4t+4)\rceil$, and decompose h into a binary polynomial vector $\mathbf{b}_h \in \mathcal{R}_2^k$, with $o := \lceil \log(\beta+1)\rceil$ and $h = \sum_{i\in[o]} \delta_i b_h^{(i)}$, where $\boldsymbol{\delta} := (1, 2, ..., \beta + 1 - 2^{\lfloor \log \beta \rfloor})$ is a "gadget" vector. The proof then shows that $\mathbf{b}_h$ is binary and satisfies Equation (1.6).

Finally, we describe the construction of a verifiable refresh. At a high level, it combines verifiable decryption and multi-encryption with a consistency check on the message. Rather than directly revealing the decrypted message $\hat{m}$ during verifiable decryption, we first prove it in range $[-t, t+1]$ using its binary decomposition $\mathbf{b}_m$. Then, we show $\langle \vec{0}^{d-k}||\vec{2}^k, \vec{m}\rangle = \langle \vec{0}^{d-k}||\vec{2}^k, \vec{m}'\rangle$ and $\hat{m}' \in \mathcal{R}_2$. Last, we use $\hat{m}'$ for the subsequent (verifiable) encryption.

Plug-and-Play Event-Oriented Linkable Tag Scheme. Inspired by the lattice-based construction of verifiable random function (VRF) [38], we outline how to construct a plug-and-play event-oriented linkable tag scheme using the proof of rounding and instantiated via $\mathsf{LNP22}$.

The tag scheme outputs a pair (π, tag), where π is a proof of knowledge of a private key used to generate both a public key and a linkable tag tag for a specific event. In our construction, the tag tag is defined as

$$\mathbf{v}_H = \lfloor \mathbf{A}_H \cdot \mathbf{s} \mod \hat{q} \rceil_{\hat{p}}, \tag{1.7}$$

where $\mathbf{A}_H \in \mathcal{R}_{\hat{q}}^{n'\times m} \leftarrow \mathsf{hash}(\mathsf{event})$ is computed from the event string, and $\mathbf{s}$ is the private key from the mmPKE in Equation (1.1). Here, we require that $\nu \ll \hat{q}$ such that $\|\mathbf{s}\|_\infty \le \nu$ and that $\hat{p}$ divides $\hat{q}$

As shown in [38], such a tag $\mathbf{v}_H$ is computationally indistinguishable from a random value under the Module Learning with Rounding (MLWR) assumption. Moreover, a single private key can generate a (practically) unbounded number of tags (e.g., more than 2^{128}) for different events with suitable parameters. The non-frameability of the tag—i.e., the inability of an adversary to produce a valid proof for another user's tag without knowing the corresponding private key—is ensured by the Module Short Integer Solutions (MSIS) assumption.

We now explain how to instantiate the proof of rounding in Equation (1.7) via $\mathsf{LNP22}$. We rewrite Equation (1.7) as

$$\hat{q}/\hat{p} \cdot \mathbf{v}_H \bmod \hat{q} \equiv \mathbf{A}_H \cdot \mathbf{s} - \mathbf{e}_H \bmod \hat{q}, \tag{1.8}$$

where $\|(\mathbf{s}, \mathbf{b}-\mathbf{A}^\top\mathbf{s})\|_\infty \le \nu$ and $\mathbf{e}_H \in \{0, ..., \hat{q}/\hat{p}-1\}^{n'd}$.

Since $\hat{q}$ is not a prime and does not match the modulus q of the proof system or mmPKE, we cannot directly prove this relation over modulus $\hat{q}$. To overcome this, we require that $\hat{q} \ll q$ and transform Equation (1.8) into the following relation over modulus q (which is equivalent to the relation over the integers):

$$\hat{q}/\hat{p} \cdot \mathbf{v}_H - \mathbf{A}_H \cdot \mathbf{s} + \mathbf{e}_H + \hat{q} \cdot \mathbf{v} = 0,$$

where $\|\mathbf{v}\|_\infty \le \beta$, and $\beta := (\hat{q}/\hat{p}\cdot\hat{p} + dm\hat{q}\nu + \hat{q}/\hat{p} - 1)/\hat{q}$.

Therefore, the proof of rounding can be efficiently realized using $\mathsf{LNP22}$ by showing the conditions: $\|(-\hat{q}/\hat{p}\cdot\mathbf{v}_H + \mathbf{A}_H\cdot\mathbf{s} - \mathbf{e}_H)/\hat{q}\|_\infty \le \psi\cdot\beta$, $\|(\mathbf{s}, \mathbf{b} - \mathbf{A}^\top\mathbf{s})\|_\infty \le \nu$, $\mathbf{e}_H \in \{0, ..., \hat{q}/\hat{p}-1\}^{n'd}$, where ψ is the relaxation factor used in ARP.

Overview of Lether. At a high level, Lether follows the paradigm of Anonymous Zether [13], [14], combining our verifiable Ref-AH mmPKE and our event-oriented linkable ring signature, unified through $\mathsf{LNP22}$. The general structure of the system is as follows.

Each user generates a public-private key pair $(\mathbf{b}, \mathbf{s})$ of mmPKE and registers in the system by submitting $(\mathbf{b}, \pi)$, where π is a proof of knowledge of the private key $\mathbf{s}$ corresponding to $\mathbf{b}$, generated using $\mathsf{LNP22}$.

Each account $(\mathbf{u}, v)$ indexed by the associated public key $\mathbf{b}$ is initialized with balance $m = 0$ or funded with an amount $m \in [0, \dots, \mathrm{MAX}]$ as an individual ciphertext:

$$\mathbf{u} := \mathbf{A}\mathbf{r} + \mathbf{e}_u, \quad v := \langle \mathbf{b}, \mathbf{r}\rangle + y + \lfloor q/(2t+2)\rceil \cdot (\hat{m} + t_d),$$

where $\hat{m}$ is a binary polynomial such that $\langle \vec{0}^{d-k}||\vec{2}^k, \vec{m}\rangle = m$, and t_d is a shift factor, i.e., a polynomial with all coefficients equal to the constant t. For compatibility, we shift the message space of our mmPKE from $\{-t, \dots, t+1\}^d$ to $\{0, \dots, 2t+1\}^d$ by adding t_d to the encoded balance.

To transact an amount $m \in [0, \dots, \mathrm{MAX}]$, the sender selects an anonymous set of public keys $(\mathbf{b}_i)_{i\in[N]}$, including her own $\mathbf{b}_{\ell_\mathsf{s}}$ and the recipient's $\mathbf{b}_{\ell_\mathsf{r}}$, to hide both identities.

The sender then selects two binary indicator vectors $\mathbf{b}^{(\mathsf{s})}, \mathbf{b}^{(\mathsf{r})} \in \{0,1\}^N$, where $b_{\ell_\mathsf{s}}^{(\mathsf{s})} = 1$ and $b_{\ell_\mathsf{r}}^{(\mathsf{r})} = 1$, with all other entries set to zero. These vectors are used to index the sender and recipient in the transaction proof Π.

Specifically, the sender verifiably encrypts the amount m into a ciphertext $(\mathbf{c}, (c_i)_{i\in[N]})$ for $(\mathbf{b}_i)_{i\in[N]}$ via our mmPKE as $\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u$ and

$$c_i := \langle \mathbf{b}_i, \mathbf{r}\rangle + y_i + \lfloor q/(2t+1)\rceil \cdot (b_i^{(\mathsf{r})} - b_i^{(\mathsf{s})}) \cdot \hat{m},$$

where $\hat{m} \in \mathcal{R}_2$ satisfies $\langle \vec{0}^{d-k}||\vec{2}^k, \vec{m}\rangle = m$.

To prevent overdraft attacks, the sender must provide a verifiable decryption showing that the resulting balance $\hat{m}'$ of her account $\sum_{i=1}^N b_i^{(\mathsf{s})} \cdot (\mathbf{u}_i + \mathbf{c}, v_i + c_i)$ is non-negative. Specifically, it must satisfy $\langle \vec{0}^{d-k}||\vec{2}^k, \vec{m}' - \vec{t}_d\rangle \in [0, \mathrm{MAX}]$ and $\hat{m}' \in [0, 2t+1]^d$.

Finally, to authorize the transaction, the sender generates a ring signature proving knowledge of the private key $\mathbf{s}_{\ell_{\mathsf{s}}}$ corresponding to the public key $\sum_{i=1}^{N} b_i^{(\mathsf{s})} \cdot \mathbf{b}_i$. To ensure double-spending prevention, the sender also provides a proof of rounding for the linkable tag $\mathbf{v}_H := \lfloor \mathbf{A}_H \cdot \mathbf{s}_{\ell_{\mathsf{s}}} \mod \hat{q} \rceil_{\hat{p}}$ where $\mathbf{A}_H \leftarrow \mathsf{hash}(\mathrm{Lether}||H)$ and H is the current epoch obtained from the blockchain state.

To manage additive homomorphism depth, the system maintains a counter for each account, tracking the number of transactions since its last refresh or initialization. Once this counter reaches t, all subsequent transactions are paused until the user performs a verifiable refresh, after which the counter is reset.

2. Novel Building Blocks of Lether

In this section, we develop the novel building blocks of Lether, including a verifiable Ref-AH mmPKE scheme and a plug-and-play event-oriented linkable tag scheme. Due to space constraints, we defer the formal preliminaries to Appendix A.

2.1. Verifiable Ref-AH mmPKE

We propose the *first* lattice-based verifiable refreshable additively-homomorphic mmPKE. We begin by formally defining the notion of refreshable additively-homomorphic mmPKE (Ref-AH mmPKE), and then proceed to describe its verifiability.

Definition 2.1 (Ref-AH mmPKE)**.** A Ref-AH mmPKE scheme with a public-private key pair space $\mathcal{K}$, a message space $\mathcal{M}$, a multi-recipient ciphertext space $\mathcal{C}$, and an individual ciphertext space $\mathcal{C}_{\mathsf{s}}$ consists of the following algorithms.

- $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$: On input a security parameter 1^λ and a recipient number N, it outputs a public parameter $\mathsf{pp}_{\mathsf{Enc}}$ (which is an implicit input to all remaining algorithms).
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mmKGen}()$: It outputs a public-private key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{K}$.
- $\mathbf{ct} := (\widehat{\mathsf{ct}}, (\widehat{\mathsf{ct}}_i)_{i\in[N]}) \leftarrow \mathsf{mmEnc}((\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]}; \mathsf{r}, (\mathsf{r}_i)_{i\in[N]})$: On input N public keys $(\mathsf{pk}_i)_{i\in[N]}$, N messages $(\mathsf{m}_i)_{i\in[N]}$, $(N+1)$ randomnesses r, $(\mathsf{r}_i)_{i\in[N]}$, it outputs the multi-recipient ciphertext $\mathbf{ct} := (\widehat{\mathsf{ct}}, (\widehat{\mathsf{ct}}_i)_{i\in[N]})$.
- $\mathsf{ct}_i := (\widehat{\mathsf{ct}}, \widehat{\mathsf{ct}}_i)/\perp \leftarrow \mathsf{mmExt}(i, \mathbf{ct})$: On input a multi-recipient ciphertext $\mathbf{ct} \in \mathcal{C}$, and an index $i \in [N]$, it deterministically outputs the individual ciphertext $\mathsf{ct}_i \in \mathcal{C}_{\mathsf{s}}$ or a symbol $\perp$ to indicate extraction failure.
- $\mathsf{m}/\perp \leftarrow \mathsf{mmDec}(\mathsf{sk}, \mathsf{ct})$: On input a private key sk, and an individual ciphertext $\mathsf{ct} \in \mathcal{C}_{\mathsf{s}}$, it outputs a message $\mathsf{m} \in \mathcal{M}$ or a symbol $\perp$ to indicate decryption failure.
- $\mathsf{ct}'/\perp \leftarrow \mathsf{mmRef}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$: On input a public-private key pair $(\mathsf{pk}, \mathsf{sk})$, and an individual ciphertext $\mathsf{ct} \in \mathcal{C}_{\mathsf{s}}$, it outputs a refreshed individual ciphertext $\mathsf{ct}' \in \mathcal{C}_{\mathsf{s}}$ or a symbol $\perp$ to indicate refresh failure.

Remark 2.2 (Additive Homomorphism). Let t be a positive integer. We say an mmPKE scheme is t-level additively-homomorphic, if for any $\lambda, N \in \mathbb{N}$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda)$, any $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mmKGen}()$ and $\mathsf{m}_i^{(j)} \in \mathcal{M}$ for all $i \in [N]$, $j \in [t]$, the following holds,

$$\Pr\left[\begin{array}{c} \forall i \in [N],\ \mathsf{ct}_i^{(j)} \leftarrow \mathsf{mmExt}(i, \mathbf{ct}^{(j)}), \\ \mathsf{mmDec}(\mathsf{sk}_i, \mathsf{ct}_i^{(0)} \oplus \cdots \oplus \mathsf{ct}_i^{(t-1)}) \\ = \mathsf{m}_i^{(0)} + \cdots + \mathsf{m}_i^{(t-1)} : \\ \mathbf{ct}^{(j)} \leftarrow \mathsf{mmEnc}((\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i^{(j)})_{i\in[N]}) \end{array}\right] = 1.$$

where $\oplus$ denotes the additively-homomorphic evaluation.

Remark 2.3 (Refreshability). Suppose an mmPKE scheme is t-level additively homomorphic. We say that the mmPKE is refreshable, if for any $\lambda, N \in \mathbb{N}$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mmKGen}()$, any individual ciphertext $\mathsf{ct} \in \mathcal{C}_{\mathsf{s}}$ encrypted under pk and evaluated additively at most t times, there exists a PPT algorithm $\mathsf{ct}' \leftarrow \mathsf{mmRef}((\mathsf{pk}, \mathsf{sk}), \mathsf{ct})$ such that $\mathsf{mmDec}(\mathsf{sk}, \mathsf{ct}') = \mathsf{mmDec}(\mathsf{sk}, \mathsf{ct})$ and ct' is a fresh ciphertext that supports at least t further additive homomorphic evaluations with other fresh ciphertexts.

The definitions of correctness and security for Ref-AH mmPKE follow those of the basic mmPKE [24]. Therefore, we omit the formal definitions due to space constraints and refer the reader to [24] for further details.

As shown in [24], under the *knowledge-of-secret-key* (KOSK) assumption—i.e., that the challenger knows the adversary's private keys—the chosen-plaintext attack security of (Ref-AH) mmPKE ($\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$) ensures that an adversary cannot distinguish the challenge multi-recipient ciphertext, even if it is one or several of the recipients. Furthermore, with the KOSK compiler in [24], the KOSK assumption can be eliminated if each recipient proves knowledge of their private key corresponding to their public key.

Table 5: Summary of main notations used in Lether, including our mmPKE and tag scheme.

Notation	Description
λ	security parameter
ζ	correctness parameter
N	# of recipients, # of anonymous set
m, n	# of rows of $\mathbf{A}$, # of columns of $\mathbf{A}$
n'	dimension of linkable tag $\mathbf{v}_H$
q	modulus in mmPKE
d	ring dimension of $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$
ν	ℓ_∞-norm bound on private key $(\mathbf{s}_i, \mathbf{e}_i)$
$\bar{\nu}$	support size $\bar{\nu} \leq 2\nu + 1$ of private key $(\mathbf{s}_i, \mathbf{e}_i)$
σ_0	Gaussian width of $(\mathbf{r}, \mathbf{e}_u)$ in the ciphertext
σ_1	Gaussian width of y_i in the ciphertext
$\hat{q}, \hat{p}$	moduli in tag scheme
t	levels of additively-homomorphic evaluation in mmPKE
k	# of bits of the message and the amount.

Construction 2.4 (Ref-AH mmPKE)**.** Let λ be a security parameter. Let m, n, d, q, N, ν, t be positive integers. Let σ_0, σ_1 be Gaussian widths. For the message space $\mathcal{M} = [0, 2^k - 1]$, our refreshable t-level additively-homomorphic mmPKE is shown in Algorithm 1. mmExt is defined by picking $(\mathbf{c}, c_i)$ from $(\mathbf{c}, (c_i)_{i\in[N]})$. We summarize the notations in Table 5.

Our mmPKE scheme shares the same correctness and security analyses as [24]; we therefore only state the following lemmas.

Theorem 2.5 (Correctness)**.** *Let $\mathbf{e}_i, \mathbf{s}_i, \mathbf{r}, \mathbf{e}_u, y_i$ be random variables that have the corresponding distribution as in*

Algorithm 1 Ref-AH mmPKE

```
procedure mmSetup($1^\lambda, N$)
    $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m\times n})$
    return $\mathsf{pp}_{\mathsf{Enc}} := \mathbf{A}$
end procedure

procedure mmKGen()
    $\mathbf{s}, \mathbf{e} \leftarrow \mathcal{U}(\mathbb{S}_\nu^m) \times \mathcal{U}(\mathbb{S}_\nu^n)$
    $\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e}$
    return $(\mathsf{pk} := \mathbf{b}, \mathsf{sk} := \mathbf{s})$
end procedure

procedure mmEnc($(\mathsf{pk}_i = \mathbf{b}_i)_{i\in[N]}, (\mathsf{m}_i = m_i \in \mathbb{Z}_{2^k})_{i\in[N]}$)
    $\mathbf{r}, \mathbf{e}_u \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$
    $\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u$
    for all $i \in [N]$
        $y_i \leftarrow \mathcal{D}_{\sigma_1}$
        $\hat{m}_i \in \mathcal{R}_2 \leftarrow \mathsf{BinPoly}(m_i)$
        $c_i := \langle \mathbf{b}_i, \mathbf{r}\rangle + y_i + \lfloor q/(2t+2)\rceil \cdot \hat{m}_i$
    end for
    return $\mathbf{ct} := (\mathbf{c}, (c_i)_{i\in[N]})$
end procedure

procedure mmDec($\mathsf{ct} = (\mathbf{c}, c), \mathsf{sk} = \mathbf{s}$)
    $\hat{m} \leftarrow \lfloor c - \langle \mathbf{c}, \mathbf{s}\rangle \rceil_{2t+2}$
    return $\mathsf{m} := \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}\rangle$
end procedure

procedure mmRef($\mathsf{ct} = (\mathbf{c}, c), (\mathsf{pk}, \mathsf{sk}) = (\mathbf{b}, \mathbf{s})$)
    $m \leftarrow \mathsf{mmDec}((\mathbf{c}, c), \mathbf{s})$
    return $\mathsf{ct}' := (\mathbf{c}', c') \leftarrow \mathsf{mmEnc}(\mathbf{b}, m)$
end procedure
```

Construction 2.4. Denote ζ as

$$\sum_{i\in[N]} \Pr\left[\ \|\langle \mathbf{e}_i, \mathbf{r}\rangle + y_i - \langle \mathbf{s}_i, \mathbf{e}_u\rangle\|_\infty \geq \lfloor q/(4t+4)\rceil\ \right].$$

We say our Ref-AH mmPKE in Construction 2.4 is ζ-correct.

