

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

Hongxiao Wang
The University of Hong Kong
Hong Kong, China
hxxwang@cs.hku.hk

Ron Steinfeld
Monash University
Melbourne, Australia
Ron.Steinfeld@monash.edu

Muhammed F. Esgin
Monash University
Melbourne, Australia
Muhammed.Esgin@monash.edu

Siu-Ming Yiu
The University of Hong Kong
Hong Kong, China
smyiu@cs.hku.hk

Abstract

We introduce Lether, the *first* practical 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 (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.

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 optimize the implementations of LNP22 lattice-based zero-knowledge proof system (Crypto '22) in the LaZer library (CCS '24), to support efficient batching of various proof components. Overall, for a typical transaction, the total communication cost becomes about 68 KB, with the associated zero-knowledge proof accounting for about 51 KB of this total. Each of proof generation and verification take a fraction of a second 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 lattice settings.

Keywords

Post-Quantum, Lattices, Zero-Knowledge, Account-Based Blockchain

1 Introduction

Rapid progress in quantum computing [10] 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, recently, the Ethereum Foundation [20] 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 [39], Monero [37] (with a series of RingCT protocols [15, 16, 27, 41, 45, 48, 49]), and Quisquis [18]. Among them, only MatRiCT [16] and MatRiCT⁺ [15] 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 [7], named Zether, which provided a blueprint for achieving privacy and anonymity. Building on this, Anonymous Zether [11] 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 multiple directions, including forward security [24] and full anonymity [36, 38].² As discussed in [7, 11, 18, 24], 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 [7].

To the best of our knowledge, there is currently *no* private payment scheme for *account-based* blockchains in the *post-quantum* setting. As detailed below, the main reason is the lack of essential

¹An anonymity level of $1/N$ indicates that a real spender’s account is hidden within an anonymity set of size N .

²However, to date, works such as [36, 38] that achieve full anonymity in account-based blockchain payments remain largely *theoretical* and lack *practical implementations*, even in the classical (quantum-vulnerable) setting, let alone in post-quantum settings, due to their reliance on somewhat inefficient building blocks such as FHE, accumulators, or complex NIZKS.

building blocks: (i) unbounded-level additively-homomorphic multi-message multi-recipient public key encryption (mmPKE), and (ii) event-oriented linkable ring signatures with support for multiple tags (events).

1.1 Paradigm of Anonymous Zether

We first recall the paradigm of Anonymous Zether [7, 11] that we also follow. Originally, Anonymous Zether was not presented in a modular fashion, and the core building blocks with all of their required features were not as explicit. Here, we identify three core components needed for Anonymous Zether and make their required features explicit:

- A *multi-message multi-recipient public key encryption (mmPKE) scheme* [5, 26] that supports unbounded-level additive homomorphism, verifiable multi-encryption, and verifiable decryption. 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 compactness feature is critical in keeping the transactions efficient, particularly in the lattice setting where ciphertexts/proofs are large.
- An *event-oriented linkable ring signature* [3, 43] (also known as a prefix/scoped linkable ring signature [6, 25]) that allows signers to anonymously produce *at most* one signature for each *event* using their long-term private keys.³
- A *highly modular non-interactive zero-knowledge (NIZK) proof system* that integrates the verifiable mmPKE, the event-oriented linkable ring signature, a proof that the spender’s amount equals the recipient’s amount (i.e., a balance proof), and a proof that spender’s post-transaction balance together with the transaction amount lies within a valid range (i.e., a range proof).

In Anonymous Zether, the balance bal of each account is encrypted under the account public key pk , yielding a ciphertext $\text{acc}[\text{pk}] \leftarrow \text{Enc}(\text{pk}, \text{bal})$. The ciphertext is stored on the blockchain and can be indexed by the corresponding public key. We outline how the spender makes a transaction as follows:

- First, the spender selects a set of public keys $(\text{pk}_i)_{i \in [N]}$ as the *anonymity set*, including her own public key pk_s and the recipient’s public key pk_r .
- Second, the spender performs a verifiable multi-encryption of the amount vector as $\text{ct} := (\text{ct}_i)_{i \in [N]} \leftarrow \text{mmEnc}((\text{pk}_i)_{i \in [N]}, (\text{m}_i)_{i \in [N]})$, where $\text{m}_s = -\text{amt}$, $\text{m}_r = \text{amt}$, $\text{m}_i = 0$ for all $i \in [N] \setminus \{s, r\}$, and the amount $\text{amt} \in [0, \text{MAX}]$. 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 spender’s or recipient’s public key.
- Third, the spender proves via verifiable decryption that the updated balance of her account is non-negative, i.e., $0 \leq \text{bal}'_s \leftarrow \text{Dec}(\text{acc}[\text{pk}_s] + \text{ct}_s, \text{sk}_s)$. This ensures that the spender is not *overdrawn*, i.e., her initial balance satisfies $\text{bal}_s \geq \text{amt}$.

³As shown in [46], event-oriented linkability is the *most general* form: by setting the event as a fixed string, the ring, or the message, one can obtain one-time, ring-based, or message-based linkability, respectively.

- Fourth, the spender produce an *event-oriented* linkable ring signature for the current *epoch* H ⁴ to anonymously authorizes the transaction, which can be linked using a tag tag_H . Specifically, the spender must prove knowledge of her private key sk_s in the anonymity set $(\text{pk}_i)_{i \in [N]}$ and demonstrate that the tag tag_H is correctly formed. This ensures that the spender cannot *double-spend* in the current epoch H .
- Finally, the spender outputs a proof Π attesting that all the above conditions hold, along with the multi-recipient ciphertext ct and the linkable tag tag_H .

Once the proof Π is verified,⁵ the system updates all accounts in the anonymity set by homomorphically evaluating $\text{acc}[\text{pk}_i] := \text{acc}[\text{pk}_i] + \text{ct}_i$ for all $i \in [N]$. Here, the unbounded-level homomorphism property ensures that every account is updated correctly.

1.2 Existing Challenges

In attempting to build a post-quantum account-based private blockchain payment protocol following the paradigm of Anonymous Zether [7, 11], we identify several significant challenges, as summarized below.

Challenge I: construct lattice-based unbounded-level additively homomorphic mmPKE along with efficient verification mechanisms. Currently, the only existing mmPKE scheme based on standard lattice assumptions, proposed in [44], does not support additive homomorphism and lacks efficient mechanisms for verifiable multi-encryption and decryption.⁶ What’s more, generally, lattice-based (mm)PKE schemes do not support *unbounded-level* additive homomorphic evaluation without resorting to expensive operations such as bootstrapping [21]. This limitation is inherent to lattice-based setting, as noise accumulates with each homomorphic operation. Moreover, the message encoding techniques employed in lattice-based RingCT protocols [15, 16] to avoid homomorphic evaluation are not applicable in our setting, as the spender cannot compute the so-called “corrector” terms without knowing the recipient’s balance. Therefore, designing a lattice-based mmPKE scheme that enables unbounded-level additively-homomorphic evaluation in an efficient manner—while also supporting efficient range proofs, verifiable multi-encryption, and verifiable decryption—has been a significant open problem.

Challenge II: construct lattice-based event-oriented linkable ring signatures for multiple events. To the best of our knowledge, there exists no *practical* lattice-based event-oriented linkable ring or group signature that supports *many* events (i.e., tags). The

⁴Here, the epoch H is the “event” of the event-oriented linkable ring signature. In Anonymous Zether [7, 11], time is divided into epochs. To prevent *front-running attacks*, each involved account, including the spender’s, is updated at the end of the epoch. However, because the spender’s update amount is *negative*, to prevent *double-spending attacks* within each epoch, the spender is allowed to spend *at most one transaction per epoch*. This “locking” mechanism is achieved by requiring the spender to generate an “epoch-based” linkable ring signature for each transaction. As discussed in [7, 11], with a carefully chosen epoch length, usability is expected to remain largely unaffected. We refer the reader to [7, 11] for further details.