Theorem 2.6 (Security)**.** *Define the distribution $\chi := \mathcal{U}(\mathbb{S}_\nu)$, $\chi_0 := \mathcal{D}_{\mathbb{Z}^{(m+n+1)d}, \sqrt{\Sigma_1}}$, and $\chi_1 := \mathcal{D}_{\mathbb{Z}^{Nd}, \sqrt{\Sigma_{\mathbf{y}}}}$, where $\Sigma_1 = \begin{pmatrix} \sigma_1 I_d & \mathbf{0} \\ \mathbf{0} & \sigma_0 I_{(m+n)d} \end{pmatrix}$, $\Sigma_{\mathbf{y}} = \sigma_1 I_{Nd}$. Define the distribution $\mathcal{S}$ such that the matrix $\mathbf{R} \leftarrow \mathcal{S}$ can be sampled as $\mathbf{R} = \begin{pmatrix} 0 & -\mathbf{s}_0^\top & \mathbf{e}_0^\top \\ \vdots & \vdots & \vdots \\ 0 & -\mathbf{s}_{N-1}^\top & \mathbf{e}_{N-1}^\top \end{pmatrix} \in \mathcal{R}^{N\times(1+m+n)}$ where $\mathbf{s}_i \leftarrow \mathcal{U}(\mathbb{S}_\nu^n)$, $\mathbf{e}_i \leftarrow \mathcal{U}(\mathbb{S}_\nu^m)$ for each $i \in [N]$.*

Our Ref-AH mmPKE in Construction 2.4 is $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ secure under $\mathsf{MLWE}_{\mathcal{R},n,m,q,\chi}$ and $\mathsf{MatrixHint\text{-}MLWE}_{\mathcal{R},m,n,q,\chi_0}^{N,\chi_1,\mathcal{S}}$ assumptions.

The properties of additive homomorphism and refreshability follow directly. For appropriate parameter settings (i.e., when the accumulated noise remains within bounds), our Ref-AH mmPKE supports at least t levels of additive homomorphism.

Furthermore, as mentioned earlier, the KOSK assumption can be removed by requiring each recipient to prove knowledge of their private key during a registration phase, specifically demonstrating that $\|(\mathbf{s}, \mathbf{b} - \mathbf{A}^\top \mathbf{s})\|_\infty \leq \nu$. Note that, since here the number of recipients N is polylogarithmic in the security parameter, we can directly apply LNP22 without concern for exponential soundness degradation, as discussed in [24], [41].

We now present the three types of verifiability supported by our Ref-AH mmPKE scheme: verifiable multi-encryption, verifiable decryption, and verifiable refresh.

Verifiable multi-encryption for mmPKE. The verifiable multi-encryption for (Ref-AH) mmPKE is a batched verifiable encryption that offers significant savings in both bandwidth and computation, compared to the naive approach of applying separate PKE and NIZK for each recipient. As in standard verifiable encryption, it produces a proof π to guarantee the well-formedness of the multi-recipient ciphertext $\mathbf{ct}$.

In general, a verifiable (multi-)encryption scheme must satisfy three properties: completeness, simulatability, and soundness. Completeness requires that any honestly generated ciphertext, along with its corresponding proof, is always accepted by the verifier. Simulatability ensures the existence of a simulator such that no adversary can distinguish between real and simulated ciphertext. Soundness guarantees that no adversary can convince the verifier of a false ciphertext. We formalize the soundness property as follows.

Definition 2.7 (Soundness in Verifiable Multi-Encryption)**.** Let $\mathsf{mmPKE} = (\mathsf{mmSetup}, \mathsf{mmKGen}, \mathsf{mmEnc}, \mathsf{mmDec}, \mathsf{mmExt})$ be an mmPKE scheme for message space $\mathcal{M}$. Let $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ be a verifiable multi-encryption scheme for mmPKE. We say that Π is sound if for any PPT adversary $\mathcal{A}$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$, any $\mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mmKGen}(\mathsf{pp})$ for all $i \in [N]$, the following probability is negligible in λ,

$$\Pr\left[\begin{array}{c} \Pi.\mathsf{Verify}((\mathsf{pk}_i)_{i\in[N]}, \mathbf{ct}, \pi) = 1\ \wedge \\ \exists i \in [N],\ \mathsf{ct}_i \leftarrow \mathsf{mmExt}(\mathbf{ct}, i),\ \text{such that} \\ \mathsf{mmDec}(\mathsf{ct}_i, \mathsf{sk}_i) = \bot\ \vee\ \mathsf{mmDec}(\mathsf{ct}_i, \mathsf{sk}_i) \notin \mathcal{M} : \\ (\pi, \mathbf{ct}) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_\Pi, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}) \end{array}\right].$$

Then, using $\mathsf{LNP22} = (\mathsf{LNP22.Setup}, \mathsf{LNP22.P}, \mathsf{LNP22.V})$ as a black-box NIZK proof system, we construct our verifiable multi-encryption scheme for our Ref-AH mmPKE as follows.

Construction 2.8 (Verifiable Multi-Encryption for mmPKE)**.** After generating the multi-recipient ciphertext $(\mathbf{c}, (c_i)_{i\in[N]})$ using $\mathsf{mmEnc}((\mathbf{b}_i)_{i\in[N]}, (m_i)_{i\in[N]}; (\mathbf{r}, \mathbf{e}_u), (y_i)_{i\in[N]})$, the prover takes as input the witness $\mathsf{wit} := ((\hat{m}_i)_{i\in[N]}, \mathbf{r})$ and the statement $\mathsf{stat} := (\mathbf{A}, (\mathbf{b}_i)_{i\in[N]}, (\mathbf{c}, (c_i)_{i\in[N]}))$. The proof is generated as $\pi \leftarrow \mathsf{LNP22.P}(R_{\text{enc}}, \mathsf{stat}, \mathsf{wit})$ and verified by $0/1 \leftarrow \mathsf{LNP22.V}(R_{\text{enc}}, \mathsf{stat}, \pi)$, where the proof relation R_{enc} is defined as follows:

$$R_{\text{enc}} = \left\{\begin{array}{l} (\mathbf{A}, (\mathbf{b}_i)_{i\in[N]}, (\mathbf{c}, (c_i)_{i\in[N]})); ((\hat{m}_i)_{i\in[N]}, \mathbf{r}) : \\ \left\|\begin{matrix} \mathbf{r} \\ \mathbf{c} - \mathbf{A}\mathbf{r} \end{matrix}\right\|_\infty \leq \psi \cdot \beta_0, \\ \left\|\begin{matrix} c_1 - \langle \mathbf{b}_1, \mathbf{r}\rangle - \lfloor \frac{q}{2t+2}\rceil \cdot \hat{m}_1 \\ \vdots \\ c_N - \langle \mathbf{b}_N, \mathbf{r}\rangle - \lfloor \frac{q}{2t+2}\rceil \cdot \hat{m}_N \end{matrix}\right\|_\infty \leq \psi \cdot \beta_1, \\ (\hat{m}_1 || \cdots || \hat{m}_N) \in \{0,1\}^{Nd} \end{array}\right\}, \quad (2.1)$$

where $\psi = 189$, $\beta_0 = \sqrt{(m+n)d} \cdot \tau\sigma_0$, $\beta_1 = \sqrt{Nd} \cdot \tau\sigma_1$, and each $\hat{m}_i \leftarrow \mathsf{BinPoly}(m_i)$ denotes the binary decomposition of message m_i for all $i \in [N]$.

The completeness and simulatability of our verifiable multi-encryption follow directly from those of $\mathsf{LNP22}$. We establish its soundness through the following lemma.

Lemma 2.9 (Soundness in Verifiable Multi-Encryption)**.** *Suppose* $\mathsf{LNP22}$ *is knowledge sound. Then, our verifiable multi-encryption in Construction 2.8 is sound if the following probability is negligible,*

$$\sum_{i\in[N]} \Pr[\ \|\langle \mathbf{e}_i, \bar{\mathbf{r}}\rangle + \bar{y}_i - \langle \mathbf{s}_i, \bar{\mathbf{e}}_u\rangle\|_\infty \geq \lfloor q/(4t+4)\rceil\ ] \tag{2.2}$$

where $(\mathbf{e}_i, \mathbf{s}_i)$ *is the private key in mmPKE and* $(\bar{\mathbf{r}}, \bar{\hat{m}}_i)$ *is the extracted witness along with* $\bar{\mathbf{e}}_u := \mathbf{c} - \mathbf{A}\bar{\mathbf{r}}$, $\bar{y}_i := c_i - \langle \mathbf{b}_i, \bar{\mathbf{r}}\rangle - \lfloor q/(2t+2)\rceil \cdot \bar{\hat{m}}_i$.

Proof. If $\mathsf{LNP22}$ is knowledge sound, there exists a PPT extractor that can extract the witness $(\bar{\mathbf{r}}, (\bar{\hat{m}}_i)_{i\in[N]})$ satisfying the relation R_{enc} in Equation (2.1).

Denote $\bar{\mathbf{e}}_u := \mathbf{c} - \mathbf{A}\bar{\mathbf{r}}$ and $\bar{y}_i := c_i - \langle \mathbf{b}_i, \bar{\mathbf{r}}\rangle - \lfloor q/(2t+2)\rceil \cdot \bar{\hat{m}}_i$. The value $(\mathbf{c}, c_i)$ in mmDec algorithm is

$$\mathbf{c} = \mathbf{A}\bar{\mathbf{r}} + \bar{\mathbf{e}}_u, \qquad c_i = \langle \mathbf{b}_i, \bar{\mathbf{r}}\rangle + \bar{y}_i + \lfloor q/(2t+2)\rceil \cdot \bar{\hat{m}}_i.$$

Since the public key $\mathbf{b}_i = \mathbf{A}^\top \mathbf{s}_i + \mathbf{e}_i$, we can obtain

$$c_i - \langle \mathbf{c}, \mathbf{s}_i\rangle = \langle \mathbf{e}_i, \bar{\mathbf{r}}\rangle + \bar{y}_i - \langle \mathbf{s}_i, \bar{\mathbf{e}}_u\rangle + \lfloor q/(2t+2)\rceil \cdot \bar{\hat{m}}_i.$$

Therefore, the decryption will fail if the decryption error $\|\langle \mathbf{e}_i, \bar{\mathbf{r}}\rangle + \bar{y}_i - \langle \mathbf{s}_i, \bar{\mathbf{e}}_u\rangle\|_\infty \leq \lfloor q/(4t+4)\rceil$. □

Verifiable decryption for mmPKE. The verifiable decryption for (Ref-AH) mmPKE follows the same structure as that for standard PKE. Here, we introduce a stronger security notion, called *unforgeability*, which enhances traditional notion of soundness. At a high level, unforgeability guarantees that decrypting the ciphertext under a given public key with its corresponding private key yields the message m.

The completeness of verifiable decryption requires that a proof generated for an honestly decrypted message is always accepted. Simulatability ensures the existence of a simulator such that no adversary can distinguish between real and simulated decrypted messages. We formalize unforgeability as follows.

Definition 2.10 (Unforgeability in Verifiable Decryption)**.** Let $\mathsf{mmPKE} = (\mathsf{mmSetup}, \mathsf{mmKGen}, \mathsf{mmEnc}, \mathsf{mmDec}, \mathsf{mmExt})$ be an mmPKE scheme. Let $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ be a verifiable decryption scheme for mmPKE.

We say that Π is unforgeable if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$, any $\mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, the following probability is negligible in λ,

$$\Pr\left[\begin{array}{c} \Pi.\mathsf{Verify}(\mathsf{pk}_i, \mathsf{ct}_i, \mathsf{m}'_i, \pi) = 1\ \wedge\ \mathsf{m}'_i \neq \mathsf{m}_i : \\ ((\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_\Pi); \\ \mathbf{ct} \leftarrow \mathsf{mmEnc}((\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]}); \\ (\pi, \mathsf{m}'_i) \leftarrow \mathcal{A}_1(\mathbf{ct}, \mathsf{st});\ \mathsf{ct}_i \leftarrow \mathsf{mmExt}(\mathbf{ct}, i) \end{array}\right].$$

Then, using $\mathsf{LNP22} = (\mathsf{LNP22.Setup}, \mathsf{LNP22.P}, \mathsf{LNP22.V})$ as a black-box NIZK proof system, we construct the verifiable decryption scheme for our Ref-AH mmPKE as follows.

Construction 2.11 (Verifiable Decryption for mmPKE)**.** Suppose $\nu = 1, \bar{\nu} = 2$. After decrypting the individual ciphertext $(\mathbf{c}, c)$ under the public key $\mathbf{b}$ using $\mathsf{mmDec}((\mathbf{c}, c), \mathbf{s})$ to the encoded message $\hat{m}$ and obtain the decryption error h, the prover takes as input the the witness $\mathsf{wit} := (\mathbf{s}, \mathbf{b}_h)$ and the statement $\mathsf{stat} := (\mathbf{A}, \mathbf{b}, (\mathbf{c}, c), \hat{m})$, where $\mathbf{b}_h \leftarrow \mathsf{Bit}(h)$ is the coefficient-wise binary decomposition of h. The proof is generated as $\pi \leftarrow \mathsf{LNP22.P}(R_{\text{dec}}, \mathsf{stat}, \mathsf{wit})$ and verified by $0/1 \leftarrow \mathsf{LNP22.V}(R_{\text{dec}}, \mathsf{stat}, \pi)$, where the proof relation R_{dec} is defined as follows:

$$R_{\text{dec}} = \left\{\begin{array}{l} (\mathbf{A}, \mathbf{b}, (\mathbf{c}, c), m); (\mathbf{s}, \mathbf{b}_h) : \\ c - \langle \mathbf{c}, \mathbf{s}\rangle - \lfloor q/(2t+2)\rceil \cdot m = \sum_{i\in[o]} \delta_i \cdot b_h^{(i)}, \\ (\mathbf{s}||(\mathbf{b} - \mathbf{A}^\top \mathbf{s})||\mathbf{b}_h) \in \{0,1\}^{(m+n+o)\cdot d} \end{array}\right\} \tag{2.3}$$

where $o = \lceil \log(\lfloor q/(4t+4)\rceil)\rceil$, $\mathbf{b}_h = (b_h^{(1)}, ..., b_h^{(o)})$, $\boldsymbol{\delta} = (\delta_1, \delta_2, ..., \delta_o) = (1, 2, ..., \lfloor q/(4t+4)\rceil - 2^{\lfloor \log(q/(4t+4)-1)\rfloor})$.

The completeness and simulatability of our verifiable multi-encryption follow directly from those of $\mathsf{LNP22}$. We establish its unforgeability through the following lemma.

Lemma 2.12 (Unforgeability in Verifiable Decryption)**.** *Suppose* $\mathsf{LNP22}$ *is knowledge sound. Then, our verifiable decryption in Construction 2.11 is unforgeable if* $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ *assumption is hard.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that can generate a valid proofs π for different message $m \neq m'$. If $\mathsf{LNP22}$ is knowledge sound, there exists a PPT extractor that can extract the witness $(\bar{\mathbf{s}}, \bar{\mathbf{b}}_h)$ from the proofs satisfying the relation R_{dec} in Equation (2.3).

Denote $\bar{\mathbf{e}} := \mathbf{b} - \mathbf{A}^\top \bar{\mathbf{s}}$ and $\mathbf{e} := \mathbf{b} - \mathbf{A}^\top \mathbf{s}$. We first argue that $\bar{\mathbf{s}} = \mathbf{s}$. Otherwise, the adversary can break the $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ assumption for the instance of $[\mathbf{I}|\mathbf{A}]$ with the solution $[(\bar{\mathbf{e}} - \mathbf{e})|(\bar{\mathbf{s}} - \mathbf{s})]^\top$.

Denote $\bar{h} := \sum_{i=1}^{o} \delta_i \cdot \bar{b}_h^{(i)}$ and

$$h := c - \langle \mathbf{c}, \bar{\mathbf{s}}'\rangle - \lfloor q/(2t+2)\rceil \cdot m'. \tag{2.4}$$

Thus, we have $\bar{h}, h < \lfloor q/(4t+4)\rceil$ and $\|\bar{h} - h\|_\infty < \lfloor q/(2t+2)\rceil$. Next, we can obtain from R_{dec}:

$$c - \langle \mathbf{c}, \bar{\mathbf{s}}\rangle - \lfloor q/(2t+2)\rceil \cdot m = \bar{h}, \tag{2.5}$$

After subtracting Equation (2.5) and Equation (2.4), we can obtain $\lfloor q/(2t+2)\rceil \cdot (m - m') = h - \bar{h}$. Since $m \neq m'$ and $\|\lfloor q/(2t+2)\rceil \cdot (m - m')\|_\infty \geq \lfloor q/(2t+2)\rceil$, it contradicts the soundness of $\mathsf{LNP22}$. □

Verifiable Refresh for mmPKE. After at most t additively homomorphic evaluations, one can decrypt a ciphertext $(\mathbf{c}, c)$ to obtain the message m, and then re-encrypt it into a fresh ciphertext $(\mathbf{c}', c')$ using fresh randomness $(\mathbf{r}', \mathbf{e}'_u) \leftarrow \mathcal{D}^n_{\sigma_0} \times \mathcal{D}^m_{\sigma_1}$. Informally, verifiable refresh ensures that the message embedded in the refreshed ciphertext is identical to that in the evaluated ciphertext, and that the randomness used in the refreshed ciphertext is fresh (i.e., short).

The completeness of verifiable refresh requires that a proof generated for an honestly refreshed ciphertext is always accepted. Simulatability guarantees the existence of a simulator such that no adversary can distinguish between real and simulated refreshed ciphertexts. We formalize unforgeability as follows.

Definition 2.13 (Unforgeability in Verifiable Refresh)**.** Let $\mathsf{mmPKE} = (\mathsf{mmSetup}, \mathsf{mmKGen}, \mathsf{mmEnc}, \mathsf{mmDec}, \mathsf{mmExt})$ be an mmPKE scheme. Let $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ be a verifiable refresh scheme for mmPKE.

We say that Π is unforgeable if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$, any $\mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mmKGen}(\mathsf{pp})$ for all $i \in [N]$ the following probability is negligible in λ,

$$\Pr\left[\begin{array}{c} \Pi.\mathsf{Verify}(\mathsf{pk}_i, \mathsf{ct}_i, \mathsf{ct}'_i, \pi) = 1 \wedge \\ (\mathsf{mmDec}(\mathsf{sk}_i, \mathsf{ct}'_i) = 0 \vee \mathsf{mmDec}(\mathsf{sk}_i, \mathsf{ct}'_i) \neq \mathsf{m}_i) : \\ ((\mathsf{m}_i)_{i\in[N]}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_\Pi, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}); \\ \mathbf{ct} \leftarrow \mathsf{mmEnc}((\mathsf{pk}_i)_{i\in[N]}, (\mathsf{m}_i)_{i\in[N]}); \\ (\pi, \mathsf{ct}'_i) \leftarrow \mathcal{A}_1(\mathbf{ct}, \mathsf{st});\ \mathsf{ct}_i \leftarrow \mathsf{mmExt}(\mathbf{ct}, i) \end{array}\right].$$

Then, using $\mathsf{LNP22} = (\mathsf{LNP22.Setup}, \mathsf{LNP22.P}, \mathsf{LNP22.V})$ as a black-box NIZK proof system, we construct our verifiable refresh scheme for our Ref-AH mmPKE as follows.

Construction 2.14 (Verifiable Refresh for mmPKE)**.** Suppose $\nu = 1$ and $\bar{\nu} = 2$. The verifiable refresh is shown in Algorithm 2. The proof is verified by $0/1 \leftarrow \mathsf{LNP22.V}(R_{\text{ref}}, \mathsf{stat}, \pi)$, where the proof relation R_{ref} is defined as follows:

$$R_{\text{ref}} = \left\{\begin{array}{l} \mathsf{stat} = (\mathbf{A}, \mathbf{b}, (\mathbf{c}, c), (\mathbf{c}', c')); \\ \mathsf{wit} = (\mathbf{s}, \mathbf{r}', \hat{m}, \hat{m}', \mathbf{b}_m, \mathbf{b}_h) : \\ c - \langle \mathbf{c}, \mathbf{s}\rangle - \lfloor q/(2t+2)\rceil \cdot \hat{m} = \sum_{i\in[o]} \delta_i b_h^{(i)}, \\ \left\| \begin{matrix} \mathbf{r}' \\ \mathbf{c}' - \mathbf{A}\mathbf{r}' \end{matrix} \right\|_\infty \le \psi \cdot \beta_0, \\ \|c' - \langle \mathbf{b}, \mathbf{r}'\rangle - \lfloor q/(2t+2)\rceil \cdot (\hat{m}' + t_d)\|_\infty \le \psi \cdot \beta_1, \\ \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}} - \vec{t}_d\rangle = \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}'\rangle, \\ \hat{m} = \sum_{i\in[o']} \delta'_i \cdot b_m^{(i)}, \\ \mathsf{bin} \in \{0,1\}^{(m+n+o'+o+1)d} \end{array}\right\} \tag{2.6}$$

where $\psi = 189$, $\mathsf{bin} := (\mathbf{s}||(\mathbf{b} - \mathbf{A}^\top\mathbf{s})||\mathbf{b}_m||\mathbf{b}_h||\hat{m}')$, $\mathbf{b}_m = (b_m^{(1)}, ..., b_m^{(o')})$, $\boldsymbol{\delta}' = (\delta'_1, \delta'_2 ..., \delta'_{o'}) = (1, 2, ..., 2t + 2 - 2^{\lfloor \log(2t+1)\rfloor})$, and $\boldsymbol{\delta}$ along with $\mathbf{b}_h$ are the same as in Equation (2.3).

To fit the setting in Lether, as mentioned earlier, the message space is defined as $\mathcal{M} = [0, \dots, 2^k - 1]$ and the encoding space is shifted from $\{-t, \dots, t+1\}$ to $\{0, \dots, 2t+1\}^d$ by adding the constant offset t_d.

Algorithm 2 $\mathsf{VRefresh}$

```
Input: $\mathsf{pk} = \mathbf{b}$, $\mathsf{sk} = \mathbf{s}$, $\mathsf{ct} = (\mathbf{c}, c)$
 1: $\hat{m} := \lfloor c - \langle \mathbf{c}, \mathbf{s}\rangle \rceil_{2t+2}$
 2: $h := c - \langle \mathbf{c}, \mathbf{s}\rangle - \lfloor q/(2t+2)\rceil \cdot \hat{m}$
 3: $\mathbf{b}_m \leftarrow \mathsf{Bit}(\hat{m})$
 4: $\mathbf{b}_h \leftarrow \mathsf{Bit}(h)$
 5: $m := \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}} - \vec{t}_d\rangle$
 6: $\hat{m}' \leftarrow \mathsf{BinPoly}(m)$
 7: $(\mathbf{r}', \mathbf{e}'_u) \leftarrow \mathcal{D}^n_{\sigma_0} \times \mathcal{D}^m_{\sigma_0}$
 8: $y' \leftarrow \mathcal{D}_{\sigma_1}$
 9: $\mathbf{c}' := \mathbf{A}\mathbf{r}' + \mathbf{e}'_u$
10: $c' := \langle \mathbf{b}, \mathbf{r}'\rangle + y' + \lfloor q/(2t+2)\rceil \cdot (\hat{m}' + t_d)$
11: $\pi' \leftarrow \mathsf{LNP22.P}(R_{\text{ref}}, \mathsf{stat}, \mathsf{wit})$ where $R_{\text{ref}}$ is defined in Equation (2.6)
12: return $\mathsf{ref} = (\mathsf{pk} := \mathbf{b}, \mathsf{ct}' := (\mathbf{c}', c'), \pi')$
```

The completeness and simulatability of our verifiable multi-encryption follow directly from those of LNP22. We establish its unforgeability through the following lemma.

Lemma 2.15 (Unforgeability in Verifiable Refresh)**.** *Suppose* LNP22 *is knowledge sound. Then, our verifiable refresh in Construction 2.14 is unforgeable if* $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ *assumption is hard and the following probability is negligible,*

$$\Pr[\ \|\langle \mathbf{e}, \bar{\mathbf{r}}'\rangle + \bar{y}' - \langle \mathbf{s}, \bar{\mathbf{e}}'_u\rangle\| \ge \lfloor q/(4t+4)\rceil\ ] \tag{2.7}$$

where $(\mathbf{e}, \mathbf{s})$ *is the private key in mmPKE and* $(\bar{\mathbf{r}}', \bar{\hat{m}}')$ *is the extracted witness along with* $\bar{\mathbf{e}}'_u := \mathbf{c}' - \mathbf{A}\bar{\mathbf{r}}'$, $\bar{y}' := c' - \langle \mathbf{b}, \bar{\mathbf{r}}'\rangle - \lfloor q/(2t+2)\rceil \cdot \bar{\hat{m}}'$.

Proof sketch. The proof combines the arguments from Lemma 2.9 and Lemma 2.12. Briefly speaking, the unforgeability of the verifiable decryption ensures that the decrypted message $\hat{m}$ is valid, while the soundness of the verifiable multi-encryption guarantees that the refreshed ciphertext is well-formed. Moreover, the relation $\hat{m} \in \{0, \dots, 2t+1\}^d$ together with $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}} - \vec{t}_d\rangle = \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}'\rangle$ ensures the consistency between the messages in the evaluated and refreshed ciphertexts. □

Parameter Setting. To parameterize our verifiable Ref-AH mmPKE scheme, we require the following properties to be satisfied:

- The Ref-AH mmPKE scheme must be mmIND- $\mathsf{CPA}^{\mathsf{KOSK}}$ secure, support t-level additive homomorphism, and achieve ζ-correctness for $\zeta \le 2^{-128}$.
- The verifiable multi-encryption must satisfy completeness, soundness, and simulatability.
- The verifiable decryption must satisfy completeness, unforgeability, and simulatability.
- The verifiable refresh must satisfy completeness, unforgeability, and simulatability.

We describe how to choose the parameters step by step in the following.

First, we set $\nu = 1$ and $\bar{\nu} = 2$ so that the proof of the private key can be efficiently generated using the binary proof of LNP22. For example, in the registration phase, each public key $\mathbf{b}_i$ is proven to satisfy $\mathbf{b}_i = \mathbf{A}^\top\mathbf{s}_i + \mathbf{e}_i$ where $(\mathbf{s}_i||\mathbf{e}_i) \in \{0,1\}^{(m+n)d}$.

Second, in the verifiable multi-encryption phase, the multi-recipient ciphertext $(\mathbf{c}, (c_i)_{i\in[N]})$ is proven to satisfy

$$\mathbf{c} = \mathbf{A}\mathbf{r} + \mathbf{e}_u, \quad c_i = \langle \mathbf{b}_i, \mathbf{r}\rangle + y_i + \lfloor q/(2t+1)\rceil \cdot m_i,$$

where $\|(\mathbf{r}, \mathbf{e}_u)\|_\infty \le \psi \cdot \sqrt{(m+n)d} \cdot \tau\sigma_0$ and $\|y_i\|_\infty \le \psi \cdot \sqrt{Nd} \cdot \tau\sigma_1$.

Third, to ensure the soundness of verifiable multi-encryption, the unforgeability of verifiable refresh, and to guarantee at least t-level additive homomorphism for a verified fresh or refreshed ciphertext, we must ensure that the accumulated decryption error remains below the bound $\lfloor q/(4t+4)\rceil$. That is, the following inequality should hold with overwhelming probability:

$$t \cdot \|\langle \mathbf{e}_i, \mathbf{r}\rangle - \langle \mathbf{s}_i, \mathbf{e}_u\rangle + y_i\|_\infty < \lfloor q/(4t+4)\rceil\,.$$

Fourth, we fix the parameters as $d = 64$, $\tau = 1.6$, $N = 16$, and $\psi = 189$. To ensure the security of our Ref-AH mmPKE and its verifiability, using the script from [24], we ensure that the hardness assumptions $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ for $\bar{\chi} := \mathcal{U}(\mathbb{S}_\nu)$, $\mathsf{MatrixHint\text{-}MLWE}^{N,\chi_1,\mathcal{S}}_{\mathcal{R},m+N,n,q,\chi_0}$, and $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ achieve 128-bit security. We obtain $\sigma_0 = 15.9$, $\sigma_1 = 30560$, $q \approx 2^{46}$, $m = 30$, $n = 26$, and $t = 61$.

Finally, we verify that the above parameters also satisfy the soundness conditions of LNP22 using the scripts provided in [28], [39]. Consequently, the resulting sizes are as follows: $|\mathbf{b}_i| = 9.34$ KB, $|\mathbf{c}| = 10.78$ KB, $|c_i| = 0.36$ KB, and $|\mathbf{ct}| = 16.53$ KB. The total communication cost of a verifiable refresh is given by $|\mathsf{ref}| = |\pi'| + |\mathbf{c}| + |c_i|$ and the total communication cost of registration is given by $|\mathsf{reg}| = |\pi| + |\mathbf{b}_i|$, which is a one-time cost for each user.

2.2. Plug-and-Play Event-Oriented Linkable Tag

In this subsection, we propose the *first* plug-and-play, event-oriented linkable tag scheme based on lattices. Our construction can directly enables a lattice-based event-oriented ring signature used in Lether, where the private key is derived from our Ref-AH mmPKE. We present the details of this ring signature in the next section.

Definition 2.16 (Tag Scheme)**.** A tag scheme with a public-private key pair space $\mathcal{K}$, a tag space $\mathcal{T}$ consists of the following algorithms.

- $(\mathsf{pp}_{\mathsf{tag}}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$: On input a security parameter, it outputs a public parameter (which is an implicit input to other algorithms) and a public-private key pair.
- $\mathsf{tag} \leftarrow \mathsf{TagGen}(\mathsf{sk}, \mathsf{event})$: On input a private key and an event, it outputs the linkability tag.
- $0/1 \leftarrow \mathsf{Link}(\mathsf{event}, \mathsf{tag}, \mathsf{tag}')$: On input an event and two tags, it outputs 1 if they are linked, and 0 otherwise.
- $\pi \leftarrow \mathsf{Prove}(\mathsf{pk}, \mathsf{tag}, \mathsf{event}, \mathsf{sk})$: On input the statement and witness, it proves knowledge of a private key that was used to create both a linkable tag for a specific event and a public key.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{pk}, \mathsf{tag}, \mathsf{event}, \pi)$: On input the statement and proof, it outputs 1 if the proof is valid, and 0 otherwise.

Then, we present the security model of our tag scheme. Following [22], we model the tag proof algorithm as a signature of knowledge, omitting the signed message for simplicity. In general, a signature of knowledge should satisfy completeness, simulatability, and extractability. Due to space constraints, we omit the formal definitions of these properties and refer the reader to [22], [42] for more details.

Furthermore, we formalize additional properties—namely, event-oriented linkability, multi-tag anonymity, and non-frameability—adapted from [23], as follows. In particular, we explicitly define anonymity in the setting where multiple tags are generated for different events.

Let $\mathsf{mmPKE} = (\mathsf{mmSetup}, \mathsf{mmKGen}, \mathsf{mmEnc}, \mathsf{mmDec}, \mathsf{mmExt})$ be an mmPKE scheme. Let $\mathsf{Tag} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{TagGen}, \mathsf{Link})$ be a tag scheme with a tag space $\mathcal{T}$ and a public-private key pair derived from that of mmPKE.

Definition 2.17 (Event-Oriented Linkability)**.** We say that Tag is event-oriented linkable if for any PPT adversary $\mathcal{A}$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$, any $\mathsf{pp}_{\mathsf{tag}} \leftarrow \mathsf{Setup}(1^\lambda)$, the following probability is negligible in λ,

$$\Pr\left[\begin{array}{c} \forall i \in \{0,1\},\ \mathsf{Verify}(\mathsf{pk}, \mathsf{event}, \mathsf{tag}_i, \pi_i) = 1 \\ \wedge\ \mathsf{Link}(\mathsf{event}, \mathsf{tag}_0, \mathsf{tag}_1) = 0 : \\ (\mathsf{event}, \mathsf{pk}, (\mathsf{tag}_i, \pi_i)_{i\in\{0,1\}}) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_{\Pi}) \end{array}\right].$$

Definition 2.18 (Multi-Tag Anonymity)**.** We say that Tag is multi-tag anonymous if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$, any $\mathsf{pp}_{\mathsf{tag}} \leftarrow \mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mmKGen}(\mathsf{pp})$, the following probability is negligible in λ,

$$\left|\Pr\left[\begin{array}{c} b = b' : \\ ((\mathsf{event}_i)_{i\in[\ell]}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_{\mathsf{tag}}, \mathsf{pk}); \\ \forall i \in [\ell],\ \mathsf{tag}^0_i \leftarrow \mathsf{TagGen}(\mathsf{sk}, \mathsf{event}_i),\ \mathsf{tag}^1_i \leftarrow \mathcal{T}; \\ b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}_1((\mathsf{tag}^b_i)_{i\in[\ell]}, \mathsf{st}) \end{array}\right] - \frac{1}{2}\right|.$$

Definition 2.19 (Non-Frameability)**.** We say that Tag is non-frameable if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $\mathsf{pp}_{\mathsf{Enc}} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$, any $\mathsf{pp}_{\mathsf{tag}} \leftarrow \mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{mmKGen}(\mathsf{pp})$, the following probability is negligible in λ,

$$\Pr\left[\begin{array}{c} \mathsf{Verify}(\mathsf{pk}', \mathsf{event}, \mathsf{tag}', \pi') = 1 \wedge \mathsf{tag}' = \mathsf{tag} : \\ (\mathsf{event}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_{\mathsf{tag}}, \mathsf{pk}); \\ \mathsf{tag} \leftarrow \mathsf{TagGen}(\mathsf{sk}, \mathsf{event}); \\ (\mathsf{tag}', \pi', \mathsf{pk}') \leftarrow \mathcal{A}_1(\mathsf{tag}, \mathsf{st}) \end{array}\right].$$

Then, using $\mathsf{LNP22} = (\mathsf{LNP22.Setup}, \mathsf{LNP22.P}, \mathsf{LNP22.V})$ as a black-box NIZK proof system, we construct our tag scheme for our Ref-AH mmPKE as follows.

Construction 2.20 (Tag Scheme for mmPKE)**.** Suppose $\nu = 1$ and $\bar{\nu} = 2$. Suppose $\hat{p}$ divides $\hat{q}$ and $\hat{q}/\hat{p} = 2$. Suppose $\hat{q} \ll q$, where q is the modulus for LNP22 and mmPKE.

- $\mathsf{Setup}(1^\lambda)$: It outputs a hash function $\mathsf{hash} : \{0,1\}^* \to \mathcal{R}^{n'\times m}_{\hat{q}}$ (modeled as a random oracle in the security analysis) and a private-public key pair $(\mathbf{b}, \mathbf{s}) \leftarrow \mathsf{mmKGen}()$.
- $\mathsf{TagGen}(\mathbf{s}, \mathsf{event})$: it outputs a linkable tag $\mathbf{v}_H := \lfloor \mathbf{A}_H \cdot \mathbf{s} \bmod \hat{q} \rceil_{\hat{p}}$, where $\mathbf{A}_H \leftarrow \mathsf{hash}(\mathsf{event})$.
- $\mathsf{Link}(\mathbf{v}, \mathbf{v}')$: It outputs 1 if $\mathbf{v}_H = \mathbf{v}'_H$, and 0 otherwise.
- Prove and Verify: Inputting the witness $\mathsf{wit} := (\mathbf{s}, \mathbf{e}_H)$, where $\mathbf{e}_H := \mathbf{A}_H \cdot \mathbf{s} - \hat{q}/\hat{p} \cdot \mathbf{v}_H \mod \hat{q}$ is the rounding error, and the statement $\mathsf{stat} := (\mathbf{A}_H, \mathbf{b}, \mathbf{v}_H)$, the tag proof is generated by $\pi \leftarrow \mathsf{LNP22.P}(R_{\mathsf{tag}}, \mathsf{stat}, \mathsf{wit})$ and verified by $0/1 \leftarrow \mathsf{LNP22.V}(R_{\mathsf{tag}}, \mathsf{stat}, \pi)$, where the proof relation R_{tag} is defined as:

$$R_{\mathsf{tag}} = \left\{\begin{array}{l} (\mathbf{A}_H, \mathbf{b}, \mathbf{v}_H); (\mathbf{s}, \mathbf{e}_H) : \\ \|(\hat{q}/\hat{p} \cdot \mathbf{v}_H + \mathbf{e}_H - \mathbf{A}_H \cdot \mathbf{s})/\hat{q}\|_\infty \le \psi \cdot \beta, \\ (\mathbf{s}\|(\mathbf{b} - \mathbf{A}^\top \mathbf{s})\|\mathbf{e}_H) \in \{0,1\}^{(m+n+n')d} \end{array}\right\} \tag{2.8}$$

where $\psi = 189$ and $\beta = \sqrt{n'd} \cdot (dm\hat{q}\nu + \hat{q}/\hat{p} - 1 + \hat{q}/\hat{p} \cdot \hat{p})/\hat{q}$.

As remarked in Section 1.2, we prove the rounding relation $\mathbf{v}_H = \lfloor \mathbf{A}_H \cdot \mathbf{s} \bmod \hat{q} \rceil_{\hat{p}}$ by showing that $\|(\hat{q}/\hat{p} \cdot \mathbf{v}_H + \mathbf{e}_H - \mathbf{A}_H \cdot \mathbf{s})/\hat{q}\|_\infty \le \psi \cdot \beta$, using LNP22 through a series of transformations.

The completeness, simulatability, and extractability of our tag proof algorithm follow directly from LNP22. We demonstrate the remaining properties through the following lemmas.