⁵In Anonymous Zether [7, 11], it is assumed that the proof Π is verified before the end of the current epoch. We note that delayed validation may affect liveness; however, it does not compromise balance correctness or enable double spending.

⁶We also note that [28] proposes a lattice-based multi-message multi-recipient key encapsulation mechanism (mmKEM) but based on *non-standard* lattice assumption, named Oracle-MLWE.

Blockchain	Scheme	PQ	Anony.	Transaction				Refresh				Eval. Level
Model			N	$ \text{tx} $	$ \Pi $	t_p	t_v	$ \text{ref} $	$ \pi' $	t_p	t_v	T
Account-Based	Lether (this paper)	✓	16	67.8	51.1	0.53	0.34	54.9	43.8	0.42	0.32	60
	Anony. Zether [11]	×	16	6.0	3.7	2.50	0.15	—	—	—	—	—
UTXO-Based	MatRiCT ⁺ [15]	✓	11	43.0	29.0	0.10	≤ 0.01	—	—	—	—	—
	Monero [37]	×	16	2.1	2.0	0.08	0.02	—	—	—	—	—

Table 1: Summary of private blockchain payment protocols (sizes in KB, times in seconds). ‘—’ denotes not applicable. PQ indicates post-quantum security. For the Transaction and Refresh phases, we present the communication sizes ($|\text{tx}|$, $|\text{ref}|$), the sizes of the associated proofs ($|\Pi|$, $|\pi'|$)—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 fixed at $N = 16$ (Anony.), following the current Monero [37] setting. The evaluation level for additively-homomorphic operations on fresh or refreshed accounts is fixed at $T = 60$ (Eval. Level).

only existing constructions support only a single event (i.e., one-time) or a very limited number of events, as in [42], where the tag takes the form $\text{tag} := \text{As}$ with $\text{A} \leftarrow \text{hash}(\text{event})$ and the secret s is committed in advance. The main limitation of such constructions arises from the contradiction between the need for fresh noise (to ensure indistinguishability from random values) and the uniqueness required for tag linkability. Specifically, to argue indistinguishability between ℓ tags $(\text{A}_i \cdot s)_{i \in [\ell]}$ and random values, the secret s must include at least ℓ fresh noise components and be committed in advance—leading to the linear cost with the tag/event number, which is highly inefficient. Therefore, constructing a *practical* lattice-based event-oriented linkable ring or group signature that supports multiple events remains a significant challenge.

1.3 Our Contributions

Lether: a practical post-quantum account-based private blockchain payment system. In this work, we propose *Lether*, the *first* practical post-quantum account-based private blockchain payment system based on lattice assumptions. While Lether follows the high-level Anonymous Zether paradigm [7, 11], its realization in the post-quantum setting is non-trivial and requires addressing challenges absent in classical constructions (see Section 1.2), due to the inherently noisy nature of lattice-based cryptography. To this end, we develop two novel building blocks: (i) a verifiable refreshable additively-homomorphic mmPKE (Ref-AH mmPKE) scheme, and (ii) a plug-and-play event-oriented linkable tag scheme. We further optimize the implementation of the lattice-based NIZK scheme LNP22 [31] in the LaZer library [35] to efficiently combine these building blocks with other proofs (e.g., range proofs). These tools may also be of independent interest.

We emphasize that our contributions go beyond merely shifting Anonymous Zether [7, 11] to the post-quantum setting. Being the *first* post-quantum account-based private blockchain payment scheme, our work also paves the way for post-quantum secure counterparts of other practical yet quantum-vulnerable account-based blockchain payment protocols, such as the forward-secure variant *Pride CT* [24].

To accommodate the inherent constraints of the lattice-based setting, we introduce a lightweight mechanism, termed *refresh*, which has no direct analogue in classical constructions, where each account ciphertext is periodically refreshed to support an unbounded

number of transactions via homomorphic evaluations. This mechanism is particularly well-suited for blockchain environments, providing an efficient alternative to bootstrapping or large-modulus schemes in FHE. By a careful choice of parameters, the amortized overhead of a refresh operation is minimized to be below 1.5% in terms of both communication and computation. Furthermore, we believe that under a deployment where a sufficiently large fraction of registered accounts stay active and periodically refresh, the refresh mechanism has negligible impact on forming an effective anonymity set for transactions. A detailed discussion of the refresh mechanism is provided in Section 1.4.

Optimized implementations of NIZK for Lether. We also optimize the implementations of LNP22 [31] in the LaZer library [35] that are tailored to Lether. Briefly, our optimized NIZK implementation supports batching/aggregating various proofs (e.g., verifiable multi-encryption, integer proofs, and tag proofs). Overall, these optimizations reduce the proof size by about 20% and achieve roughly a 4× speedup, compared to the original LNP22 implementation in the LaZer library.

We provide a summary of private blockchain payment protocols across both post-quantum and traditional settings in Table 1. The number of bits supported for the balances and amounts in Lether is fixed at $k = 32$, which is the same as in (Anonymous) Zether [7, 11] and *Pride CT* [24], and can be extended to 64 bits by applying a simple transformation from [7]. Considering that the size of post-quantum constructions is often *at least an order of magnitude* larger than that of their traditional counterparts [4], we believe that Lether is already practical for real-world applications, particularly in terms of computational cost. We further note that although account-based blockchain payments are a bit less efficient than UTXO-based blockchain payments (even in the pre-quantum setting) due to differences in their underlying techniques, the former offers unique advantages, such as wallet efficiency and extensive functionalities, as discussed above, which typically UTXO-based schemes cannot provide.

We summarize our main building blocks below. For technical overview, we refer the reader to Section 2.

Verifiable Ref-AH mmPKE. We construct the *first* verifiable Ref-AH mmPKE scheme from lattices, extending the basic mmPKE in [44]. Our scheme supports T -level additive homomorphism, meaning that a fresh ciphertext can undergo up to T homomorphic additions. Here, we introduce the notion of *refreshability*, which

Scheme	Recip. N	$ \text{ct} $	$ \pi $	Add-Hom	Dec-Ind
Cons 3.5	16	17	28	✓	✓
[31]	16	16	304	×	✓
[30]	16	144	144	✓	×

Table 2: Comparison of verifiable encryption schemes for $N = 16$ recipients (Recip.). We report the sizes of the multi-recipient ciphertext $|\text{ct}|$ and its well-formedness proof $|\pi|$ in kilobytes (KB). We further indicate whether the schemes support additive homomorphism (Add-Hom) and decryption-time independent of adversary’s runtime (Dec-Ind).

Scheme	$ \pi $	Add-Hom	Exa-Pro
Cons 3.7	31 KB	✓	✓
[33]	≈ 50 KB	×	✓
[22]	≥ 1 MB*	×	×
[2]	≈ 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 [40]

Table 3: Comparison of verifiable decryption schemes. For a fair comparison, we adapt other works to support unforgeability and estimate the proof size $|\pi|$ for valid decryption. We further indicate whether the schemes support additive homomorphism (Add-Hom) and an exact-norm proof (Exa-Pro).

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 further formalize and realize three types of verifiability for Ref-AH mmPKE: *verifiable multi-encryption*, *verifiable decryption*, and *verifiable refresh*, where the resulting proof and ciphertext sizes are the primary contributors to the transaction communication.