Lemma 2.21 (Event-Oriented Linkability in Tag Scheme)**.** *Suppose* LNP22 *is knowledge sound. Then our tag scheme in Construction 2.20 is event-oriented linkable if* $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ *assumption is hard.*

Proof. Suppose there exists an adversary $\mathcal{A}$ can generate two valid proofs π_0, π_0 along with different tags $\mathbf{v}_H^0 \neq \mathbf{v}_H^1$ for the same event, towards the same public key $\mathbf{b}$. Thus, if LNP22 is knowledge soundness, there exists a PPT extractor that can extract two witnesses $(\bar{\mathbf{s}}_0, \bar{\mathbf{e}}_H^0)$, $(\bar{\mathbf{s}}_1, \bar{\mathbf{e}}_H^1)$ satisfying the relation R_{tag} in Equation (2.8).

Similar to Lemma 2.12, we can argue that $\bar{\mathbf{s}}_0 = \bar{\mathbf{s}}_1$ holds if the $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ assumption is hard.

Denote $\bar{\mathbf{v}}_i := (\hat{q}/\hat{p} \cdot \mathbf{v}_H^i + \bar{\mathbf{e}}_H^i - \mathbf{A}_H \cdot \bar{\mathbf{s}}_i)/\hat{q}$ for $i \in \{0,1\}$. Thus, for $i \in \{0,1\}$, we have

$$\mathbf{A}_H \cdot \bar{\mathbf{s}}_i - \bar{\mathbf{e}}_H^i - \hat{q}/\hat{p} \cdot \mathbf{v}_H^i + \hat{q} \cdot \bar{\mathbf{v}}_i \mod q \equiv 0 \mod q.$$

Then, we moduli $\hat{q}$ for the both sides of the above equation to obtain, for $i \in \{0,1\}$

$$\mathbf{A}_H \cdot \bar{\mathbf{s}}_i - \bar{\mathbf{e}}_H^i - \hat{q}/\hat{p} \cdot \mathbf{v}_H^i \mod \hat{q} \equiv 0 \mod \hat{q}, \quad (2.9)$$

Later, we subtract Equation (2.9) for $i = \{0,1\}$ to get

$$\bar{\mathbf{e}}_H^1 - \bar{\mathbf{e}}_H^0 \mod \hat{q} \equiv \hat{q}/\hat{p} \cdot (\mathbf{v}_H^0 - \mathbf{v}_H^1) \mod \hat{q}.$$

Since $\mathbf{v}_H^0 \neq \mathbf{v}_H^1$, $\hat{q}/\hat{p} = 2$, and $\bar{\mathbf{e}}_H^0, \bar{\mathbf{e}}_H^1 \in \{0,1\}^{n'd}$, we have $\|\hat{q}/\hat{p} \cdot (\mathbf{v}_H^0 - \mathbf{v}_H^1)\|_\infty \geq \hat{q}/\hat{p}$ and $\|\bar{\mathbf{e}}_H^1 - \bar{\mathbf{e}}_H^0\|_\infty \leq \hat{q}/\hat{p} - 1$. Therefore, it contradicts the knowledge soundness of LNP22. □

Lemma 2.22 (Multi-Tag-Anonymity in Tag Scheme)**.** *Our tag scheme in Construction 2.20 is multi-tag-anonymous if* $\mathsf{MLWR}_{\mathcal{R},n,n',\hat{q},\hat{p},\chi}$ *assumption for* $\chi = \mathcal{U}(\mathbb{S}_\nu)$ *is hard.*

Proof Sketch. The tags are generated as $\mathbf{v}_H^i := \lfloor \mathbf{A}_H^i \cdot \mathbf{s} \mod \hat{q} \rceil_{\hat{p}}$ for $i \in [\ell]$, which are instances of the MLWR assumption. Therefore, as demonstrated in [38], when ℓ is polynomial (or even exponential) in λ, the honestly generated tags are indistinguishable from uniformly random values under the hardness of the MLWR assumption. □

Lemma 2.23 (Non-Frameability in Tag Scheme)**.** *Suppose* LNP22 *is knowledge sound. Then our tag scheme in Construction 2.20 is non-frameable if* $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ *assumption for* $\bar{\chi} := \mathcal{U}(\mathbb{S}_\nu)$ *and* $\mathsf{MSIS}_{\mathcal{R},n',(m+n'),\hat{q},\beta}$ *assumption for* $\beta = \max(\hat{q}/\hat{p} - 1, \nu)$ *are hard.*

Proof. Given the honestly-generated tag $\mathbf{v}_H$ for the event, suppose there exists an adversary $\mathcal{A}$ can generate a valid proofs π' with the same tag $\mathbf{v}_H$ for the same event, towards the different public key $\mathbf{b}'$. Thus, if LNP22 is knowledge soundness, there exists a PPT extractor that can extract a witness $(\bar{\mathbf{s}}', \bar{\mathbf{e}}'_H)$ satisfying the relation in Equation (2.8).

Therefore, we have

$$\mathbf{v}_H \mod \hat{q} \equiv \mathbf{A}_H \cdot \mathbf{s} + \mathbf{e}_H \mod \hat{q} \quad (2.10)$$

where $\mathbf{s}$ is the private key provided by the challenger bounded by $\|\mathbf{s}\|_\infty \leq \nu$ and $\mathbf{e}_H \in \{0, ..., \hat{q}/\hat{p} - 1\}^{n'd}$.

Since $\hat{q}/\hat{p} = 2$, $\nu = 1$, we have

$$\mathbf{v}_H \mod \hat{q} \equiv \mathbf{A}_H \cdot \bar{\mathbf{s}}' + \bar{\mathbf{e}}'_H \quad (2.11)$$

where $(\bar{\mathbf{s}}||\mathbf{e}_H) \in \{0,1\}^{(n+n')d}$.

Then, we subtract Equation (2.11) from Equation (2.10) to obtain a short solution $[\mathbf{s} - \bar{\mathbf{s}} \,|\, \mathbf{e}_H - \bar{\mathbf{e}}_H]$ to the $\mathsf{MSIS}_{\mathcal{R},m,(m+n),q,\nu}$ instance defined by $[\mathbf{A}_H \,|\, \mathbf{I}_{n'}]^\top$. If the solution $[\mathbf{s} - \bar{\mathbf{s}} \,|\, \mathbf{e}_H - \bar{\mathbf{e}}_H]$ is zero, then the adversary can break the $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ assumption. □

Parameter Setting. To parameterize our tag scheme, in addition to the requirements already specified for our verifiable Ref-AH mmPKE, we require the following additional properties to be satisfied:

- $\mathsf{MLWR}_{\mathcal{R},n,n',\hat{q},\hat{p},\chi}$ problem for $\chi = \mathcal{U}(\mathbb{S}_\nu)$ holds is hard at 128-bit security.
- $\mathsf{MSIS}_{\mathcal{R},n',(m+n'),\hat{q},\beta}$ problem for $\beta = \max(\hat{q}/\hat{p} - 1, \nu)$ is hard at 128-bit security.

We briefly outline the parameter selection process as follows.

First, we adopt the same parameters used in our mmPKE scheme, including $\nu = 1$, $m = 30$, $\bar{\nu} = 2$, $d = 64$, and $q \approx 2^{46}$. Here, we must ensure that $\hat{q} \ll q$, i.e., $q \geq (\psi+1) \cdot \beta$, where $\beta := (dm\hat{q}\nu + \hat{q}/\hat{p} - 1 + \hat{q}/\hat{p} \cdot \hat{p})/\hat{q}$.

Next, we set $\hat{q}/\hat{p} = 2$. To minimize the tag size, we choose $\hat{q} \approx 2^{11}$ and $\hat{p} \approx 2^{10}$. Following prior works [11], [28], [34], [38], we adopt a root Hermite factor (RHF) of approximately 1.0045 to estimate the practical hardness of MLWR and MSIS at the 128-bit security level, which yields $n' := 2$. This results in a tag size of $|\mathbf{v}_H| = 0.16$ KB. We defer the detailed discussion of the LNP22 parameters to the next section, as the tag proof will be integrated into the transaction proof there.

3. Lether: an Account-Based Private Blockchain Payment from Lattices

In this section, we provide the full details of *Lether*. Due to space constraints, we defer the definitions of *Lether* to Appendix B. Here, we specify the algorithms Setup, AddrGen, AnTransfer, and Verify as follows. The algorithms TagGen and AmtGen are invoked within AnTransfer, where TagGen corresponds to the tag scheme in Construction 2.20, and AmtGen is instantiated by mmEnc of the Ref-AH mmPKE in Construction 2.4.

The remaining algorithms are straightforward. The Register algorithm is from LNP22.V. The RollOver algorithm consists of mmExt and the additive homomorphic evaluation of Ref-AH mmPKE from Construction 2.4. The LinkTag algorithm is from the tag scheme in Construction 2.20. The ReadBalance algorithm is built using mmDec from Ref-AH mmPKE, followed by subtracting the constant offset $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{t}_d \rangle$.

When a fresh or refreshed account has undergone t homomorphic evaluations, its owner can refresh the account by running the VRefresh algorithm from Construction 2.14, which can then be publicly verified by the system.

Like (Anonymous) Zether [13], [14], we set $k := 32$ bits as the maximum value supported in the payment system, i.e., MAX $= 2^{32} - 1$. As noted in Zether [13], one can represent a large range of values by composing smaller ones—for example, using two 32-bit amounts to support 64-bit payments. To prevent integer overflow attacks, we ensure that the actual message space is larger than MAX.

Algorithm 3 initializes the system parameters. It first selects an anonymous set size $N \in \mathbb{N}$ and a hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{R}_{\hat{q}}^{n' \times m}$. Then, it generates the public parameters for the Ref-AH mmPKE scheme and for LNP22, and combines them to form the system-wide public parameters, which are treated as implicit inputs to the later algorithms.

Algorithm 3 $\mathsf{Setup}(1^\lambda)$ $\qquad$ λ is the security parameter

```
1: Choose an anonymous set size $N \in \mathbb{Z}_q$
2: Pick a hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{R}_{\hat{q}}^{n' \times m}$
3: $\mathsf{pp}_{\mathsf{Enc}} := \mathbf{A} \leftarrow \mathsf{mmSetup}(1^\lambda, N)$
4: $\mathsf{pp}_\Pi \leftarrow \mathsf{LNP22.Setup}(1^\lambda)$
5: return $\mathsf{pp} := (\mathsf{pp}_{\mathsf{Enc}}, \mathsf{pp}_\Pi, \mathcal{H})$
```

Suppose $\nu = 1$ and $\bar{\nu} = 2$. Algorithm 4 generates a public-private key pair along with a corresponding proof. It first runs mmKGen from the Ref-AH mmPKE scheme to obtain the public-private key pair, and then executes $\mathsf{LNP22.P}(R_{\text{pk}}, \mathsf{stat} = (\mathbf{A}, \mathbf{b}), \mathsf{wit} = \mathbf{s})$ to generate the proof π, where the proof relation R_{pk} is defined as follows:

$$R_{\text{pk}} = \left\{ (\mathbf{s} || \mathbf{b} - \mathbf{A}^\top \mathbf{s}) \in \{0,1\}^{(m+n)d} \right\}. \tag{3.1}$$

Later, the public key $\mathbf{b}$ together with the proof π can be verified in the Register algorithm via $0/1 \leftarrow \mathsf{LNP22.V}(R_{\text{pk}}, (\mathbf{A}, \mathbf{b}), \pi)$.

Algorithm 4 $\mathsf{AddrGen}()$

```
1: $(\mathsf{pk} := \mathbf{b},\ \mathsf{sk} := \mathbf{s}) \leftarrow \mathsf{mmKGen}(\mathsf{pp}_{\mathsf{Enc}})$
2: $\pi \leftarrow \mathsf{LNP22.P}(R_{\text{pk}}, (\mathbf{A}, \mathbf{b}), \mathbf{s})$ where $R_{\text{pk}}$ is defined in Equation (3.1)
3: return $(\mathsf{reg} := (\mathsf{pk}, \pi),\ \mathsf{sk})$
```

Algorithm 5 generates an anonymous transaction. In the anonymous transfer, the sender uses an anonymous set to hide the identity of herself and the recipient. Thus, each account in the anonymous should be treated as the sender and recipient. We begin by defining the proof relation R_{an} along with its statement and witness as follows,

$$R_{\text{an}} = \left\{ \begin{array}{l} \mathsf{stat} = (\mathbf{A}, \mathbf{A}_H, (\mathbf{b}_i, \mathbf{u}_i, v_i)_{i\in[N]}, (\mathbf{c}, (c_i)_{i\in[N]}), \mathbf{v}_H); \\ \mathsf{wit} = (\mathbf{s}_{\ell_\mathsf{s}}, \mathbf{r}, \mathbf{b}^{(\mathsf{s})}, \mathbf{b}^{(\mathsf{r})}, \hat{m}, \mathbf{b}_{m'}, bt_{m'}, \mathbf{e}_H, \mathbf{b}_h) : \\ \mathbf{b}^{(\mathsf{s})}, \mathbf{b}^{(\mathsf{r})} \in \{0,1\}^N \in \mathbb{Z}_q^N, \ \sum_{i\in[N]} b_i^{(\mathsf{s})} = 1, \ \sum_{i\in[N]} b_i^{(\mathsf{r})} = 1, \\ \mathsf{bin} \in \{0,1\}^{(m+n+2+n'+o+o')d}, \\ \left\| \begin{array}{c} \mathbf{r} \\ \mathbf{c} - \mathbf{A}\mathbf{r} \end{array} \right\|_\infty \le \psi \cdot \beta_0, \\ \left\| \begin{array}{c} c_1 - \langle \mathbf{b}_1, \mathbf{r} \rangle - \lfloor \frac{q}{2t+2} \rceil (b_1^{(\mathsf{r})} - b_1^{(\mathsf{s})})\hat{m} \\ \vdots \\ c_N - \langle \mathbf{b}_N, \mathbf{r} \rangle - \lfloor \frac{q}{2t+2} \rceil (b_N^{(\mathsf{r})} - b_N^{(\mathsf{s})})\hat{m} \\ (2 \cdot \mathbf{v}_H + \mathbf{e}_H - \mathbf{A}_H \cdot \mathbf{s}_{\ell_0})/\hat{q} \end{array} \right\|_\infty \le \psi \cdot \beta_1, \\ \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}' - \vec{t}_d \rangle = \langle \vec{0}^{d-k} || \vec{2}^k, \vec{bt}_{m'} \rangle, \\ \sum_{i\in[N]} b_i^{(\mathsf{s})} (v_i + c_i - \langle \mathbf{u}_i + \mathbf{c}, \mathbf{s}_{\ell_0} \rangle) - \lfloor q/(2t+2) \rceil \cdot \hat{m}' = h \end{array} \right\}. \tag{3.2}$$

where $h := \sum_{i\in[o]} \delta_i \cdot b_h^{(i)}$, $\hat{m}' := \sum_{i=[o']} \delta'_i \cdot b_{m'}^{(i)}$ are the same as in Equation (2.6) and

$$\mathsf{bin} := (\mathbf{s}_{\ell_\mathsf{s}} \ || \sum_{i\in[N]} b_i^{(\mathsf{s})} \mathbf{b}_i - \mathbf{A}^\top \mathbf{s}_{\ell_0} \ || \ \hat{m} \ || \ bt_{m'} \ || \ \mathbf{e}_H \ || \ \mathbf{b}_{m'} \ || \ \mathbf{b}_h).$$

The statement includes the public parameters $\mathbf{A}, \mathbf{A}_H$ for the current epoch H, the anonymous set $(\mathbf{b}_i)_{i\in[N]}$ with the corresponding accounts $(\mathbf{u}_i, v_i)_{i\in[N]}$, the set of amount ciphertexts $(\mathbf{c}, (c_i)_{i\in[N]})$, and the linkable tag $\mathbf{v}_H$.

The witness contains:

- the private key $\mathbf{s}_{\ell_\mathsf{s}}$ of the sender;
- the randomness $\mathbf{r}$ used in the amount ciphertexts $(\mathbf{c}, (c_i)_{i\in[N]})$;
- the binary strings $\mathbf{b}^{(\mathsf{s})}, \mathbf{b}^{(\mathsf{r})}$ indicating the indices of the sender and recipient, respectively, where only $b_{\ell_\mathsf{s}}^{(\mathsf{s})} = 1$, $b_{\ell_\mathsf{r}}^{(\mathsf{r})} = 1$, and all other elements are 0;
- a binary polynomial $\hat{m} \leftarrow \mathsf{BinPoly}(m)$ whose coefficients represent the binary decomposition of the integer amount m;
- a polynomial vector $\mathbf{b}_{m'} \leftarrow \mathsf{Bin}(\hat{m}')$ representing the coefficient-wise binary decomposition of the polynomial balance $\hat{m}' \in \mathcal{R}_{2t+2}$ in the sender's account *after* the transaction;
- a binary polynomial $bt_{m'} \leftarrow \mathsf{BinPoly}(m')$ whose coefficients are the binary decomposition of the integer balance $m' := \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}' \rangle$ in the sender's account *after* the transaction;
- the rounding error $\mathbf{e}_H$ related to the linkable tag $\mathbf{v}_H$;
- a polynomial vector $\mathbf{b}_h \leftarrow \mathsf{Bin}(h)$ which is the coefficient-wise binary decomposition of the decryption error h during the decryption of the sender's account after the transaction.

The proof relation consists of four parts, which are detailed as follows.

First, we must ensure that both $\mathbf{b}^{(\mathsf{s})}$ and $\mathbf{b}^{(\mathsf{r})}$ are binary strings over integers with Hamming weight equal to 1. We use $\mathbf{b}^{(\mathsf{s})}$ as an example: we first show that $\mathbf{b}^{(\mathsf{s})} \in \{0,1\}^N \subset \mathbb{Z}_q^N$, and then prove that $\sum_{i\in[N]} b_i^{(\mathsf{s})} = 1$.

Second, we prove that a concatenation of several vectors—specifically, $\mathbf{b}_{m'}$, $\mathbf{b}_h$, $\hat{m}$, $bt_{m'}$, $\mathbf{e}_H$, $\mathbf{s}_{\ell_\mathsf{s}}$, and $\mathbf{e}_{\ell_\mathsf{s}} := (\mathbf{b}_{\ell_\mathsf{s}} - \mathbf{A}^\top \mathbf{s}_{\ell_\mathsf{s}})$—forms a polynomial vector with binary coefficients. We choose parameters such that $\hat{q}/\hat{p} = 2$, $\nu = 1$, and $\bar{\nu} = 2$, which guarantees that the grounding error $\mathbf{e}_H \in \{0,1\}^{n'd}$ in the tag generation and that the private key $(\mathbf{s}, \mathbf{e}) \in \{0,1\}^{(m+n)d}$ are the binary polynomials. This setting is motivated by the fact that LNP22 can efficiently prove that polynomials are composed of binary coefficients.