For verifiable multi-encryption, as shown in Table 2, our construction outperforms the state-of-the-art in communication size for $N = 16$ recipients, achieving an order-of-magnitude reduction.⁷ 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 [2, 22, 33]. Informally, unforgeability ensures that no adversary can generate two valid proofs for different decryption outputs of the same (honestly generated) ciphertext under a *legitimate* private key. This property is *missing* in prior works but is crucial in practice, particularly under lattice-based assumptions. For example, in existing works [2, 22, 33], an adversary can use a *mismatching* private key to decrypt a ciphertext, resulting in an *incorrect* plaintext. Since these works do not check the *validity* of the input private key during verifiable decryption, an adversary can honestly run the decryption algorithm with a *mismatching*

⁷Our scheme also demonstrates a similar advantage over related constructions such as [14].

Scheme	Sign. Size	Event-Link	Multi-Tag
Cons 3.12 + [31]	93 KB	✓	✓
[31]	92 KB	×	×
[16]	148 KB	×	×
[42]	386 KB	✓	×

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

key and still generate a valid proof, thereby misleading others into accepting an *incorrect* plaintext.

To this end, we 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 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 and guarantees consistency between the input ciphertext and the refreshed ciphertext.

Plug-and-play event-oriented linkable tag scheme. We propose a plug-and-play event-oriented linkable tag scheme from lattices that can transform most existing lattice-based ring or group signature schemes including [13, 15, 16, 31, 32, 34, 47] to support event-oriented linkability—the *most general* form of linkability (as introduced above)—with *many* tags (events), and with *negligible* overhead. To the best of our knowledge, no existing lattice-based ring/group signature scheme supports such property.

We demonstrate the effectiveness of our technique by extending the state-of-the-art lattice-based group signature scheme from [31] to support event-oriented linkability. As shown in Table 4, our extended scheme increases the signature size by only about 1%, while supporting many tags.

New formal definitions for account-based private blockchain payments. As an additional contribution, we propose new formal definitions 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 [7, 11].

1.4 Discussion on Refresh

The refresh mechanism is introduced to address the inherent limitation of lattice-based homomorphic encryption, where noise accumulates with each operation and eventually threatens correctness. In Lether, refresh mechanism serves as a *lightweight* and *tunable* maintenance operation that restores evaluated ciphertexts to a fresh state, enabling continued homomorphic evaluation without resorting to expensive techniques such as bootstrapping. In this subsection, we further discuss its practical implications in terms of usability, potential risks, and mitigations.

Usability. Since refresh is a *low-frequency* maintenance operation with a small amortized cost, its impact on usability is *negligible* for

typical active users (e.g., frequent spenders). For inactive users who primarily receive funds, in practice, they are typically informed (e.g., via off-chain communication) when funds are sent. Thus, refresh can be naturally triggered when the user becomes active (e.g., checking funds), without requiring continuous online presence.

Moreover, the evaluation level T can be flexibly adjusted depending on deployment requirements, trading off refresh frequency and communication overhead (see Table 6). Since our construction incurs only $O(\log T)$ overhead, increasing T has a mild impact on performance. For example, when increasing T from 60 to 1000, the transaction size increases from approximately 68 KB to 80 KB, and the refresh size from approximately 55 KB to 65 KB, which remain practical.

Denial-of-service considerations. The refresh mechanism introduces a potential denial-of-service attack: an adversary may attempt to force a target account to refresh by repeatedly including it in anonymity sets. However, such attacks can be mitigated by *cost asymmetry*. Specifically, an adversary must generate T transactions to force a single refresh, while the target user only needs to perform one refresh operation. Increasing T (e.g., $T = 1000$) further raises the attack cost, making such attacks less practical in realistic deployments.

Privacy considerations. The counter associated with each account may reveal maintenance or activity state, but it does not introduce additional linkage beyond what is already observable from public transaction activity. Since all transactions are publicly recorded, activity levels can already be inferred from transaction history. We also note that prior systems such as Zether [7] use counters to prevent replay attacks.

Mitigations. We also provide several practical approaches can further mitigate the impact of refresh in real-world deployments.

- *Receiver-friendly variants.* Lether can be naturally extended to a PrideCT [24]-style setting, where sender anonymity is removed. In this case, only the sender’s counter must remain below T , while updates to receiver accounts can remain pending when their counters reach T . This allows inactive recipients to continue receiving funds without immediate refresh, and defer refresh until they become active.
- *Subsidized refresh.* Refresh operations can be supported by relayers or protocol-level mechanisms (e.g., fee sponsorship), so that the cost is not borne by the targeted user. Even if an adversary repeatedly triggers refresh, the burden can be shifted away from the victim, thereby weakening denial-of-service incentives.

2 Technical Overview

In this section, we provide an overview of our techniques to address the challenges listed in Section 1.2. We begin by showing how to extend the basic mmPKE scheme [44] to support additive homomorphism and refreshability, and how to realize its verifiability using LNP22 [31]. We then present the construction of a plug-and-play event-oriented linkable tag scheme based on Learning with Rounding (LWR), which is also instantiated via LNP22. Finally, we outline how to optimize the implementations of LNP22 to efficiently integrate these components and build the Lether system.

Ref-AH mmPKE. We start by recalling the basic mmPKE construction in [44], named mmCipher. 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

$$\mathbf{b}_i := \mathbf{A}^\top \mathbf{s}_i + \mathbf{e}_i \quad (2.1)$$

where $(\mathbf{s}_i, \mathbf{e}_i) \leftarrow \mathcal{U}(\mathbb{S}_v^m) \times \mathcal{U}(\mathbb{S}_v^n)$ are uniformly sampled from $[-v, \dots, v]$. To encrypt N messages $(\hat{m}_i \in \{0, 1\}^d \subseteq \mathcal{R}_2)_{i \in [N]}$ for N recipients with public keys $(\mathbf{b}_i)_{i \in [N]}$, the multi-recipient ciphertext $(c, (c_i)_{i \in [N]})$ is computed as

$$\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u, \quad (2.2)$$

$$c_i = \langle \mathbf{b}_i, \mathbf{r} \rangle + y_i + \lfloor q/2 \rfloor \cdot \hat{m}_i, \quad (2.3)$$

where $(\mathbf{r}, \mathbf{e}_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$ and $y_i \leftarrow \mathcal{D}_{\sigma_1}$ are sampled independently from the discrete Gaussian distributions with widths σ_0 and σ_1 , respectively. With suitable parameters, under MLWE assumption, the adversary (even the malicious recipients) cannot break the other recipients’ ciphertext.

For an individual ciphertext (c, c_i) , decryption is computed as $\lfloor c_i - \langle \mathbf{c}, \mathbf{s}_i \rangle \rfloor_2$. Using Equations (2.1) to (2.3), we obtain:

$$c_i - \langle \mathbf{c}, \mathbf{s}_i \rangle = \langle -\mathbf{s}_i \| \mathbf{e}_i, \mathbf{e}_u \| \mathbf{r} \rangle + y_i + \lfloor q/2 \rfloor \cdot \hat{m}_i, \quad (2.4)$$

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

The basic mmPKE [44] only supports binary messages in $\{0, 1\}^d$, and thereby cannot satisfy additive homomorphism.

Here, we adopt a simple yet efficient technique to address this limitation. Without loss of generality, we first set the integer message m (as well as the integer used in the Lether system) satisfying $m \in [0, \text{MAX}]$, where $\text{MAX} = 2^k - 1$ and $32 \leq k \leq d$. We then encode each integer message m in binary form $\hat{m} \in \mathcal{R}_2$ such that $m = \langle \vec{0}^{d-k} \| \vec{2}^k, \vec{\hat{m}} \rangle$, where $\vec{2}^k = (2^0, \dots, 2^{k-1})$ and $\vec{\hat{m}}$ 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:

$$c_i = \langle \mathbf{b}_i, \mathbf{r} \rangle + y_i + \lfloor q/(2T+2) \rfloor \cdot \hat{m}_i. \quad (2.5)$$

Thus, the homomorphic addition can be performed in coefficient-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. Now, our scheme supports up to T levels of additive homomorphism with *fresh/refreshed* ciphertexts, as long as the accumulated noise remains within required bounds, i.e., $\|h\|_\infty \leq \lfloor q/(4T+4) \rfloor$.

Finally, while unbounded-level homomorphism cannot be achieved due to possible message overflow and the accumulation of noise beyond the correctness bound, we circumvent this limitation by introducing a *refresh* mechanism. Roughly speaking, each recipient decrypts the evaluated ciphertext to obtain $\hat{m} \in \mathcal{R}_{2T+2}$, and then decodes it to an integer message $m := \langle \vec{0}^{d-k} \| \vec{2}^k, \vec{\hat{m}} \rangle$. Subsequently, the integer message m is re-encoded into binary form and re-encrypted with fresh randomness, yielding a refreshed ciphertext, which can again support T levels of homomorphic evaluation.

Looking ahead to the Lether scheme, our Ref-AH mmPKE offers several advantages: (i) The multi-recipient ciphertext size is significantly smaller (e.g., only 17 KB for $N = 16$ recipients). (ii) The ciphertext has a linear structure, which enables efficient proofs of well-formedness, especially when the modulus of the LNP22

matches that of Ref-AH mmPKE, as demonstrated later. (iii) The encoded message format supports efficient range proofs via LNP22, where only binary proofs are required and can be batched. (iv) Any fresh/refreshed ciphertext (account) supports at least T homomorphic evaluations with fresh ciphertexts (transactions).

Verifiability of Ref-AH mmPKE. We classify the verifiability of Ref-AH mmPKE used in Lether system into three types: *verifiable multi-encryption*, *verifiable decryption*, and *verifiable refresh*.

In our work, we employ LNP22 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 [31, 35] and summarized in Appendix B.3, these capabilities can be extended to prove integers (e.g., binary bits), polynomials with binary coefficients, and range proofs for ℓ_∞ -norms (since $\|\cdot\|_2 \geq \|\cdot\|_\infty$). Typically, the proof size of LNP22 is linear in both the witness size and the number of norm/range proofs.

Towards verifiable multi-encryption, we must prove that Equations (2.2) and (2.5) hold, together with the bounds

$$\|(\mathbf{r}, \mathbf{e})\|_\infty \leq \beta_0, \|(y_1 \| \cdots \| y_N)\|_\infty \leq \beta_1, \text{ and } \hat{m}_i \in \{0, 1\}^d \subseteq \mathcal{R}_2,$$

where β_0 and β_1 are the randomness bounds determined by the parameters of Ref-AH mmPKE scheme.

We define the witness as $\text{wit} := (\mathbf{r}, (\hat{m}_i)_{i \in [N]})$ and the statement as $\text{stat} := ((\mathbf{b}_i)_{i \in [N]}, (\mathbf{c}, (c_i)_{i \in [N]}))$. After setting the modulus of LNP22 equal to that of Ref-AH mmPKE in the implementation, we only need to prove the relations: $\|\mathbf{r}\|(\mathbf{c} - \mathbf{A}\mathbf{r})\|_\infty \leq \beta_0$ and

$$\left\| \begin{array}{c} c_1 - \langle \mathbf{b}_1, \mathbf{r} \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m}_1 \\ \vdots \\ c_N - \langle \mathbf{b}_N, \mathbf{r} \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m}_N \end{array} \right\|_\infty \leq \beta_1. \quad (2.6)$$

As mentioned before, under this setting, commitments/witnesses are required only for $\mathbf{r}$, $\{\hat{m}_i\}_{i \in [N]}$, without requiring any other randomness such as $\mathbf{e}_u$ or $\{y_i\}_{i \in [N]}$, which significantly reduces the proof size. In particular, this design yields a proof size for verifiable multi-encryption that is *independent* of the size of anonymity set in Lether, since only m_s and m_r (for spender and recipients, respectively) need to be committed, while all other messages (for decoy accounts) are fixed to zero.

Notably, to guarantee the security of Ref-AH mmPKE, the ℓ_∞ -norms of the randomness values, particularly y_i , are relatively large. As a result, we cannot prove their exact ℓ_2 -norms. Instead, we carefully adopt ARP to ensure soundness and correctness, and set $\beta_0 := \psi \cdot \sqrt{(m+n)d} \cdot \tau\sigma_0$ and $\beta_1 := \psi \cdot \sqrt{Nd} \cdot \tau\sigma_1$, where ψ is the relaxation factor of ARP and τ is the Gaussian tail bound. These randomness bounds are also used to derive the decryption error bound and the supported level of homomorphic evaluation.

Towards verifiable decryption, we need to guarantee the uniqueness of the decrypted message under the correct private key to ensure our stronger *unforgeability* notion. To achieve this, we add a proof of knowledge of the private key corresponding to the public key, i.e., $\|(\mathbf{s}, \mathbf{b} - \mathbf{A}^\top \mathbf{s})\|_\infty \leq \nu$. This approach can be applied to existing schemes [2, 22, 33] to enhance them with unforgeability.

To construct verifiable decryption, we prove Equation (2.4), i.e.,

$$\mathbf{c} - \langle \mathbf{c}, \mathbf{s} \rangle = h + \lfloor q/(2T+2) \rfloor \cdot \hat{m}, \quad (2.7)$$

with $\|h\|_\infty \leq \lfloor q/(4T+4) \rfloor$. Since the ℓ_∞ -norm bound on h is quite large—close to Ref-AH mmPKE modulus—we cannot use ARP when LNP22 shares the same modulus as Ref-AH mmPKE. Instead, we decompose h to its bits and prove the well-formedness of the bits and the reconstruction of h satisfying Equation (2.7).

Towards verifiable refresh, this construction 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 (i) prove that each coefficient of $\hat{m}$ lies in the range $[-T, T+1]$ using its binary decomposition $\mathbf{b}_m$; (ii) show that $\langle \vec{0}^{d-k} \| \vec{2}^k, \vec{\hat{m}} \rangle = \langle \vec{0}^{d-k} \| \vec{2}^k, \vec{\hat{m}}' \rangle$ and $\hat{m}' \in \{0, 1\}^d \subseteq \mathcal{R}_2$; and (iii) use $\hat{m}'$ for the subsequent verifiable encryption.

Plug-and-Play Event-Oriented Linkable Tag Scheme. In general, 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. We construct the tag scheme from an LWR-based pseudorandom function (PRF), such as the one in [12]. Specifically, the tag tag is defined as

$$\mathbf{v}_H = \lfloor \mathbf{A}_H \cdot \mathbf{s} \bmod \hat{q} \rfloor_{\hat{p}}, \quad (2.8)$$

where $\mathbf{A}_H \in \mathcal{R}_{\hat{q}}^{n' \times m} \leftarrow \text{hash}(\text{event})$ is derived from the event string, and $\mathbf{s}$ is the private key from Ref-AH mmPKE in Equation (2.1). We require that $\nu \ll \hat{q}$ such that $\|\mathbf{s}\|_\infty \leq \nu$ and that $\hat{p}$ divides $\hat{q}$.

Regarding security, we show that: (i) the pseudorandomness of the tag is guaranteed under the Module Learning with Rounding (MLWR) assumption. Furthermore, as analyzed in [12], a single private key can generate a (practically) unbounded number of tags (e.g., more than 2^{128}) for different events with suitable parameters; (ii) the non-frameability of the tag—namely, 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.