Third, we combine the proofs of the well-formedness of the multi-recipient ciphertext in Equation (2.1) and the linkable tag in Equation (2.8) to improve efficiency. Here, we leverage the binary strings $\mathbf{b}^{(\mathsf{s})}$ and $\mathbf{b}^{(\mathsf{r})}$ to implicitly encode the messages in the ciphertexts as $m_i = (b_i^{(\mathsf{r})} - b_i^{(\mathsf{s})}) \cdot m$. Specifically, for decoy accounts, $b_i^{(\mathsf{s})} = b_i^{(\mathsf{r})} = 0$, so $m_i = 0$; for the sender, $b_{\ell_\mathsf{s}}^{(\mathsf{s})} = 1$, $b_{\ell_\mathsf{r}}^{(\mathsf{r})} = 0$, thus $m_{\ell_\mathsf{s}} = -m$; and for the recipient, $m_{\ell_\mathsf{r}} = m$. Our protocol also supports the special case where the sender and recipient are the same, i.e., $\ell_\mathsf{s} = \ell_\mathsf{r}$, in which case $m_{\ell_\mathsf{s}} = m_{\ell_\mathsf{r}} = 0$, which does not affect the balance property. Additionally, we slightly increase the bound to $\beta_1 := \sqrt{Nd \cdot (\tau\sigma_1)^2 + \beta^2}$ for $\beta = \sqrt{n'd} \cdot (dm\hat{q}\nu + \hat{q}/\hat{p} - 1 + \hat{q}/\hat{p} \cdot \hat{p})/\hat{q}$ to accommodate the combined proof.

Fourth, we use the verifiable decryption to show that the decrypted message of the sender's account after the transaction is non-negative. Since we shift the message space for balances from $\{-t, \ldots, t+1\}$ to $\{0, \ldots, 2t+1\}$ by adding the constant offset t_d, we need to shift it back by subtracting t_d during the range proof.

Algorithm 5 AnTransfer

```
Input: $(\mathsf{pk}_i = \mathbf{b}_i)_{i\in[N]}$, $\ell_\mathsf{s}$, $\ell_\mathsf{r}$, $\mathsf{sk}_{\ell_\mathsf{s}} = \mathbf{s}_{\ell_\mathsf{s}}$, $\mathsf{amt} = m$, $\mathbb{S}$
 1: $H \leftarrow \mathbb{S}$
 2: $\mathcal{H} \leftarrow \mathsf{pp}$
 3: $\mathbf{A}_H := \mathcal{H}(\text{Lether}||H)$
 4: $\mathbf{v}_H := \lfloor \mathbf{A}_H \cdot \mathbf{s}_{\ell_\mathsf{s}} \mod \hat{q} \rfloor_{\hat{p}}$
 5: $\mathbf{e}_H := \mathbf{A}_H \cdot \mathbf{s} - \hat{q}/p \cdot \mathbf{v}_H \mod \hat{q}$
 6: $(\mathbf{r}, \mathbf{e}_u) \leftarrow \mathcal{D}^n_{\sigma_0} \times \mathcal{D}^m_{\sigma_0}$
 7: $(y_i)_{i\in[N]} \leftarrow \mathcal{D}^N_{\sigma_1}$
 8: $\hat{m} \leftarrow \mathsf{BinPoly}(m)$
 9: $\hat{m}_{\ell_\mathsf{s}} := -\hat{m}$, $\hat{m}_{\ell_\mathsf{r}} := \hat{m}$, $\hat{m}_i := 0$ for $i \in [N]/\{\ell_\mathsf{s}, \ell_\mathsf{r}\}$
10: $(\mathbf{c}, (c_i)_{i\in[N]}) \leftarrow \mathsf{mmEnc}(\mathbf{A}, (\mathbf{b}_i)_{i\in[N]}, (\hat{m}_i)_{i\in[N]}; (\mathbf{r}, \mathbf{e}_u), (y_i)_{i\in[N]})$
11: $\mathbf{b}^{(\mathsf{s})} := (b^{(\mathsf{s})}_1, ..., b^{(\mathsf{s})}_N) \in \{0,1\}^N$, where $b^{(\mathsf{s})}_{\ell_\mathsf{s}} := 1$, $b^{(\mathsf{s})}_i := 0$ for $i \in [N]/\{\ell_\mathsf{s}\}$
12: $\mathbf{b}^{(\mathsf{r})} := (b^{(\mathsf{r})}_1, ..., b^{(\mathsf{r})}_N) \in \{0,1\}^N$, where $b^{(\mathsf{r})}_{\ell_\mathsf{r}} := 1$, $b^{(\mathsf{r})}_i := 0$ for $i \in [N]/\{\ell_\mathsf{r}\}$
13: $\mathsf{acc} \leftarrow \mathbb{S}$
14: for $i \in [N]$, $(\mathbf{u}_i, v_i) \leftarrow \mathsf{acc}[\mathsf{pk}_i]$
15: $(\mathbf{u}'_{\ell_\mathsf{s}}, v'_{\ell_\mathsf{s}}) := (\mathbf{u}_{\ell_\mathsf{s}} + \mathbf{c}, v_{\ell_\mathsf{s}} + c_{\ell_\mathsf{s}})$
16: $\hat{m}' := \lfloor v'_{\ell_\mathsf{s}} - \langle \mathbf{u}'_{\ell_\mathsf{s}}, \mathbf{s}_{\ell_\mathsf{s}} \rangle \rceil_{2t+2}$
17: $h := v'_{\ell_\mathsf{s}} - \langle \mathbf{u}'_{\ell_\mathsf{s}}, \mathbf{s}_{\ell_\mathsf{s}} \rangle - \lfloor q/(2t+2) \rceil \cdot \hat{m}'$
18: $\mathbf{b}_{m'} \leftarrow \mathsf{Bit}(\hat{m}')$
19: $\mathbf{b}_h \leftarrow \mathsf{Bit}(h)$
20: $m' := \langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}' - \vec{t}_d \rangle$
21: $bt_{m'} \leftarrow \mathsf{BinPoly}(m')$
22: $\mathit{\Pi} \leftarrow \mathsf{LNP22.P}(R_{\text{an}}, \mathsf{stat}, \mathsf{wit})$ where $R_{\text{an}}$ is defined in Equation (3.2)
23: return $\mathsf{tx} = ((\mathsf{pk}_i := \mathbf{b}_i)_{i\in[N]}, \mathbf{ct} := (\mathbf{c}, (c_i)_{i\in[N]}), \mathsf{tag} := \mathbf{v}_H, \mathit{\Pi})$
```

Algorithm 6 verifies the transaction generated by Algorithm 5. It first constructs the corresponding statement and then validates the proof included in the transaction using the LNP22.V algorithm.

Algorithm 6 Verify(tx, $\mathbb{S}$)

```
1: Phrase $((\mathsf{pk}_i = \mathbf{b}_i)_{i\in[N]}, \mathbf{ct} = (\mathbf{c}, (c_i)_{i\in[N]}), \mathsf{tag} = \mathbf{v}_H, \mathit{\Pi}) \leftarrow \mathsf{tx}$
2: $H \leftarrow \mathbb{S}$
3: $\mathcal{H} \leftarrow \mathsf{pp}$
4: $\mathbf{A}_H := \mathcal{H}(\text{Lether}||H)$
5: $\mathsf{acc} \leftarrow \mathbb{S}$
6: for $i \in [N]$, $(\mathbf{u}_i, v_i) \leftarrow \mathsf{acc}[\mathsf{pk}_i]$
7: return $0/1 \leftarrow \mathsf{LNP22.V}(R_{\text{an}}, \mathsf{stat}, \mathit{\Pi})$ where $R_{\text{an}}$ is defined in Equ. (3.2)
```

The correctness of Lether follows from the completeness of LNP22 and the correctness of our Ref-AH mmPKE scheme. Under our parameter selection, the probability that decryption succeeds is at least $1 - 2^{-128}$, implying that Lether achieves statistical correctness.

The security proof is deferred to Appendix C due to space constraints.

3.1. Parameter Setting and Implementation

We first present the parameter settings for Lether. At a high level, we must ensure that the parameters satisfy not only the requirements of the verifiable Ref-AH mmPKE and the tag scheme (as discussed in Section 2.1 and Section 2.2), but also the conditions required by LNP22 when combining these proofs. We summarize the LNP22 requirements as follows:

- **Switched modulus condition:** As discussed in Section 1.2, to prove the well-formedness of the linkable tag via LNP22, we must ensure that the equation
$$\hat{q}/\hat{p} \cdot \mathbf{v}_H - \mathbf{A}_H \cdot \mathbf{s}_{\ell_0} - \mathbf{e}_H + \hat{q} \cdot \mathbf{v} = 0$$
does not wrap around modulo q, where $\|\mathbf{v}\|_\infty \leq \beta_1$ as defined in R_{an}. This leads to the requirement that
$$q \geq (\hat{q}/\hat{p} \cdot \hat{p} + md \cdot \hat{q}\nu + \hat{q}/\hat{p} - 1) + \hat{q} \cdot \beta_1.$$
- **Binary proof condition:** To prove that a polynomial vector $\mathbf{b} \in \{0,1\}^{m'd}$ has binary coefficients, LNP22 reduces this to checking $\langle \vec{b}, \vec{b} - \vec{1}^{m'd} \rangle = 0$ over $\mathbb{Z}_q$. To avoid wraparound modulo q, suppose $\|\mathbf{b}\|_\infty \leq B$, we must ensure $q \geq dm' \cdot B^2 + dm' \cdot B$. In our setting, $B \leq \psi \cdot \sqrt{m'd}$ and $m' \leq m + n + 2 + n' + o + o'$.
- **ARP condition:** Following [28, Lemma 2.9], to ensure the soundness of ARP proofs in R_{an}, we must satisfy
$$q \geq 41 \cdot (N + n') \cdot d \cdot \psi \cdot \beta_1, \quad \text{and} \quad q \geq 41 \cdot (m + n) \cdot d \cdot \psi \cdot \beta_0.$$
- **Inner product condition:** To ensure that the inner product in R_{an} does not wrap around modulo q, we require $q \geq 2^k \cdot (2t+1)$.

To satisfy the above conditions, we set $k := 32$ and choose the following parameters for Lether. For the mmPKE: $q \approx 2^{46}$, $N = 16$, $m = 30$, $n = 26$, $t = 62$, $\nu = 1$, $\bar{\nu} = 2$, $\sigma_0 = 15.9$, and $\sigma_1 = 30560$. For the tag scheme: $\hat{q} \approx 2^{11}$, $\hat{p} \approx 2^{10}$, with $\hat{q}/\hat{p} = 2$, and $n' = 2$. Note that the modulus q is also shared in LNP22 to maintain compatibility across components.

Denial of Service. We analyze the probability of a denial-of-service attack, i.e., the likelihood that an adversary, given a linkable tag $\mathbf{v}_H$, can generate a valid transaction with the same tag without knowing the associated private key. Such an attack can be reduced to the non-frameability property of the tag scheme.

In the end, we implement Lether in C[7] on a standard desktop machine equipped with an Intel i7-11850H CPU running at 2.50GHz.

Our implementation is based on the LaZer library [39].[8] Although the LaZer library provides a simple-to-use Python interface, it only supports proving simple linear relations, and the moduli used in the NIZK system and the target relations are inconsistent. Thus, to support the relations required in Lether, we extensively restructure the internal functions of LaZer, as well as the parameter generation scripts. The newly added or modified code exceeds 3,000 lines. Results for Lether are presented in Table 1, where each runtime value is averaged over 100 runs.

7. Our implementation: https://github.com/LetherSub/Lether-artifact
8. https://github.com/lazer-crypto/lazer

References

[1] M. H. Devoret and R. J. Schoelkopf, "Superconducting circuits for quantum information: an outlook," *Science*, vol. 339, no. 6124, pp. 1169–1174, 2013.

[2] E. Foundation. (2025) Introducing zknox. Posted on X (formerly Twitter), March 26, 2025. [Online]. Available: https://x.com/ethereumfndn/status/1896592240228893072

[3] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 459–474.

[4] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[5] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22*. Springer, 2017, pp. 456–474.

[6] T. H. Yuen, S.-f. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 2020, pp. 464–483.

[7] R. W. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling private payments without trusted setup," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 31–48.

[8] T. Zheng, S. Gao, Y. Song, and B. Xiao, "Leaking arbitrarily many secrets: Any-out-of-many proofs and applications to ringct protocols," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2533–2550.

[9] N. Wang, Q. Wang, D. Liu, M. F. Esgin, and A. Abuadbba, "Bulletct: Towards more scalable ring confidential transactions with transparent setup," *Cryptology ePrint Archive*, 2025.

[10] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu, "MatRiCT: efficient, scalable and post-quantum blockchain confidential transactions protocol," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 567–584.

[11] M. F. Esgin, R. Steinfeld, and R. K. Zhao, "MatRiCT+: More efficient post-quantum private blockchain payments," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1281–1298.

[12] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "Quisquis: A new design for anonymous cryptocurrencies," in *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25*. Springer, 2019, pp. 649–678.

[13] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 423–443.

[14] B. E. Diamond, "Many-out-of-many proofs and applications to anonymous zether," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1800–1817.

[15] Y. Guo, H. Karthikeyan, A. Polychroniadou, and C. Huussin, "Pride ct: Towards public consensus, private transactions, and forward secrecy in decentralized payments," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3904–3922.

[16] A. Sarencheh, H. Khoshakhlagh, A. Kavousi, and A. Kiayias, "Dart: Decentralized, anonymous, and regulation-friendly tokenization," *Cryptology ePrint Archive*, 2025.

[17] V. Madathil and A. Scafuro, "Prifhete: Achieving full-privacy in account-based cryptocurrencies is possible," *Cryptology ePrint Archive*, 2023.

[18] K. Kurosawa, "Multi-recipient public-key encryption with shortened ciphertext," in *Public Key Cryptography*, D. Naccache and P. Paillier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 48–63.

[19] M. Bellare, A. Boldyreva, and J. Staddon, "Randomness re-use in multi-recipient encryption schemeas," in *Public Key Cryptography — PKC 2003*, Y. G. Desmedt, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 85–99.

[20] M. H. Au, S. S. Chow, W. Susilo, and P. P. Tsang, "Short linkable ring signatures revisited," in *Public Key Infrastructure: Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006. Proceedings 3*. Springer, 2006, pp. 101–115.

[21] M. Backes, N. Döttling, L. Hanzlik, K. Kluczniak, and J. Schneider, "Ring signatures: logarithmic-size, no setup—from standard assumptions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 281–311.

[22] J. Bootle, K. Elkhiyaoui, J. Hesse, and Y. Manevich, "Dualdory: logarithmic-verifier linkable ring signatures through preprocessing," in *European Symposium on Research in Computer Security*. Springer, 2022, pp. 427–446.

[23] A. Haque, S. Krenn, D. Slamanig, and C. Striecks, "Logarithmic-size (linkable) threshold ring signatures in the plain model," in *IACR International Conference on Public-Key Cryptography*. Springer, 2022, pp. 437–467.

[24] H. Wang, R. Steinfeld, M.-J. O. Saarinen, M. F. Esgin, and S.-M. Yiu, "Post-quantum multi-message public key encryption from extended reproducible PKE," Cryptology ePrint Archive, Paper 2025/1000, 2025. [Online]. Available: https://eprint.iacr.org/2025/1000

[25] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[26] Y. Tang, D. Pan, Q. Ye, Y. Li, and J. Yu, "Event-oriented linkable group signature from lattice," *IEEE Transactions on Consumer Electronics*, 2024.

[27] M. F. Esgin, R. Steinfeld, and R. K. Zhao, "Efficient verifiable partially-decryptable commitments from lattices and applications," in *IACR International Conference on Public-Key Cryptography*. Springer, 2022, pp. 317–348.

[28] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, "Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general," in *Annual International Cryptology Conference*. Springer, 2022, pp. 71–101.

[29] V. Lyubashevsky and G. Neven, "One-shot verifiable encryption from lattices," in *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I 36*. Springer, 2017, pp. 293–323.

[30] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Shorter lattice-based zero-knowledge proofs via one-time commitments," in *IACR International Conference on Public-Key Cryptography*. Springer, 2021, pp. 215–241.

[31] K. Gjøsteen, T. Haines, J. Müller, P. Rønne, and T. Silde, "Verifiable decryption in the head," in *Australasian Conference on Information Security and Privacy*. Springer, 2022, pp. 355–374.

[32] D. F. Aranha, C. Baum, K. Gjøsteen, and T. Silde, "Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1467–1481.

[33] T. Silde, "Short paper: Verifiable decryption for bgv," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 381–390.

[34] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu, "Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications," in *Annual International Cryptology Conference*. Springer, 2019, pp. 115–146.

[35] T. H. Yuen, M. F. Esgin, J. K. Liu, M. H. Au, and Z. Ding, "Dualring: generic construction of ring signatures with efficient instantiations," in *Annual International Cryptology Conference*. Springer, 2021, pp. 251–281.

[36] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Smile: set membership from ideal lattices with applications to ring signatures and confidential transactions," in *Annual International Cryptology Conference*. Springer, 2021, pp. 611–640.

[37] V. Lyubashevsky, N. K. Nguyen, M. Plançon, and G. Seiler, "Shorter lattice-based group signatures via "almost free" encryption and other optimizations," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2021, pp. 218–248.

[38] M. F. Esgin, R. Steinfeld, D. Liu, and S. Ruj, "Efficient hybrid exact/relaxed lattice proofs and applications to rounding and vrfs," in *Annual International Cryptology Conference*. Springer, 2023, pp. 484–517.

[39] V. Lyubashevsky, G. Seiler, and P. Steuer, "The lazer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 3125–3137.

[40] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 738–755.

[41] R. del Pino and S. Katsumata, "A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling," in *Advances in Cryptology – CRYPTO 2022*, Y. Dodis and T. Shrimpton, Eds. Cham: Springer Nature Switzerland, 2022, pp. 306–336.

[42] M. Chase and A. Lysyanskaya, "On signatures of knowledge," in *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*. Springer, 2006, pp. 78–96.

[43] J. Groth, R. Ostrovsky, and A. Sahai, "Perfect non-interactive zero knowledge for np," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 339–358.