Then, we outline how to prove the well-formedness of the tag in Equation (2.8) (i.e., the proof of rounding) via LNP22. We first rewrite Equation (2.8) as

$$\hat{q}/\hat{p} \cdot \mathbf{v}_H \bmod \hat{q} \equiv \mathbf{A}_H \cdot \mathbf{s} - \mathbf{e}_H \bmod \hat{q}, \quad (2.9)$$

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

Here, the challenge is that $\hat{q}$ is not prime and does not match the modulus q of LNP22 or Ref-AH mmPKE, and therefore we cannot directly prove this relation modulo $\hat{q}$. To address this issue, we set $\hat{q} \ll q$ and transform Equation (2.9) 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 $\|\mathbf{v}\|_\infty \leq \hat{\beta}$ for $\hat{\beta} := (\hat{q}/\hat{p} \cdot \hat{p} + d\hat{m}\hat{q}\nu + \hat{q}/\hat{p} - 1)/\hat{q}$.

Therefore, the proof of the well-formedness of the tag is reduced to the following conditions, which can be efficiently realized using LNP22:

$$\|(-\hat{q}/\hat{p} \cdot \mathbf{v}_H + \mathbf{A}_H \cdot \mathbf{s} - \mathbf{e}_H)/\hat{q}\|_\infty \leq \hat{\beta}', \quad (2.10)$$

together with $\|(\mathbf{s}, \mathbf{b} - \mathbf{A}^\top \mathbf{s})\|_\infty \leq v'$ and $\mathbf{e}_H \in \{0, \dots, \hat{q}/\hat{p} - 1\}^{n'd}$, where $\hat{p}' := \psi\sqrt{n'd}\hat{p}$ and $v' := \psi\sqrt{(m+n)d}v$ are the corresponding ℓ_2 -norm bounds,⁸ and ψ is the relaxation factor of ARP.

Looking ahead to the Lether system, with carefully chosen parameters, we can batch the ARP for Equation (2.10) into the ARP for Equation (2.6) as follows:

$$\left\| \begin{array}{c} c_1 - \langle \mathbf{b}_1, \mathbf{r} \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m}_1 \\ \vdots \\ c_N - \langle \mathbf{b}_N, \mathbf{r} \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m}_N \\ (-\hat{q}/\hat{p} \cdot \mathbf{v}_H + \mathbf{A}_H \cdot \mathbf{s} - \mathbf{e}_H)/\hat{q} \end{array} \right\|_\infty \leq \beta',$$

where the new bound satisfies $\beta' \approx \sqrt{\hat{\beta}'^2 + \beta_1^2}$. Moreover, we set $\hat{q}/\hat{p} := 2$ to reduce the range proof of $\mathbf{e}_H$ to binary proofs. These optimizations allow the well-formedness of the linkable tag to be proven at essentially *no additional cost*, without affecting the security or usability of either the tag scheme or other components; hence, we refer to our tag scheme as “plug-and-play”.

Building Lether via optimized implementations of LNP22. At a high level, Lether follows the paradigm of Anonymous Zether [7, 11], combining our verifiable Ref-AH mmPKE with event-oriented linkable ring signature, unified through our optimized implementations of LNP22. The overall structure of the system is as follows.

Each user generates a public-private key pair $(\mathbf{b}, \mathbf{s})$ for Ref-AH 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 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 $m \in [0, \text{MAX}]$ as an individual ciphertext:

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

where $\hat{m} \in \mathcal{R}_2$ is a binary polynomial satisfying $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}} \rangle = m$, and T_d is a shift offset, i.e., a polynomial whose coefficients are all equal to T . For compatibility, we shift the message space of Ref-AH 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 $m \in [0, \text{MAX}]$, the spender first selects N public keys $(\mathbf{b}_i)_{i \in [N]}$, including her own public key $\mathbf{b}_s$, the recipient's public key $\mathbf{b}_r$, and additional decoy accounts' public keys $(\mathbf{b}_i)_{i \in [N] \setminus \{r, s\}}$, forming an anonymity set that hides both the spender's and the recipient's identities.

The spender then selects two binary indicator vectors $\mathbf{b}^{(s)}, \mathbf{b}^{(r)} \in \{0, 1\}^N \subset \mathbb{Z}_q^N$, where $b_s^{(s)} = 1, b_r^{(r)} = 1$, and all other entries are set to zero. These indicator vectors are used to index the spender and recipient in the transaction proof. Accordingly, the spender must prove the well-formedness of these vectors, namely that they are binary *integer* vectors whose entries sum to 1.

Subsequently, the spender verifiably multi-encrypts the amount m into a multi-recipient ciphertext $(\mathbf{c}, (c_i)_{i \in [N]})$ for $(\mathbf{b}_i)_{i \in [N]}$ via Ref-AH mmPKE, where $\mathbf{c} := \mathbf{A}\mathbf{r} + \mathbf{e}_u$ and for $i \in [N]$,

$$c_i := \langle \mathbf{b}_i, \mathbf{r} \rangle + y_i + \lfloor q/(2T+2) \rfloor \cdot (b_i^{(r)} - b_i^{(s)}) \cdot \hat{m}. \quad (2.11)$$

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

⁸As in our verifiable multi-encryption, we use ℓ_2 -norms to bound ℓ_∞ -norms, i.e., $\|\cdot\|_2 \geq \|\cdot\|_\infty$ during the proof.

This construction induces a *quadratic* relation among the witnesses $\mathbf{b}^{(s)}, \mathbf{b}^{(r)}$, and $\hat{m}$. Each message of c_i is defined as $\hat{m}_i := (b_i^{(r)} - b_i^{(s)}) \cdot \hat{m}$, which guarantee balance correctness, i.e., $\hat{m}_s + \hat{m}_r = 0$ and $\hat{m}_i = 0$ for all $i \in [N] \setminus \{s, r\}$, even including the special case $s = r$ (i.e., the spender is the recipient).

To prevent overdraft attacks, the spender must further provide a verifiable decryption showing that the resulting balance $\hat{m}'$ of her account, computed as $\sum_{i=1}^N b_i^{(s)} \cdot (\mathbf{u}_i + \mathbf{c}_i + c_i)$, is *non-negative*, i.e., $\langle \vec{0}^{d-k} || \vec{2}^k, \vec{\hat{m}}' - \vec{T}_d \rangle \in [0, \text{MAX}]$ and $\hat{m}' \in [0, 2T+1]^d$.

Finally, to authorize the transaction and prevent double-spending attacks, the spender generates an event-oriented (epoch-based) ring signature, proving knowledge of the private key $\mathbf{s}_s$ corresponding to the public key $\sum_{i=1}^N b_i^{(s)} \cdot \mathbf{b}_i$, together with the well-formedness of the linkable tag $\mathbf{v}_H := \lfloor \mathbf{A}_H \cdot \mathbf{s}_s \bmod \hat{q} \rfloor_{\hat{p}}$, where $\mathbf{A}_H \leftarrow \text{hash}(\text{Lether} || H)$ and H denotes 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 the counter reaches T , subsequent transactions are suspended until the user performs a verifiable refresh, after which the counter is reset. We provide implementation details in Section 4.2.

3 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 full preliminaries to Appendix B.

Preliminaries. Let $[n] = \{1, \dots, n\}$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$. Bold lowercase/uppercase letters denote vectors/matrices over $\mathcal{R}_q$ (e.g., $\mathbf{A} \in \mathcal{R}_q^{m \times n}$). For $\mathbf{a} \in \mathcal{R}_q^m$, we let $\vec{a} \in \mathbb{Z}_q^{md}$ denote its coefficient embedding. We use $\langle \cdot, \cdot \rangle$ for the inner product and $\|\cdot\|_2, \|\cdot\|_1, \|\cdot\|_\infty$ for the usual norms (applied coefficient-wise to polynomial elements/vectors). We write $x := y$ for assignment and $x \leftarrow \mathcal{D}$ for sampling. We use $\mathcal{U}(\mathbb{S}_v)$ to denote the uniform distribution over $\mathbb{S}_v := \{-v, \dots, v\}$ (or $\{0, \dots, v\}$ when a support $0 < \bar{v} \leq 2v+1$ is specified), and $\mathcal{D}_\sigma$ to denote the discrete Gaussian distribution. We write $\lfloor \cdot \rfloor$ for rounding. We use standard encoding/decoding operators as in the lattice literature. In particular, an integer message is encoded as a binary polynomial in $\mathcal{R}_2$ via a deterministic map $\text{BinPoly}(\cdot)$, and a polynomial element is encoded as a binary polynomial vector in $\mathcal{R}_2$ via a deterministic map $\text{Bit}(\cdot)$; both encodings can be decoded using the corresponding gadget vector.

Lattice assumptions. We rely on standard lattice hardness assumptions, including MLWE, MLWR, and MSIS, as well as Matrix Hint-MLWE. At a high level, Matrix Hint-MLWE models leakage/hints of the secret through a sampled matrix and admits reductions from standard MLWE under appropriate parameter choices.

LNP22 interface. We use LNP22 = (LNP22.Setup, LNP22.P, LNP22.V) as a black-box lattice-based NIZK proof system for proving and verifying relations R in this paper. We rely on the completeness, knowledge soundness, and simulatability guarantees established in LNP22 [31].

3.1 Verifiable Ref-AH mmPKE

We propose the *first* lattice-based verifiable Ref-AH mmPKE scheme. Compared to the basic mmPKE [44], we additionally introduce the properties of additive homomorphism, refreshability, and verifiability. The syntax of Ref-AH mmPKE is defined below, and the formal security properties are deferred to Appendix C.1.

Definition 3.1 (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 $\mathcal{C}_s$ consists of the following algorithms.

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

Correctness requires that ciphertexts decrypt correctly for all intended recipients. Chosen-plaintext security (CPA) guarantees security even in the presence of corrupted recipients. Additive homomorphism enables ciphertexts to be combined to obtain an encryption of the sum of the underlying messages, while refreshability enables ciphertext re-randomization without affecting decryption.

The lattice-based constructions of Ref-AH mmPKE are as follows.

Construction 3.2 (Ref-AH mmPKE). Let λ be a security parameter. Let m, n, d, q, N, v, 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 (c, c_i) from $(c, (c_i)_{i \in [N]})$.

Our Ref-AH mmPKE scheme shares the same correctness and security analysis as [44]⁹; we therefore only state the following lemmas. Additive homomorphism and refreshability are also easy to realize. In particular, for appropriate parameter settings (i.e., when the accumulated noise remains within bounds), our Ref-AH mmPKE supports additive homomorphism for at least T levels.

Lemma 3.3 (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 Construction 3.2. Denote $\zeta := \sum_{i \in [N]} \Pr [\|\langle \mathbf{e}_i, \mathbf{r} \rangle + y_i - \langle \mathbf{s}_i, \mathbf{e}_u \rangle\|_\infty \geq \lfloor q/(4T+4) \rfloor]$. We say our Ref-AH mmPKE in Construction 3.2 is ζ -correct.

⁹The analysis is obtained by combining that of extended reproducible PKE with the mmPKE compiler.

Algorithm 1 Ref-AH mmPKE

```

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

5: procedure mmKGen()
6:    $\mathbf{s}, \mathbf{e} \leftarrow \mathcal{U}(\mathbb{S}_v^m) \times \mathcal{U}(\mathbb{S}_v^n)$ 
7:    $\mathbf{b} := \mathbf{A}^\top \mathbf{s} + \mathbf{e}$ 
8:   return  $(\text{pk} := \mathbf{b}, \text{sk} := \mathbf{s})$ 
9: end procedure

10: procedure mmEnc( $(\text{pk}_i = \mathbf{b}_i)_{i \in [N]}, (m_i = m_i \in \mathbb{Z}_{2^k})_{i \in [N]}$ )
11:    $\mathbf{r}, \mathbf{e}_u \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$ 
12:    $\mathbf{c} := \mathbf{A} \mathbf{r} + \mathbf{e}_u$ 
13:   for all  $i \in [N]$ 
14:      $y_i \leftarrow \mathcal{D}_{\sigma_1}$ 
15:      $\hat{m}_i \in \mathcal{R}_2 \leftarrow \text{BinPoly}(m_i)$ 
16:      $c_i := \langle \mathbf{b}_i, \mathbf{r} \rangle + y_i + \lfloor q/(2T+2) \rfloor \cdot \hat{m}_i$ 
17:   end for
18:   return  $\text{ct} := (c, (c_i)_{i \in [N]})$ 
19: end procedure

20: procedure mmDec( $\text{ct} = (c, c), \text{sk} = \mathbf{s}$ )
21:    $\hat{m} \leftarrow \lfloor c - \langle \mathbf{c}, \mathbf{s} \rangle \rfloor_{2T+2}$ 
22:   return  $m := \langle \hat{m}^{d-k} \rangle_{2^k}, \hat{m}$ 
23: end procedure

24: procedure mmRef( $\text{ct} = (c, c), (\text{pk}, \text{sk}) = (\mathbf{b}, \mathbf{s})$ )
25:    $m \leftarrow \text{mmDec}((c, c), \mathbf{s})$ 
26:   return  $\text{ct}' := (c', c') \leftarrow \text{mmEnc}(\mathbf{b}, m)$ 
27: end procedure

```

Lemma 3.4 (Security). Define the distribution $\chi := \mathcal{U}(\mathbb{S}_v)$, $\chi_0 := \mathcal{D}_{\mathbb{Z}^{(m+n+1)d}, \sqrt{\Sigma_1}}$, and $\chi_1 := \mathcal{D}_{\mathbb{Z}^{Nd}, \sqrt{\Sigma_y}}$, where $\Sigma_1 = \begin{pmatrix} \sigma_1 I_d & 0 \\ 0 & \sigma_0 I_{(m+n)d} \end{pmatrix}$, $\Sigma_y = \sigma_1 I_{Nd}$. Define the distribution $\mathcal{S}$ such that the matrix $\mathbf{R} \leftarrow \mathcal{S}$ can be sampled as $\mathbf{R} = \begin{pmatrix} \mathbf{0}^\top & -(\mathbf{s}_0 \parallel \dots \parallel \mathbf{s}_{N-1})^\top & (\mathbf{e}_0 \parallel \dots \parallel \mathbf{e}_{N-1})^\top \end{pmatrix} \in \mathcal{R}^{N \times (1+m+n)}$ where $\mathbf{s}_i \leftarrow \mathcal{U}(\mathbb{S}_v^n)$, $\mathbf{e}_i \leftarrow \mathcal{U}(\mathbb{S}_v^m)$ for each $i \in [N]$. Our Ref-AH mmPKE in Construction 3.2 is $\text{mmIND-CPA}^{\text{KOSK}}$ secure under $\text{MLWE}_{\mathcal{R}, n, m, q, \chi}$ and $\text{MatrixHint-MLWE}_{\mathcal{R}, m, n, q, \chi_0}^{N, \chi_1, \mathcal{S}}$ assumptions.

We then use LNP22 as a black-box NIZK to realize the three types of verifiability supported by our Ref-AH mmPKE: 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 ct .

In general, a verifiable (multi-)encryption scheme must satisfy three properties: completeness, simulatability, and soundness. Completeness requires that any honestly generated ciphertext and its 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 an invalid ciphertext. We defer the formal definition of soundness to Appendix C.1.