Appendix A. Preliminaries

A.1. Notation

Let $\lambda \in \mathbb{N}$ denote the security parameter. For a positive integer n, we denote the set $\{1, ..., n\}$ by $[n]$. For a positive b, t, we denote the integer vector $(b^0, b^1, ..., b^{t-1})$ by $\vec{b}^t$. For a positive integer q, we denote $\mathbb{Z}_q$ as the integers modulo q and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ as the polynomials modulo q and $X^d + 1$. For a positive integer b, we denote a polynomial with all coefficients equal to b as $b_d \in \mathcal{R}_q$. For positive integer ν, we write $\mathbb{S}_\nu$ to denote the set of polynomials in $\mathcal{R}_q$ with infinity norm bounded by ν. The size of the $\mathbb{S}_\nu$ coefficient support is denoted $\bar{\nu} \leq 2\nu + 1$; for example $\nu = 1, \bar{\nu} = 2$ indicates binary polynomials. We denote bold lowercase letters as vectors of polynomial elements, e.g., $\mathbf{u} \in \mathcal{R}_q^m$, bold uppercase letters as matrices of polynomial elements, e.g., $\mathbf{U} \in \mathcal{R}_q^{m\times n}$, and lowercase letters with an arrow as vectors of integers or reals, e.g., $\vec{a} \in \mathbb{Z}_q^m$. For a polynomial vector (or element when $m = 1$), e.g., $\mathbf{a} \in \mathcal{R}_q^m$, we define $\vec{a} \in \mathbb{Z}_q^{md}$ as the integer vector concatenated by the coefficients of each polynomial in $\mathbf{a}$. For the vectors over integers and polynomials, we denote their inner product as $\langle \cdot, \cdot \rangle$, e.g., $\langle \vec{a}, \vec{b} \rangle$ and $\langle \mathbf{a}, \mathbf{b} \rangle$.

We denote rounding operation as $\lfloor \cdot \rceil$, e.g., $\lfloor a \rceil$ rounds the result to the nearest integer of a. We denote assignment as $:=$, e.g., $x := y$ assigns the value of y to x. We denote sampling or output as $\leftarrow$, e.g., $x \leftarrow \mathcal{D}$ indicates that x is sampled from the distribution $\mathcal{D}$, and $x \leftarrow \mathsf{A}(y)$ denotes that x is the output of probabilistic polynomial time (PPT) algorithm A given input y. Particularly, we write $x \leftarrow S$ when $x \in S$ is sampled uniformly randomly from the finite set S. We denote the uniform distribution on a set S as $\mathcal{U}(S)$. For a vector $\mathbf{a}$ (or $\vec{a}$), we write $\|\mathbf{a}\|$, $\|\mathbf{a}\|_1$, and $\|\mathbf{a}\|_\infty$ to denote its ℓ_2-norm, ℓ_1-norm and ℓ_∞-norm, respectively.

In the end, we define some useful functions as follows:

- $\hat{m} \in \mathcal{R}_2 \leftarrow \mathsf{BinPoly}(m \in \mathbb{Z}_q)$: Given an integer $m \in \mathbb{Z}_q$ with $\|m\|_\infty < 2^d$, this function outputs a binary polynomial $\hat{m} \in \mathcal{R}_2$ whose coefficients represent the binary decomposition of m.
- $\mathbf{b}_h \in \mathcal{R}_2^m \leftarrow \mathsf{Bit}(h \in \mathcal{R}_q)$: Given a polynomial $h \in \mathcal{R}_q$ with $\|h\|_\infty \leq \beta$ and $m = \lceil \log(\beta + 1) \rceil$, this function outputs a binary polynomial vector $\mathbf{b}_h \in \mathcal{R}_2^m$ representing the coefficient-wise binary decomposition of h, i.e., $h = \sum_{i\in[m]} \delta_i \cdot b_h^{(i)}$, where $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_m) = (1, 2, \ldots, \beta + 1 - 2^{\lfloor \log \beta \rfloor})$.

A.2. Non-Interactive Zero Knowledge Protocol

We define non-interactive zero knowledge (NIZK) protocol as follows.

Definition A.1 (Non-Interactive Zero Knowledge Protocol)**.** Let $R_\mathcal{L}$ be a polynomial-time verifiable relation of statement-witness (x, w). Denote a language $\mathcal{L}$ as a set of statements where there exists a witness w with $(x, w) \in R_\mathcal{L}$. A NIZK protocol Π is defined as follows.

- $\mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$: Input a security parameter 1^λ, it outputs a public parameter pp_Π.
- $\pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pp}_\Pi, x, w)$: Input the public parameter pp_Π, a statement x and a witness w such that $(x, w) \in R_\mathcal{L}$, it outputs a proof π.
- $0/1 \leftarrow \Pi.\mathsf{Verify}(\mathsf{pp}_\Pi, x, \pi)$: Input the public parameter pp_Π, a statement x and a proof π, it output 1 if accepts, otherwise, it outputs 0.

We then define the properties of computational completeness, computational zero knowledge, and computational knowledge soundness for NIZK argument system following [38], [43].

Computational Completeness. A NIZK argument system Π is computational completeness if for any $(x, w) \in R_\mathcal{L}$, the following probability holds overwhelming,

$$\Pr\left[\Pi.\mathsf{Verify}(\mathsf{pp}_\Pi, x, \pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda); \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pp}_\Pi, x, w) \end{array}\right].$$

Computational Zero Knowledge. A NIZK argument system Π is computational zero knowledge if for any PPT adversary

$\mathcal{A}$, there exists a PPT simulator $\Pi.\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that the following is negligible with λ,

$$\left| \Pr\left[\begin{array}{c} (x,w) \in R_{\mathcal{L}}; \\ \mathcal{A}(\mathsf{pp}_\Pi, \pi) = 1 \end{array} \middle| \begin{array}{c} \mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda); \\ (x,w) \leftarrow \mathcal{A}(\mathsf{pp}_\Pi); \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pp}_\Pi, x, w) \end{array} \right] - \Pr\left[\begin{array}{c} (x,w) \in R_{\mathcal{L}}; \\ \mathcal{A}(\mathsf{pp}_\Pi, \pi) = 1 \end{array} \middle| \begin{array}{c} (\mathsf{pp}_\Pi, \tau) \leftarrow \mathsf{Sim}_0(1^\lambda); \\ (x,w) \leftarrow \mathcal{A}(\mathsf{pp}_\Pi); \\ \pi \leftarrow \mathsf{Sim}_1(\mathsf{pp}_\Pi, x, \tau) \end{array} \right] \right|.$$

Computational Knowledge Soundness. A NIZK argument system Π is computational knowledge soundness if for any PPT adversary $\mathcal{A}$, there exists an expected PPT extractor $\Pi.\mathcal{E}$ having full access to the adversary's state, such that the following probability is negligible with λ,

$$\Pr\left[\begin{array}{c} \Pi.\mathsf{Verify}(\mathsf{pp}_\Pi, x, \pi) = 1 \\ \wedge (x, \bar{w}) \notin R_{\mathcal{L}} \end{array} \middle| \begin{array}{c} \mathsf{pp}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda); \\ (x,\pi) \leftarrow \mathcal{A}(\mathsf{pp}_\Pi); \\ \bar{w} \leftarrow \Pi.\mathcal{E}(\mathsf{pp}_\Pi, \mathcal{A}, x, \pi) \end{array} \right].$$

A.3. NIZK Protocol in [28]

We recall one of the most efficient lattice-based NIZK protocols proposed in [28], denoted as $\mathsf{LNP22}$. We treat it as a black-box throughout this paper and do not delve into its technical details.

We first define the proof relation $R_{\mathcal{L}}$ in $\mathsf{LNP22}$ as follows:

$$R_{\mathcal{L}} = \left\{ \begin{array}{l} \mathsf{wit} = \mathbf{s},\ \mathsf{stat} = (\phi, \Psi, \Theta, \Omega): \\ \forall f \in \phi,\ f(\mathbf{s}) = 0 \text{ over } \mathcal{R}_q, \\ \forall F \in \Psi,\ F(\mathbf{s}) = 0 \text{ over } \mathbb{Z}_q, \\ \forall (\mathbf{E}, \mathbf{v}, \beta) \in \Theta,\ \|\mathbf{E}\mathbf{s} - \mathbf{v}\|_2 \le \beta, \\ \forall (\mathbf{D}, \mathbf{u}, \beta) \in \Omega,\ \|\mathbf{D}\mathbf{s} - \mathbf{u}\|_2 \le \psi \cdot \beta \end{array} \right\}. \quad \text{(A.1)}$$

For clarity, we omit the public parameters of $\mathsf{LNP22}$ and their relation to the witness in $R_{\mathcal{L}}$. The witness in $R_{\mathcal{L}}$ is a vector $\mathbf{s}$ over $\mathcal{R}_q$. The statement consists of:

- a set ϕ of linear and quadratic functions over $\mathcal{R}_q$,
- a set Ψ of linear and quadratic functions over $\mathbb{Z}_q$,
- a set Θ of exact ℓ_2 bounds on linear functions,
- a set Ω of approximate ℓ_2 bounds with relaxation factor $\psi \approx 189$.

Notably, $\mathsf{LNP22}$ can also support:

- Proving a bit $b \in \{0,1\}$ by first proving $b \in \mathbb{Z}$ via the relation $\langle \vec{b}, \vec{\delta}^{(i)} \rangle = 0$ for all $i \in \{1, \ldots, d-1\}$ where $\delta^{(i)} := X^i \in \mathcal{R}_q$, then proving $b(b-1) = 0$ over $\mathcal{R}_q$;
- Proving that a polynomial $a \in \mathcal{R}_q$ has binary coefficients by showing $\langle \vec{a}, \vec{a} - \vec{1}^d \rangle = 0$ over $\mathbb{Z}$;
- Proving an approximate ℓ_∞ norm bound via the corresponding ℓ_2 norm, i.e., $\|\mathbf{D}\mathbf{s} - \mathbf{u}\|_\infty \le \|\mathbf{D}\mathbf{s} - \mathbf{u}\|_2 \le \psi \cdot \beta$.

Construction A.2 ($\mathsf{LNP22}$ from [28])**.** Let λ be the security parameter. The $\mathsf{LNP22}$ protocol consists of the following algorithms:

- $\mathsf{pp}_{\mathsf{LNP}} \leftarrow \mathsf{LNP22.Setup}(1^\lambda)$: Given the security parameter λ, output the public parameters $\mathsf{pp}_{\mathsf{LNP}}$.
- $\pi \leftarrow \mathsf{LNP22.P}(\mathsf{pp}_{\mathsf{LNP}}, R_{\mathcal{L}}, \mathsf{stat}, \mathsf{wit})$: Given $\mathsf{pp}_{\mathsf{LNP}}$, a proof relation $R_{\mathcal{L}}$, statement stat, and witness wit as defined in Equation (A.1), output a proof π.
- $0/1 \leftarrow \mathsf{LNP22.V}(\mathsf{pp}_{\mathsf{LNP}}, R_{\mathcal{L}}, \mathsf{stat}, \pi)$: Given $\mathsf{pp}_{\mathsf{LNP}}$, the relation $R_{\mathcal{L}}$ and statement stat, and a proof π, output 1 if the proof is valid, otherwise output 0.

The completeness, knowledge soundness, and simulatability of $\mathsf{LNP22}$ are analyzed in detail in [28]; we refer the reader to the original paper for further information.

A.4. Lattice Preliminaries

We show the definition of the standard lattice-based hard problems.

Definition A.3 (MLWE Problem)**.** Let $m, n > 0$ be positive integers. Let χ be an error distribution over $\mathcal{R}$. The MLWE problem, denoted by $\mathsf{MLWE}_{\mathcal{R},m,n,q,\chi}$, asks an adversary $\mathcal{A}$ to distinguish the following two case: (1) $(\mathbf{A}, \mathbf{A}\mathbf{r} + \mathbf{e})$ for $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $\mathbf{r} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^m$, and (2) $(\mathbf{A}, \mathbf{u})$ for $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $\mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$.

Definition A.4 (MLWR Problem)**.** Let $m, n, p, q > 0$ be positive integers. Let χ be an error distribution over $\mathcal{R}$. The MLWR problem, denoted by $\mathsf{MLWR}_{\mathcal{R},m,n,q,p,\chi}$, asks an adversary $\mathcal{A}$ to distinguish the following two case: (1) $(\mathbf{A}, \lfloor \mathbf{A}\mathbf{r} \rceil_p)$ for $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $\mathbf{r} \leftarrow \chi^n$, and (2) $(\mathbf{A}, \lfloor \mathbf{u} \rceil_p)$ for $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $\mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$. Note that if p divides q, $\lfloor \mathbf{u} \rceil_p$ is itself uniform over $\mathcal{R}_p^m$.

Definition A.5 (MSIS Problem)**.** Let $m, n, \beta > 0$ be positive integers with $n > m$, given $\mathbf{A} := [\mathbf{I}_m || \mathbf{A}'] \in \mathcal{R}_q^{m \times n}$ with $\mathbf{A}' \in \mathcal{R}_q^{m \times (n-m)}$, the MSIS problem, denoted by $\mathsf{MSIS}_{\mathcal{R},m,n,q,\beta}$, asks an adversary $\mathcal{A}$ to find a short non-zero vector $\mathbf{v} \in \mathcal{R}^n$ such that $\mathbf{A}\mathbf{v} = \mathbf{0} \in \mathcal{R}_q^m$ and $\|\mathbf{v}\|_\infty \le \beta$.

Appendix B. Formal Definitions for Account-Based Private Blockchain Payments

In this section, we present a new formal definition for account-based private blockchain payment systems that is adapted from those proposed in [13], [14]. We also discuss the key differences from RingCT protocols in UTXO-based blockchains, such as [5]–[11].

The main advantage of our definition is that it explicitly captures the inherent stateful nature of blockchain environments and incorporates the notion of epochs, which is omitted in [14]. Similar to [14], we focus primarily on transfer security; for the security of funding and withdrawal (burn), we refer the reader to [13].

We begin with the formal definition.

Definition B.1 (Account-Based Private Blockchain Payment)**.** We first introduce the notation used specifically for the security model in Table 6. The blockchain state $\mathbb{S}$ consists of: (1) a table $\mathsf{acc}[\cdot]$ of registered accounts, where each public key pk indexes an account as $\mathsf{acc}[\mathsf{pk}]$; (2) a table of all verified transactions; and (3) the current epoch H, computed from the block height h and a system constant E as $\lfloor h/E \rfloor$. The epoch serves as a logical time slot in the blockchain.

The algorithms of the account-based private blockchain payment protocol are defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: Given a security parameter λ, output the system public parameters pp. We assume pp is an implicit input to all remaining algorithms.
- $(\mathsf{pk}, \mathsf{sk}, \pi) \leftarrow \mathsf{AddrGen}()$: Generate a public-private key pair $(\mathsf{pk}, \mathsf{sk})$ and a proof π of correct key generation. The public key pk serves as the account address.
- $\mathbb{S} \leftarrow \mathsf{Register}(\mathsf{pk}, \pi, \mathbb{S})$: On input a public key pk and a proof π, update the state by initializing $\mathsf{acc}[\mathsf{pk}]$ if the proof is valid.
- $\mathsf{tag}_H \leftarrow \mathsf{TagGen}(\mathsf{sk}, H)$: On input a private key sk and the current epoch H, output a linkable tag tag_H.
- $\mathbf{ct} \leftarrow \mathsf{AmtGen}((\mathsf{amt}_i)_{i\in[N]}, (\mathsf{pk}_i)_{i\in[N]})$: On input a set of amounts $(\mathsf{amt}_i)_{i\in[N]}$ and corresponding public keys $(\mathsf{pk}_i)_{i\in[N]}$, output a set of ciphertexts $\mathbf{ct}$.
- $\mathbb{S} \leftarrow \mathsf{RollOver}((\mathsf{pk}_i)_{i\in[N]}, \mathbf{ct}, \mathbb{S})$: Given a set of addresses $(\mathsf{pk}_i)_{i\in[N]}$, a set of amount ciphertexts $\mathbf{ct}$, and the blockchain state $\mathbb{S}$, update the state by setting $\mathsf{acc}[\mathsf{pk}_i] = \mathsf{ct}_i \oplus \mathsf{acc}[\mathsf{pk}_i]$ where $\oplus$ denotes the additively-homomorphic evaluation.
- $\mathsf{tx} \leftarrow \mathsf{AnTransfer}((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}, \ell_\mathsf{r}, \mathsf{sk}_{\ell_\mathsf{s}}, \mathsf{amt}, \mathbb{S})$: On input an anonymity set of public keys $(\mathsf{pk}_i)_{i\in[N]}$, sender index ℓ_s, recipient index ℓ_r, the sender's private key $\mathsf{sk}_{\ell_\mathsf{s}}$, the transfer amount amt, and the current state $\mathbb{S}$, generate amount ciphertexts $\mathbf{ct} \leftarrow \mathsf{AmtGen}((\mathsf{amt}_i)_{i\in[N]}, (\mathsf{pk}_i)_{i\in[N]})$ such that $\mathsf{amt}_{\ell_\mathsf{s}} = -\mathsf{amt}$, $\mathsf{amt}_{\ell_\mathsf{r}} = \mathsf{amt}$, and $\mathsf{amt}_i = 0$ for all $i \notin \{\ell_\mathsf{s}, \ell_\mathsf{r}\}$, a tag $\mathsf{tag}_H \leftarrow \mathsf{TagGen}(\mathsf{sk}_{\ell_\mathsf{s}}, H)$ and a proof Π. Output the transaction $\mathsf{tx} = ((\mathsf{pk}_i)_{i\in[N]}, \mathbf{ct}, \mathsf{tag}_H, \Pi)$.
- $0/1 \leftarrow \mathsf{LinkTag}(\mathsf{tag}, \mathbb{S})$: On input a tag tag and the blockchain state $\mathbb{S}$, output 1 if the tag already appears in $\mathbb{S}$, and 0 otherwise.
- $0/\mathbb{S} \leftarrow \mathsf{Verify}(\mathsf{tx}, \mathbb{S})$: On input a transaction tx and the blockchain state $\mathbb{S}$, if the proof Π is valid, update the state by recording tx; otherwise, output 0.
- $\mathsf{m} \leftarrow \mathsf{ReadBalance}((\mathsf{pk}, \mathsf{sk}), \mathbb{S})$: On input a public-private key pair $(\mathsf{pk}, \mathsf{sk})$ and the blockchain state $\mathbb{S}$, output the balance m of the account $\mathsf{acc}[\mathsf{pk}]$.

Table 6: Notations for the account-based private blockchain payment formal model.

Symbol	Description
$\mathbb{S}$	the blockchain state
$\mathsf{acc}[\mathsf{pk}]$	an account indexed by the address pk
N	# of public keys in the anonymous set
$(\mathsf{pk}_i)_{i\in[N]}$	anonymous set of public keys with $\lvert(\mathsf{pk}_i)_{i\in[N]}\rvert = N$
$\mathbf{ct}$	set of ciphertext with $\lvert\mathbf{ct}\rvert = N$
$\ell_\mathsf{s}, \ell_\mathsf{r}$	the indices to indicate the sender and recipient
amt	the transaction amount
π, Π	the proofs for the public key and the transaction
tag_H	the linkable tag of the transaction at epoch H
tx	a transaction $\mathsf{tx} = ((\mathsf{pk}_i)_{i\in[N]}, \mathbf{ct}, \mathsf{tag}_H, \Pi)$
V	a set of all valid balances/amounts, $\mathsf{V} \subseteq [0, ..., \mathsf{MAX}]$

Like [10], we use the blockchain state $\mathbb{S}$ to capture the nature of a blockchain environment and clarify how an account-based private blockchain payment system operates. Similar to [14], we assume for simplicity that each transaction takes effect immediately. Specifically, after receiving a transaction tx generated by a user, a trusted party[9] runs $\mathsf{RollOver}((\mathsf{pk}_i)_{i\in[N]}, \mathbf{ct}, \mathbb{S})$ if $\mathsf{Verify}(\mathsf{tx}, \mathbb{S}) \neq 0$ and $\mathsf{LinkTag}(\mathsf{tag}, \mathbb{S}) = 0$.