We present the constructions of verifiable multi-encryption for (Ref-AH) mmPKE as follows.

Construction 3.5 (Verifiable Multi-Encryption for mmPKE). Suppose the (Ref-AH) mmPKE shares the modulus with LNP22. After generating the multi-recipient ciphertext $(c, (c_i)_{i \in [N]})$ using $\text{mmEnc}((b_i)_{i \in [N]}, (m_i)_{i \in [N]}; (r, e_u), (y_i)_{i \in [N]})$, the prover takes as input the witness $\text{wit} := ((\hat{m}_i)_{i \in [N]}, r)$ where each $\hat{m}_i \leftarrow \text{BinPoly}(m_i)$ denotes the binary decomposition of message m_i for all $i \in [N]$, and the statement $\text{stat} := (A, (b_i)_{i \in [N]}, (c, (c_i)_{i \in [N]}))$. The proof is generated as $\pi \leftarrow \text{LNP22.P}(R_{\text{enc}}, \text{stat}, \text{wit})$ and verified by $0/1 \leftarrow \text{LNP22.V}(R_{\text{enc}}, \text{stat}, \pi)$, where the proof relation R_{enc} is defined as follows:

$$R_{\text{enc}} = \left\{ (A, (b_i)_{i \in [N]}, (c, (c_i)_{i \in [N]}); ((\hat{m}_i)_{i \in [N]}, r) : \begin{array}{l} \left\| \begin{array}{l} r \\ c - A\bar{r} \end{array} \right\|_{\infty} \leq \psi \cdot \beta_0, \\ \left\| \begin{array}{l} c_1 - \langle b_1, r \rangle - \lfloor \frac{q}{2T+2} \rfloor \cdot \hat{m}_1 \\ \vdots \\ c_N - \langle b_N, r \rangle - \lfloor \frac{q}{2T+2} \rfloor \cdot \hat{m}_N \end{array} \right\|_{\infty} \leq \psi \cdot \beta_1, \\ (\hat{m}_1 || \dots || \hat{m}_N) \in \{0, 1\}^{Nd} \end{array} \right\}, \quad (3.1)$$

where ψ is the relaxation factor, and $\beta_0 = \sqrt{(m+n)d} \cdot \tau\sigma_0$, $\beta_1 = \sqrt{Nd} \cdot \tau\sigma_1$ with Gaussian tail bound factor τ .

Its completeness and simulatability follow directly from those of LNP22. We establish its soundness through the following lemma and defer its proof to Appendix C.1.

Lemma 3.6 (Soundness in Verifiable Multi-Encryption). *Suppose LNP22 is knowledge sound. Then, our verifiable multi-encryption in Construction 3.5 is sound if the probability $\sum_{i \in [N]} \Pr[\|e_i, \bar{r}\| + \bar{y}_i - \langle s_i, \bar{e}_u \rangle]_{\infty} \geq \lfloor q/(4T+4) \rfloor$ is negligible, where (e_i, s_i) is the private key in mmPKE and $(\bar{r}, \bar{m}_i)$ is the extracted witness along with $\bar{e}_u := c - A\bar{r}$, $\bar{y}_i := c_i - \langle b_i, \bar{r} \rangle - \lfloor q/(2T+2) \rfloor \cdot \bar{m}_i$.*

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, called *unforgeability*, which enhances the original soundness. Briefly, unforgeability guarantees the *uniqueness* of the decrypted message m under the correct private key. We defer its formal definition to Appendix C.1.

Here, its completeness requires that a proof generated for an honestly decrypted message is always accepted. Its simulatability ensures the existence of a simulator such that no adversary can distinguish between real and simulated decrypted messages.

We present the constructions of verifiable decryption as follows.

Construction 3.7 (Verifiable Decryption for mmPKE). Suppose the (Ref-AH) mmPKE shares the modulus with LNP22. Suppose $v = 1$, $\bar{v} = 2$. After decrypting the individual ciphertext (c, c) under the public key b using $\text{mmDec}((c, c), s)$ to the encoded message $\hat{m}$ and obtain the decryption error $h := c - \langle c, s \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m}$, the prover takes as input the witness $\text{wit} := (s, b_h)$ and the statement $\text{stat} := (A, b, (c, c), \hat{m})$, where $b_h \leftarrow \text{Bit}(h)$ is the coefficient-wise binary decomposition of h . The proof is generated as $\pi \leftarrow \text{LNP22.P}(R_{\text{dec}}, \text{stat}, \text{wit})$ and verified by $0/1 \leftarrow \text{LNP22.V}(R_{\text{dec}}, \text{stat}, \pi)$,

where the proof relation R_{dec} is defined as follows:

$$R_{\text{dec}} = \left\{ (A, b, (c, c), \hat{m}); (s, b_h) : \begin{array}{l} c - \langle c, s \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m} = \sum_{i \in [o]} \delta_i \cdot b_h^{(i)}, \\ (s || (b - A^T s) || b_h) \in \{0, 1\}^{(m+n+o) \cdot d} \end{array} \right\}, \quad (3.2)$$

where $o = \lceil \log(\lfloor q/(4T+4) \rfloor) \rceil$, $b_h = (b_h^{(1)}, \dots, b_h^{(o)})$, $\delta = (\delta_1, \delta_2, \dots, \delta_o) = (2^0, 2^1, \dots, \lfloor q/(4T+4) \rfloor - 2^{\lfloor \log(q/(4T+4)-1) \rfloor})$.

The completeness and simulatability of our verifiable decryption follow directly from those of LNP22. We establish its unforgeability through the following lemma and defer its proof to Appendix C.1.

Lemma 3.8 (Unforgeability in Verifiable Decryption). *Suppose LNP22 is knowledge sound. Then, our verifiable decryption in Construction 3.7 is unforgeable if MSIS $_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard.*

Verifiable Refresh for mmPKE. After at most T additively homomorphic evaluations, one can decrypt a ciphertext (c, c) to obtain the message m , and then re-encrypt it into a fresh ciphertext (c', c') using fresh randomness $(r', e'_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_1}^m$. 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 for an honestly refreshed ciphertext is always accepted. Simulatability guarantees the existence of a simulator whose output is indistinguishable from a real refreshed ciphertext. Unforgeability ensures the well-formedness of the refreshed ciphertext and the consistency of the underlying messages. The formal definition of unforgeability is deferred to Appendix C.1.

Construction 3.9 (Verifiable Refresh for mmPKE). Suppose the (Ref-AH) mmPKE shares the modulus with LNP22. Suppose $v = 1$ and $\bar{v} = 2$. The verifiable refresh is shown in Algorithm 2. The proof is verified by $0/1 \leftarrow \text{LNP22.V}(R_{\text{ref}}, \text{stat}, \pi)$, where the proof relation R_{ref} is defined as follows:

$$R_{\text{ref}} = \left\{ \begin{array}{l} \text{stat} = (A, b, (c, c), (c', c')); \\ \text{wit} = (s, r', \hat{m}, \hat{m}', b_m, b_h) : \\ c - \langle c, s \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m} = \sum_{i \in [o]} \delta_i b_h^{(i)}, \\ \left\| \begin{array}{l} r' \\ c' - A\bar{r}' \end{array} \right\|_{\infty} \leq \psi \cdot \beta_0, \\ \left\| \begin{array}{l} c' - \langle b, r' \rangle - \lfloor q/(2T+2) \rfloor \cdot (\hat{m}' + T_d) \\ \langle \vec{0}^{d-k} || \vec{2}^k, \vec{m} - \vec{T}_d \rangle = \langle \vec{0}^{d-k} || \vec{2}^k, \vec{m}' \rangle, \\ \hat{m} = \sum_{i \in [o']} \delta'_i \cdot b_m^{(i)}, \\ \text{bin} \in \{0, 1\}^{(m+n+o'+o+1)d} \end{array} \right\|_{\infty} \leq \psi \cdot \beta_1, \end{array} \right\}, \quad (3.3)$$

where ψ is the relaxation factor, $\text{bin} := (s || (b - A^T s) || b_m || b_h || \hat{m}')$, $b_m = (b_m^{(1)}, \dots, b_m^{(o')})$, and $\delta' = (\delta'_1, \delta'_2, \dots, \delta'_{o'}) = (1, 2, \dots, 2T+2 - 2^{\lfloor \log(2T+1) \rfloor})$. Moreover, β_1 is defined as in Equation (3.1), while δ and b_h are defined as in Equation (3.2).