We highlight the main differences between account-based private blockchain payments and UTXO-based RingCT protocols as follows:

- All public-private key pairs generated by the AddrGen algorithm and used in the AnTransfer algorithm are long-term keys (addresses), which must be verified for well-formedness due to the requirements of mmPKE. In contrast, UTXO-based RingCT uses one-time keys (stealth addresses).
- The linkable tag $\mathsf{tag}_H \leftarrow \mathsf{TagGen}(\mathsf{sk}, H)$ is derived from the long-term private key and the current epoch in the blockchain state. In UTXO-based RingCT, the tag (or serial number) typically depends only on a one-time private key.
- The input to the AmtGen algorithm can be negative, allowing a non-negative transaction amount to be "subtracted" from the sender's balance when the ciphertext is added to the account. In contrast, the input to the Mint algorithm in UTXO-based RingCT must be non-negative, as it represents the value of a coin.
- Once a transaction tx is approved, the set of ciphertexts $\mathbf{ct}$ is added to all involved accounts $(\mathsf{pk}_i)_{i\in[N]}$. In UTXO-based RingCT, the output coins (with one-time public keys) are validated and stored on the blockchain and can later be spent by the recipients.

It can be observed that the communication cost in account-based private blockchain payments is linear in the size of the anonymity set. As stated in (Anonymous) Zether [13], [14], this is difficult to avoid in their paradigm due to the nature of account-based blockchains, where each account in the anonymity set must be treated as both a potential sender and recipient, and its balance updated with the corresponding amount ciphertext if the transaction is valid. In contrast, UTXO-based RingCT typically achieves communication cost logarithmic in the size of the anonymity set and linear in the number of input/output coins, except for the *any-out-of-many proof* in [8], which requires publishing auxiliary data for all coins in the anonymity set to hide the number of input coins.

B.1. Security Definition

We define the list $\mathcal{U}$ in Table 7. Following [10], the list $\mathcal{U}$ can be viewed as a database. A public key pk, a private key sk, and a linkable tag tag_H in $\mathcal{U}$ can serve as unique identifiers for rows to retrieve associated information. For example, $\mathcal{U}[\mathsf{pk}].\mathsf{acc}$ denotes the account associated with the public key pk. PreBal denotes the previous balance, i.e., the balance before the last transaction involving the account. CurBal denotes the current balance. IsCrpt denotes the "is corrupted" flag.

Oracles. The oracles $\mathcal{O}$ accessed by an adversary $\mathcal{A}$ are defined below.

9. In practice, this is managed by smart contracts or consensus algorithms, which is beyond the scope of this work.

$\mathcal{U}$:	pk	sk	tag_H	acc	PreBal	CurBal	IsCrpt

Table 7: Structure of the list $\mathcal{U}$ used in the security model

- $AdGen(i)$: input a query number i, it runs $(\mathsf{pk}_i, \mathsf{sk}_i, \pi_i) \leftarrow$ AddrGen() and outputs pk_i. It adds $(\mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{U}$ where IsCrpt is set to 0, automatically updates the linkable tag tag_H to $\mathcal{U}$ as the epoch H increases and the remaining fields are left empty.
- $Corrupt$(pk): input a public key pk, if $\mathcal{U}$[pk] cannot be found, it returns $\perp$ indicating failure. Otherwise, it sets $\mathcal{U}$[pk].IsCrpt to 1, and outputs $\mathcal{U}$[pk].sk and $\mathcal{U}$[pk].acc.
- $Transfer((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}, \ell_\mathsf{r}, \mathsf{amt}, \mathbb{S})$: input an anonymous set $(\mathsf{pk}_i)_{i\in[N]}$ including the sender public key $\mathsf{pk}_{\ell_\mathsf{s}}$, the recipient public key $\mathsf{pk}_{\ell_\mathsf{r}}$, a transaction amount amt, and the blockchain state $\mathbb{S}$. It first retrieve $\mathsf{sk}_{\ell_\mathsf{s}}$ from $\mathcal{U}$, runs $\mathsf{tx} \leftarrow \mathsf{AnTransfer}((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}, \ell_\mathsf{r}, \mathsf{sk}_{\ell_\mathsf{s}}, \mathsf{amt}, \mathbb{S})$ and $B \leftarrow \mathsf{Verify}(\mathsf{tx}, \mathbb{S})$. If $B = 0$ indicating the verification fails, it outputs $\perp$. Otherwise, it sets $\mathcal{U}[\mathsf{pk}_i].\mathsf{acc} = \mathcal{U}[\mathsf{pk}_i].\mathsf{acc} + \mathsf{ct}_i$, $\mathcal{U}[\mathsf{pk}_i].\mathsf{PreBal} = \mathcal{U}[\mathsf{pk}_i].\mathsf{CurBal}$ for all $i \in [N]$, and sets $\mathcal{U}[\mathsf{pk}_{\ell_\mathsf{s}}].\mathsf{CurBal} = \mathcal{U}[\mathsf{pk}_{\ell_\mathsf{s}}].\mathsf{CurBal} - \mathsf{amt}$, $\mathcal{U}[\mathsf{pk}_{\ell_\mathsf{r}}].\mathsf{CurBal} = \mathcal{U}[\mathsf{pk}_{\ell_\mathsf{r}}].\mathsf{CurBal} + \mathsf{amt}$. It returns tx.

Correctness. Informally, correctness requires that any user is able to spend her honestly generated account if the account has sufficient balance. An account-based private blockchain payments is said to be *ϵ-correct* if for any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, any $N \in \mathbb{Z}^+$, any blockchain state $\mathbb{S}$, any $(\mathsf{pk}_{\ell_\mathsf{s}}, \mathsf{sk}_{\ell_\mathsf{s}}, \pi_{\ell_\mathsf{s}}) \leftarrow \mathsf{AddrGen}()$ that has been appropriately funded or received transferred funds, and satisfies $\mathsf{m} \leftarrow \mathsf{ReadBalance}(\mathsf{sk}_{\ell_\mathsf{s}}, \mathbb{S})$, at any epoch $H \leftarrow \mathbb{S}$, with linkable tag $\mathsf{tag}_H \leftarrow \mathsf{TagGen}(\mathsf{sk}_{\ell_\mathsf{s}}, H)$ satisfying $\mathsf{LinkTag}(\mathsf{tag}_H, \mathbb{S}) = 0$, any amount $\mathsf{amt} \in \mathsf{V}$ satisfying $\mathsf{amt} \leq \mathsf{m}$, and any set $(\mathsf{pk}_i)_{i\in[N]\setminus\{\ell_\mathsf{s}\}}$ of arbitrarily registered decoy public keys along with the recipient's public key $\mathsf{pk}_{\ell_\mathsf{r}}$, the following probability holds

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{tx}, \mathbb{S}) \neq 0 : \\ \mathsf{tx} \leftarrow \mathsf{AnTransfer}((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}, \ell_\mathsf{r}, \mathsf{sk}_{\ell_\mathsf{s}}, \mathsf{amt}, \mathbb{S})\end{array}\right] \geq 1-\epsilon$$

If $\epsilon = 0$, then the protocol is said to be *perfectly correct*. If $\epsilon = \mathsf{negl}(\lambda)$, then it is said to be *statistically correct*.

Anonymity. Unlike UTXO-based RingCT, where the recipient's anonymity is guaranteed by using a one-time public key (stealth address), in the account-based setting, the identities of both the sender and the recipient are hidden among uncorrupted decoy accounts. Therefore, we adapt the anonymity security model from [10] to the account-based blockchain setting and divide it into two cases as follows.

Anonymity against non-recipient. The anonymity against non-recipient property requires that towards the non-recipient users, including the outsiders who do not involve the anonymous set and the insiders but not recipient, the addresses of both the sender and recipient are hidden among all the uncorrupted addresses in the anonymous set.

Definition B.2 (Anonymity against Non-Recipient)**.** An account-based private blockchain protocol is anonymous against non-recipient if the following holds for all PPT adversaries $\mathcal{A}$ and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$$\Pr[\mathcal{A} \text{ wins the game } \mathsf{Exp}_{\mathsf{AN}}(\mathsf{S})] \leq 1/2 + \mathsf{negl}(\lambda),$$

$$\Pr[\mathcal{A} \text{ wins the game } \mathsf{Exp}_{\mathsf{AN}}(\mathsf{R})] \leq 1/2 + \mathsf{negl}(\lambda)$$

where the game $\mathsf{Exp}_{\mathsf{AN}}$ is defined as follows:

1) $((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}^0, \ell_\mathsf{s}^1, \ell_\mathsf{r}^0, \ell_\mathsf{r}^1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp})$: $\mathcal{A}$ is given pp and access to all oracles, and then it outputs a set of registered public keys $(\mathsf{pk}_i)_{i\in[N]}$, the target sender indices $\ell_\mathsf{s}^0, \ell_\mathsf{s}^1 \in [N]$, the target recipient indices $\ell_\mathsf{r}^0, \ell_\mathsf{r}^1 \in [N]$, and some state information st to be used by $\mathcal{A}$ in the next stage.
2) $\mathsf{amt} \leftarrow \mathsf{V}$: The challenger samples an amount from set V.
3) $\mathsf{tx}_{i,j} \leftarrow \mathsf{AnTransfer}((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}^i, \ell_\mathsf{r}^j, \mathsf{sk}_{\ell_\mathsf{s}^i}, \mathsf{amt}, \mathbb{S})$ for $i, j \in \{0, 1\}$: Both ℓ_s^0 and ℓ_s^1 are spent to both ℓ_r^0 and ℓ_r^1 respectively, where $\mathsf{sk}_{\ell_\mathsf{s}^0}$, $\mathsf{sk}_{\ell_\mathsf{s}^1}$ are retrieved from $\mathcal{U}$. If $\mathsf{Verify}(\mathsf{tx}_{i,j}, \mathbb{S}) = 0$ for some $i, j \in \{0, 1\}$, then set all $\mathsf{tx}_{i,j} := \perp$.
4) $b_\mathsf{s}, b_\mathsf{r} \xleftarrow{\$} \{0, 1\}$.
5) $b'_\mathsf{s}, b'_\mathsf{r} \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{tx}_{b_\mathsf{s},b_\mathsf{r}}, \mathsf{st}, \mathbb{S})$: $\mathcal{A}$ is given access to all the oracles, the state st, the blockchain state $\mathbb{S}$ and one of the AnTransfer outputs. Then $\mathcal{A}$ outputs a guess for the real sender and recipient of $\mathsf{tx}_{b_\mathsf{s},b_\mathsf{r}}$.

$\mathcal{A}$ wins the game $\mathsf{Exp}_{\mathsf{AN}}(\mathsf{S})$ if $b'_\mathsf{s} = b_\mathsf{s}$ and wins the game $\mathsf{Exp}_{\mathsf{AN}}(\mathsf{R})$ if $b'_\mathsf{r} = b_\mathsf{r}$ if the following conditions hold:

- all public keys in $(\mathsf{pk}_i)_{i\in[N]}$ are honestly generated and verified in register algorithm,
- all accounts of $\mathsf{pk}_{\ell_\mathsf{s}^i}, \mathsf{pk}_{\ell_\mathsf{r}^i}$ for $i \in \{0, 1\}$ are not be corrupted (i.e. not queried to $Corrupt$),
- $\mathsf{tx}_{i,j} \neq \perp$ for all $i, j \in \{0, 1\}$,
- $\mathsf{pk}_{\ell_\mathsf{s}^i}$ for all $i \in \{0, 1\}$ are never queried to $Transfer$ at this epoch.

Anonymity against recipient. The anonymity against recipient property requires that towards the recipient, the address of sender is hidden among all the uncorrupted addresses in the anonymous set.

Definition B.3 (Anonymity against Recipient)**.** An account-based private blockchain payment protocol is anonymous against recipient if the following holds for all PPT adversaries $\mathcal{A}$ and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$$\Pr[\mathcal{A} \text{ wins the game } \mathsf{Exp}_{\mathsf{AR}}] \leq 1/2 + \mathsf{negl}(\lambda)$$

where the game $\mathsf{Exp}_{\mathsf{AR}}$ is defined as follows:

1) $((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}^0, \ell_\mathsf{s}^1, \ell_\mathsf{r}, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp})$: $\mathcal{A}$ is given pp and access to all oracles, and then it outputs a set of registered public keys $(\mathsf{pk}_i)_{i\in[N]}$, the target sender indices $\ell_\mathsf{s}^0, \ell_\mathsf{s}^1 \in [N]$, a recipient index $\ell_\mathsf{r} \in [N]$, and some state information st to be used by $\mathcal{A}$ in the next stage.
2) $\mathsf{amt} \leftarrow \mathsf{V}$: The challenger samples an amount from set V.
3) $\mathsf{tx}_i \leftarrow \mathsf{AnTransfer}((\mathsf{pk}_i)_{i\in[N]}, \ell_\mathsf{s}^i, \ell_\mathsf{r}, \mathsf{sk}_{\ell_\mathsf{s}^i}, \mathsf{amt}, \mathbb{S})$ for $i \in \{0, 1\}$: Both ℓ_s^0 and ℓ_s^1 are spent to ℓ_r respectively, where $\mathsf{sk}_{\ell_\mathsf{s}^0}$, $\mathsf{sk}_{\ell_\mathsf{s}^1}$ are retrieved from $\mathcal{U}$. If $\mathsf{Verify}(\mathsf{tx}_i, \mathbb{S}) = 0$ for some $i \in \{0, 1\}$, then set all $\mathsf{tx}_i := \perp$.
4) $b \xleftarrow{\$} \{0, 1\}$.

5) $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{tx}_b, \mathsf{st}, \mathbb{S})$: $\mathcal{A}$ is given access to all the oracles, the state st, the blockchain state $\mathbb{S}$ and one of the $\mathsf{AnTransfer}$ outputs. Then $\mathcal{A}$ outputs a guess bit b' for the real sender of tx_b.

$\mathcal{A}$ wins the game $\mathsf{Exp}_{\mathsf{AR}}$ if the following holds:

- all public keys in $(\mathsf{pk}_i)_{i\in[N]}$ are honestly generated and verified in register algorithm,
- both accounts of $\mathsf{pk}_{\ell_{\mathsf{s}}^i}$ for $i \in \{0,1\}$ are not be corrupted,
- $\mathsf{tx}_i \neq \perp$ for all $i \in \{0,1\}$,
- $\mathsf{pk}_{\ell_{\mathsf{s}}^i}$ for all $i \in \{0,1\}$ are never queried to $Transfer$ at this epoch.
- $b' = b$.

Confidentiality. Informally, confidentiality requires that no party other than the sender and the recipient can learn the transaction amount, even if the identities of the sender and recipient are revealed.

Definition B.4 (Confidentiality)**.** An account-based private blockchain payment protocol is confidential if the following holds for all PPT adversaries $\mathcal{A}$ and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$$\Pr[\mathcal{A} \text{ wins the game } \mathsf{Exp}_{\mathsf{CON}}] \leq 1/2 + \mathsf{negl}(\lambda)$$

where the game $\mathsf{Exp}_{\mathsf{CON}}$ is defined as follows:

1) $((\mathsf{pk}_i)_{i\in[N]}, \mathsf{amt}^0, \mathsf{amt}^1, \ell_{\mathsf{s}}, \ell_{\mathsf{r}}, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp})$: $\mathcal{A}$ is given pp and access to all oracles, and then it outputs a set of registered public keys $(\mathsf{pk}_i)_{i\in[N]}$, the target amounts $\mathsf{amt}^0, \mathsf{amt}^1 \in \mathsf{V}$, a sender index $\ell_{\mathsf{s}} \in [N]$, a recipient index $\ell_{\mathsf{r}} \in [N]$, and some state information st to be used by $\mathcal{A}$ in the next stage.
2) $\mathsf{tx}_i \leftarrow \mathsf{AnTransfer}((\mathsf{pk}_i)_{i\in[N]}, \ell_{\mathsf{s}}, \ell_{\mathsf{r}}, \mathsf{sk}_{\ell_{\mathsf{s}}}, \mathsf{amt}_i, \mathbb{S})$ for $i \in \{0,1\}$: amt^0 and amt^1 are transferred from ℓ_{s} to ℓ_{r} respectively, where $\mathsf{sk}_{\ell_{\mathsf{s}}}$ is retrieved from $\mathcal{U}$. If $\mathsf{Verify}(\mathsf{tx}_i, \mathbb{S}) = 0$ for some $i \in \{0,1\}$, then set all $\mathsf{tx}_i = \perp$.
3) $b \xleftarrow{\$} \{0,1\}$
4) $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{tx}_b, \mathsf{st}, \mathbb{S})$: $\mathcal{A}$ is given access to all the oracles, the state st, the blockchain state $\mathbb{S}$ and one of the $\mathsf{AnTransfer}$ outputs. Then $\mathcal{A}$ outputs a guess bit b' for the real amount of tx_b.

$\mathcal{A}$ wins the game $\mathsf{Exp}_{\mathsf{CON}}$ if the following holds:

- all public keys in $(\mathsf{pk}_i)_{i\in[N]}$ are honestly generated and verified in register algorithm,
- both accounts of $\mathsf{pk}_{\ell_{\mathsf{s}}}, \mathsf{pk}_{\ell_{\mathsf{r}}}$ are not be corrupted,
- $\mathsf{tx}_i \neq \perp$ for all $i \in \{0,1\}$,
- $\mathsf{pk}_{\ell_{\mathsf{s}}}$ is never queried to $Transfer$ at this epoch.
- $b' = b$.

Balance. Here, we adapt the balance security model from [6], [10] to the account-based blockchain setting. Informally, balance requires that an adversary cannot:

1) spend an inappropriate account, including an honestly generated account or an unlinked account;
2) spend her own account with an amount inconsistent with the sum of the transaction amounts of the other accounts in the anonymous set;
3) overdraft her own account;
4) double spend her own account within the same epoch.

Definition B.5 (Balance)**.** An account-based private blockchain payment protocol is balanced if the following holds for all PPT adversaries $\mathcal{A}$ and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

$$\Pr[\mathcal{A} \text{ wins the game } \mathsf{Exp}_{\mathsf{BAL}}] \leq \mathsf{negl}(\lambda)$$

where the game $\mathsf{Exp}_{\mathsf{BAL}}$ is defined as follows:

1) $(\mathsf{tx}^1, \ldots, \mathsf{tx}^t) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp})$: the adversary $\mathcal{A}$ is given access to all the oracles together with pp and outputs a set of t transactions $(\mathsf{tx}^1, \ldots, \mathsf{tx}^t)$, where $\mathsf{tx}^i = ((\mathsf{pk}^i_j)_{j\in[N]}, \mathbf{ct}^i, \mathsf{tag}^i, \Pi^i)$. Without loss of generality, we suppose the t transactions are all in the same epoch. [10]
2) From $i = 1$ to t, if $\mathsf{Verify}(\mathsf{tx}^i, \mathbb{S}) \neq 0$ and $\mathsf{LinkTag}(\mathsf{tag}^i, \mathbb{S}) = 0$, the challenger runs $\mathbb{S} \leftarrow \mathsf{RollOver}((\mathsf{pk}^i_j)_{j\in[N]}, \mathbf{ct}^i, \mathbb{S})$. As we discussed before, like Anonymous Zether [14], here we also assume each transaction takes effect immediately for simplicity.

$\mathcal{A}$ wins the game $\mathsf{Exp}_{\mathsf{BAL}}$ if the following holds

- for all $i \in \{1, ..., t\}$, all public keys in $(\mathsf{pk}^i_j)_{j\in[N]}$ are honestly-generated and verified in register algorithm. And all associated accounts are honestly maintained, i.e. the balance of each account in $(\mathsf{acc}[\mathsf{pk}^i_j])_{j\in[N]}$ is non-negative before all transactions of $\mathcal{A}$,
- $\nexists\ \mathsf{tag}^i = \mathsf{tag}^j$ for all $i, j \in [N]$ and $i \neq j$,
- $\mathsf{Verify}(\mathsf{tx}^i, \mathbb{S}) \neq 0$ and $\mathsf{LinkTag}(\mathsf{tag}^i, \mathbb{S}) = 0$ for all $i \in \{1, ..., t\}$,
- for all $j^* \in \{1, ..., t\}$, there exists at least one of the following cases after $\mathsf{RollOver}((\mathsf{pk}^{j^*}_i)_{i\in[N]}, \mathbf{ct}^{j^*}, \mathbb{S})$:
  - Case 1: for all $i \in [N]$, there exists $\mathsf{amt}^{j^*}_i < 0$ where $\mathsf{amt}^{j^*}_i := \mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{CurBal} - \mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{PreBal}$ and $\mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{IsCrpt} = 0$ or $\mathsf{pk}^{j^*}_i \neq \mathcal{U}[\mathsf{tag}^{j^*}].\mathsf{pk}$.
  - Case 2: $\sum_{i\in[N]} \mathsf{amt}^{j^*}_i \neq 0$ where $\mathsf{amt}^{j^*}_i := \mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{CurBal} - \mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{PreBal}$.
  - Case 3: for all $i \in [N]$, there exists $\mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{CurBal} < 0$.
  - Case 4: $\mathsf{tag}^{j^*} \notin (\mathcal{U}[\mathsf{pk}^{j^*}_i].\mathsf{tag})_{i\in[N]}$.

Attack scenarios of the balance model.

1) **Forgery**: The attacker attempts to create a transaction that either steals from uncorrupted accounts or spends from unlinked accounts, regardless of whether the latter are corrupted or not. The latter scenario can also be interpreted as a potential double-spending attack, since the attacker could later generate a valid proof as the legitimate sender and spend the same account again. This is captured by Case 1.
2) **Unbalanced amounts**: The attacker attempts to create a transaction in which the sum of the sender's and recipient's amounts (including amounts associated with other accounts in the anonymous set) does not equal zero. This is captured by Case 2.
3) **Overdraft**: The attacker attempts to create a transaction in which the amount transferred to the recipient exceeds the sender's available balance. In other words, the transaction causes at least one account in the anonymous set to have

10. As (Anonymous) Zether [13], [14] stated, after carefully choosing the constant E, it can be supposed that the transactions generated in any epoch will be verified in the same epoch.

a negative balance after execution. This is captured by Case 3.

4) **Double spending**: The attacker attempts to spend the same account more than once by generating distinct linkable tags. This is captured by Case 4.

Like [10], [11], our balance definition is presented as a single experiment. Moreover, our definition captures multiple attack scenarios and allows the adversary to output a set of transactions—where one transaction may serve as input to another—while Anonymous Zether [14] only allows the adversary to produce a single transaction.

In our formal definition, we explicitly state our assumptions to enable future extensions of the model by relaxing them. One possible enhancement is to remove the assumption in $\mathsf{Exp}_{\mathsf{BAL}}$ that all accounts in the anonymous set are generated and maintained honestly. This assumption is currently ensured by the soundness of previous transactions from earlier epochs. Removing it would significantly complicate the balance analysis. Hence, we retain this assumption in our work, noting that it is also implicitly assumed in (Anonymous) Zether [13], [14].

Appendix C. Security Proof

C.1. Anonymity

Theorem C.1 (Anonymity against Non-Recipients). *Lether in Section 3 is anonymous against non-recipients if* LNP22 *is simulatable, tag scheme is tag-anonymous, and Ref-AH mmPKE is* $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ *secure. More precisely, for any PPT adversary $\mathcal{A}$ against* $\mathsf{Exp}_{\mathsf{AN}}$, *there exists PPT adversaries $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$ against simulatability of* LNP22, *tag-anonymity of tag scheme, and* $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ *security of Ref-AH mmPKE, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{\mathsf{AN}}} = \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{Sim}} + \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{Tag\text{-}An}} + \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{mmIND\text{-}CPA}}.$$

Proof. Let $\mathcal{A}$ be a PPT adversary against anonymity with respect to non-recipients. We define the following sequence of games. Denote $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_i}$ as the advantage of $\mathcal{A}$ in winning Game_i.

Game$_0$: This is identical to $\mathsf{Exp}_{\mathsf{AN}}$. Thus, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{\mathsf{AN}}}$.

Game$_1$: This game is the same as Game_0, except that the challenger replaces the proof Π with a simulated proof generated by the simulator of LNP22.

Therefore, there exists a PPT adversary $\mathcal{B}_0$ whose running time is approximately that of $\mathcal{A}$ such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}\right| = \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{Sim}}.$$

Game$_2$: This game is the same as Game_1, except that the challenger replaces the linkable tag tag with a uniformly random tag sampled from the tag space.

Note that no queries of $Corrupt$ or $Transfer$ at the current epoch is allowed for the challenge sender public keys $\mathsf{pk}_{\ell_s}^0$ and $\mathsf{pk}_{\ell_s}^1$. Thus, there exists a PPT adversary $\mathcal{B}_1$ whose running time is approximately that of $\mathcal{A}$ such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}\right| = \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{Tag\text{-}An}}.$$

Game$_3$: This game is the same as Game_2, except that the challenger modifies the amount ciphertext $\mathbf{ct}$.

Specifically, the challenger changes the ciphertext $\mathbf{ct} = (\mathbf{c}, (c_i)_{i\in[N]})$ to $(\mathbf{u}, (v_i)_{i\in[N]})$, where $\mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^n)$ is sampled uniformly at random. For the uncorrupted accounts, each $v_i \leftarrow \mathcal{U}(\mathcal{R}_q)$ is also uniformly sampled. For the corrupted accounts, the ciphertext is constructed as $v_i := \langle \mathbf{u}, \mathbf{s}_i\rangle + h_i + \lfloor q/(2t+2)\rceil \cdot m_i$ where $m_i = 0$ because the corrupted accounts are all decoys.

Therefore, based on the security of mmPKE established in [24], there exists a PPT adversary $\mathcal{B}_2$ whose running time is approximately that of $\mathcal{A}$ such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3}\right| = \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{mmIND\text{-}CPA}}.$$

Note that the accounts of the challenge senders and recipients $\mathsf{pk}_{\ell_s}^i$, $\mathsf{pk}_{\ell_r}^i$ for $i \in \{0,1\}$ are uncorrupted. Hence, the ciphertexts for them are uniformly random. Furthermore, the output of AnTransfer is independent of $\mathsf{pk}_{\ell_s}^i$ and $\mathsf{pk}_{\ell_r}^i$ for $i \in \{0,1\}$, and also independent of b_s and b_r. Thus, $\mathcal{A}$ has advantage at most $1/2$ in guessing b_s or b_r in Game_3.

Collecting all the games from Game_0 to Game_3, we obtain the anonymity against non-recipients. □

Theorem C.2 (Anonymity against Recipient). *Lether in Section 3 is anonymous against recipient if* LNP22 *is simulatable, tag scheme is tag-anonymous, and Ref-AH mmPKE is* $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ *secure. More precisely, for any PPT adversary $\mathcal{A}$ against* $\mathsf{Exp}_{\mathsf{AR}}$, *there exists PPT adversaries $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$ against simulatability of* LNP22, *tag-anonymity of tag scheme, and* $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ *security of Ref-AH mmPKE, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{\mathsf{AR}}} = \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{Sim}} + \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{Tag\text{-}An}} + \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{mmIND\text{-}CPA}}.$$

Proof sketch. The proof is analogous to Theorem C.1; therefore, we provide only a sketch. **Game**$_0$ is identical to $\mathsf{Exp}_{\mathsf{AR}}$. **Game**$_1$, **Game**$_2$, and **Game**$_3$ are the same as those in Theorem C.1, except that in **Game**$_3$, since the recipient is corrupted, the challenger constructs the ciphertext as $v_i = \langle \mathbf{u}, \mathbf{s}_{\ell_r}\rangle + h_i + \lfloor q/(2t+2)\rceil m$. Thus, in **Game**$_3$, the adversary $\mathcal{A}$ has a success probability of $1/2$ in outputting $b' = b$. □

C.2. Confidentiality

Theorem C.3 (Confidentiality). *Lether in Section 3 is confidential if* LNP22 *is simulatable and Ref-AH mmPKE is* $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ *secure. More precisely, for any PPT adversary $\mathcal{A}$ against* $\mathsf{Exp}_{\mathsf{CON}}$, *there exists PPT adversaries $\mathcal{B}_0$, $\mathcal{B}_1$ against simulatability of* LNP22, *and* $\mathsf{mmIND\text{-}CPA}^{\mathsf{KOSK}}$ *security of Ref-AH mmPKE, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Exp}_{\mathsf{CON}}} = \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{Sim}} + \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{mmIND\text{-}CPA}}.$$

Proof sketch. The proof is analogous to Theorem C.1, so we provide only a sketch. **Game**$_0$ is identical to $\mathsf{Exp}_{\mathsf{CON}}$. **Game**$_1$ is the same as **Game**$_1$ in Theorem C.1. **Game**$_2$ corresponds to **Game**$_3$ in Theorem C.1. Thus, in **Game**$_2$, the adversary $\mathcal{A}$ has a success probability of $1/2$ in outputting $b' = b$. □

C.3. Balance

Theorem C.4 (Balance)**.** *Lether in Section 3 is balanced if* $\mathsf{LNP22}$ *is sound, tag scheme is event-oriented linkable, and* $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ *assumption for* $\bar{\chi} := \mathcal{U}(\mathbb{S}_\nu)$ *is hard.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the balance property in the game $\mathsf{Exp}_{\mathsf{BAL}}$. We argue that the probability that $\mathcal{A}$ wins this game is negligible. We divide the proof into the following four cases.

Case 1 (Forgery): Let $\mathcal{E}_{\text{forge}}$ denote the event where $\mathcal{A}$ wins the game such that there exists a transaction tx^{j^*} for some $j^* \in \{1,...,t\}$, and for all $i \in [N]$, there exists $\mathsf{amt}_i^{j^*} < 0$, where $\mathsf{amt}_i^{j^*} := \mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{CurBal} - \mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{PreBal}$, and either $\mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{IsCrpt} = 0$ or $\mathsf{pk}_i^{j^*} \neq \mathcal{U}[\mathsf{tag}^{j^*}].\mathsf{pk}$.

Since $\mathcal{A}$ is able to generate a valid proof Π, the extractability of $\mathsf{LNP22}$ implies the existence of an extractor that can extract a witness satisfying R_{an} in Equation (3.2).

For the first case, from the soundness of verifiable encryption for mmPKE (Lemma 2.9), we obtain that each amount in the ciphertext satisfies $\mathsf{amt}_i^{j^*} = (b_i^{(\mathsf{r})} - b_i^{(\mathsf{s})}) \cdot \hat{m}$, where $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{m} \rangle \geq 0$. Therefore, for some $i^* \in [N]$ such that $\mathsf{amt}_{i^*}^{j^*} < 0$, it must hold that $b_{i^*}^{(\mathsf{s})} = 1$. The extracted private key $\bar{\mathbf{s}}_{\ell_{\mathsf{s}}}$ must satisfy $\|(\bar{\mathbf{s}}_{\ell_{\mathsf{s}}} || \sum_{i\in[N]} b_i^{(\mathsf{s})} \cdot \mathbf{b}_i - \mathbf{A}^\top \bar{\mathbf{s}}_{\ell_{\mathsf{s}}})\|_\infty \leq \nu$. Given that $\mathcal{U}[\mathsf{pk}_{i^*}^{j^*}].\mathsf{IsCrpt} = 0$, i.e., $\mathcal{A}$ does not possess the private key corresponding to $\mathbf{b}_{i^*}$, and $b_{i^*}^{(\mathsf{s})} = 1$, $b_i^{(\mathsf{s})} = 0$ for all $i \in [N] \setminus \{i^*\}$, the adversary can solve the $\mathsf{MLWE}_{\mathcal{R},n,m,q,\bar{\chi}}$ instance on input $([\mathbf{I} \mid \mathbf{A}], \mathbf{b}_{i^*})$, where $\bar{\chi} := \mathcal{U}(\mathbb{S}_\nu)$—leading to a contradiction.

In the second case, it reduces to breaking the event-oriented linkability property of the tag scheme, as formalized in Lemma 2.21. Specifically, for some $i^* \in [N]$ such that $\mathsf{amt}_{i^*}^{j^*} < 0$ and $\mathsf{pk}_{i^*}^{j^*} \neq \mathcal{U}[\mathsf{tag}^{j^*}].\mathsf{pk}$, it implies that $\mathcal{A}$ is able to use the private key corresponding to $\mathsf{pk}_{i^*}^{j^*}$ to generate a valid tag that is unlinkable to $\mathsf{pk}_{i^*}^{j^*}$, which contradicts the event-oriented linkability of the tag scheme.

Case 2 (Unbalanced Amounts): Let $\mathcal{E}_{\text{unbalance}}$ denote the event where $\mathcal{A}$ wins the game such that there exists a transaction tx^{j^*} for some $j^* \in \{1,...,t\}$ satisfying $\sum_{i\in[N]} \mathsf{amt}_i^{j^*} \neq 0$, where $\mathsf{amt}_i^{j^*} := \mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{CurBal} - \mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{PreBal}$.

Since the adversary $\mathcal{A}$ is able to generate a valid proof Π, the extractability of $\mathsf{LNP22}$ ensures that there exists an extractor that can extract a witness satisfying R_{an} as defined in Equation (3.2).

From the soundness of verifiable encryption in mmPKE (Lemma 2.9), we know that each amount in the ciphertext satisfies $\mathsf{amt}_i^{j^*} = (b_i^{(\mathsf{r})} - b_i^{(\mathsf{s})}) \cdot \hat{m}$, where $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{m} \rangle \geq 0$. Summing over all $i \in [N]$, we obtain:

$$\sum_{i\in[N]} \mathsf{amt}_i^{j^*} = \sum_{i\in[N]} (b_i^{(\mathsf{r})} - b_i^{(\mathsf{s})}) \cdot \hat{m} = (\sum_{i\in[N]} b_i^{(\mathsf{r})} - \sum_{i\in[N]} b_i^{(\mathsf{s})}) \cdot \hat{m}.$$

Since both $\sum_{i\in[N]} b_i^{(\mathsf{r})} = 1$ and $\sum_{i\in[N]} b_i^{(\mathsf{s})} = 1$, we have $\sum_{i\in[N]} \mathsf{amt}_i^{j^*} = (1-1) \cdot \hat{m} = 0$. This contradicts the assumption that the sum is non-zero and thus violates the soundness of $\mathsf{LNP22}$.

Case 3 (Overdraft): Let $\mathcal{E}_{\text{overdraft}}$ denote the event where $\mathcal{A}$ wins the game such that there exists a transaction tx^{j^*} for some $j^* \in \{1,\ldots,t\}$ satisfying $\mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{CurBal} < 0$ for some $i \in [N]$.

Since $\mathcal{A}$ can produce a valid proof Π, the extractability of $\mathsf{LNP22}$ ensures that an extractor can recover a witness satisfying R_{an} as defined in Equation (3.2).

From the soundness of verifiable encryption in mmPKE (Lemma 2.9), each amount in the ciphertext satisfies $\mathsf{amt}_i^{j^*} = (b_i^{(\mathsf{r})} - b_i^{(\mathsf{s})}) \cdot \hat{m}$, with $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{m} \rangle \geq 0$. Let $\ell_{\mathsf{s}} := i^*$ be the index such that $b_{i^*}^{(\mathsf{s})} = 1$. Then, only the balance of account $\mathsf{acc}[\mathsf{pk}_{\ell_{\mathsf{s}}}^{j^*}]$ will be decremented by the transaction.

Assuming that all accounts have non-negative balances before the transaction, it suffices to argue that the post-transaction balance of $\mathsf{acc}[\mathsf{pk}_{\ell_{\mathsf{s}}}^{j^*}]$ remains non-negative.

From the unforgeability of verifiable decryption in mmPKE (Lemma 2.12), the decrypted balance satisfies $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{m}' - \vec{t}_d \rangle \in \{0, \ldots, 2^k - 1\}$ which implies that the balance is non-negative. This contradicts the assumption in $\mathcal{E}_{\text{overdraft}}$ and therefore violates the soundness of $\mathsf{LNP22}$.

Case 4 (Double-Spend): Let $\mathcal{E}_{\text{dspend}}$ denote the event where $\mathcal{A}$ wins the game such that there exists a transaction tx^{j^*} for some $j^* \in \{1,\ldots,t\}$ satisfying $\mathsf{tag}^{j^*} \notin (\mathcal{U}[\mathsf{pk}_i^{j^*}].\mathsf{tag})_{i\in[N]}$.

Since $\mathcal{A}$ can produce a valid proof Π, the extractability of $\mathsf{LNP22}$ ensures that an extractor can recover a witness satisfying R_{an} as defined in Equation (3.2). In particular, the extracted binary vector $\mathbf{b}^{(\mathsf{s})}$ has Hamming weight 1. Let $\ell_{\mathsf{s}} := i^*$ be the unique index such that $b_{i^*}^{(\mathsf{s})} = 1$.

Hence, the adversary effectively generates a valid tag tag^{j^*} using the private key of $\mathsf{pk}_{\ell_{\mathsf{s}}}^{j^*}$ that is not linked to the corresponding account in the system state. This directly contradicts the event-oriented linkability property of the tag scheme established in Lemma 2.21. □