To fit the Lether setting, as discussed in Section 2, the message space is defined as $\mathcal{M} = [0, \dots, 2^k - 1]$, and the encoding space is shifted from $\{-T, \dots, T+1\}^d$ to $\{0, \dots, 2T+1\}^d$ by adding the constant offset T_d . Thus, we modify the message/encoding-related part of the proof relation R_{ref} .

Algorithm 2 VRefresh

Input: $\text{pk} = \mathbf{b}, \text{sk} = \mathbf{s}, \text{ct} = (\mathbf{c}, \mathbf{e})$

- 1: $\tilde{m} := \lfloor \mathbf{c} - \langle \mathbf{c}, \mathbf{s} \rangle \rfloor_{2T+2}$
- 2: $h := \mathbf{c} - \langle \mathbf{c}, \mathbf{s} \rangle - \lfloor q/(2T+2) \rfloor \cdot \tilde{m}$
- 3: $\mathbf{b}_m \leftarrow \text{Bit}(\tilde{m})$
- 4: $\mathbf{b}_h \leftarrow \text{Bit}(h)$
- 5: $\mathbf{m} := \langle \tilde{0}^{d-k} || \tilde{2}^k, \tilde{m} - \tilde{T}_d \rangle$
- 6: $\tilde{m}' \leftarrow \text{BinPoly}(\mathbf{m})$
- 7: $(\mathbf{r}', \mathbf{e}'_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$
- 8: $\mathbf{y}' \leftarrow \mathcal{D}_{\sigma_1}$
- 9: $\mathbf{c}' := \mathbf{A}\mathbf{r}' + \mathbf{e}'_u$
- 10: $\mathbf{c}' := \langle \mathbf{b}, \mathbf{r}' \rangle + \mathbf{y}' + \lfloor q/(2T+2) \rfloor \cdot (\tilde{m}' + T_d)$
- 11: $\pi' \leftarrow \text{LNP22.P}(R_{\text{ref}}, \text{stat}, \text{wit})$ where R_{ref} is defined in Equation (3.3)
- 12: **return** $\text{ref} := (\text{ct}' := (\mathbf{c}', \mathbf{e}'), \pi')$

Its completeness and simulatability follow directly from those of LNP22. We establish its unforgeability through the following lemma and defer its proof to Appendix C.1.

Lemma 3.10 (Unforgeability in Verifiable Refresh). *Suppose LNP22 is knowledge sound. Then, our verifiable refresh in Construction 3.9 is unforgeable if $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard and the probability $\Pr[\|(\mathbf{e}, \mathbf{r}') + \mathbf{y}' - \langle \mathbf{s}, \mathbf{e}'_u \rangle\| \geq \lfloor q/(4T+4) \rfloor]$ is negligible, where $(\mathbf{e}, \mathbf{s})$ is the private key in mmPKE and $(\mathbf{r}', \tilde{m}')$ is the extracted witness along with $\mathbf{e}'_u := \mathbf{c}' - \mathbf{A}\mathbf{r}', \mathbf{y}' := \mathbf{c}' - \langle \mathbf{b}, \mathbf{r}' \rangle - \lfloor q/(2T+2) \rfloor \cdot \tilde{m}'$.*

3.2 Event-Oriented Linkable Tag

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

Definition 3.11 (Tag Scheme). A tag scheme with a public-private key pair space $\mathcal{K}$, a tag space $\mathcal{T}$ consists of the following algorithms.

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

Following [6], we model the tag proof algorithm Prove as a *signature of knowledge* and omit the signed message for simplicity. In general, a signature of knowledge should satisfy *completeness*, *simulatability*, and *extractability*. Completeness requires that any honestly generated signature on a statement with a valid witness is always accepted. Simulatability ensures that there exists a simulator whose output is computationally indistinguishable from real signatures. Extractability requires an efficient extractor that can extract a witness from any accepting signature. For formal definitions of these properties, we refer the reader to [6, 8] for details.

We further formalize additional security properties for our tag scheme, namely *event-oriented linkability*, *multi-tag anonymity*, and *non-frameability*, adapted from [25]. Event-oriented linkability requires that any two valid tags generated by the same user for the same event are publicly linkable, while tags generated for different events remain unlinkable. Multi-tag anonymity ensures that, even given many tags generated across different events, no adversary can determine which user generates a particular tag, beyond what is trivially revealed by linkability within the same event. Non-frameability guarantees that no adversary can produce a valid tag that is linkable to an honest user's tag for a given event without possessing the corresponding private key. In particular, we explicitly capture anonymity in the setting where *many tags* are generated for different events. We defer these formal definitions of tag scheme in Appendix C.2.

Then, using LNP22 as a black-box NIZK proof system, we construct our tag scheme for our Ref-AH mmPKE as follows.

Construction 3.12 (Tag Scheme for mmPKE). Let mmKGen be a sub-algorithm of Ref-AH mmPKE scheme. Suppose $v = 1$ and $\bar{v} = 2$. 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 Ref-AH mmPKE.

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

$$R_{\text{tag}} = \left\{ (\mathbf{A}_H, \mathbf{b}, \mathbf{v}_H); (\mathbf{s}, \mathbf{e}_H) : \begin{aligned} &\|(\hat{q}/\hat{p} \cdot \mathbf{v}_H + \mathbf{e}_H - \mathbf{A}_H \cdot \mathbf{s})/\hat{q}\|_\infty \leq \psi \cdot \beta, \\ &(\mathbf{s} || (\mathbf{b} - \mathbf{A}^\top \mathbf{s}) || \mathbf{e}_H) \in \{0, 1\}^{(m+n+n')d} \end{aligned} \right\} \quad (3.4)$$

where ψ is the relaxation factor, and $\beta = \sqrt{n'd} \cdot (dm\hat{q}v + \hat{q}/\hat{p} - 1 + \hat{q}/\hat{p} \cdot \hat{p})/\hat{q}$.

As discussed in Section 2, we prove the rounding relation $\mathbf{v}_H = \lfloor \mathbf{A}_H \cdot \mathbf{s} \bmod \hat{q} \rfloor_{\hat{p}}$ by showing $\|(\hat{q}/\hat{p} \cdot \mathbf{v}_H + \mathbf{e}_H - \mathbf{A}_H \cdot \mathbf{s})/\hat{q}\|_\infty \leq \psi \cdot \beta$ via 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 and defer their proofs to Appendix C.2.

Lemma 3.13 (Event-Oriented Linkability in Tag Scheme). *Suppose LNP22 is knowledge sound. Then our tag scheme in Construction 3.12 is event-oriented linkable if $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard.*

Lemma 3.14 (Multi-Tag-Anonymity in Tag Scheme). *Our tag scheme in Construction 3.12 is multi-tag-anonymous if $\text{MLWR}_{\mathcal{R}, n, n', \hat{q}, \hat{p}, \chi}$ assumption for $\chi = \mathcal{U}(\mathbb{S}_v)$ is hard.*

Lemma 3.15 (Non-Frameability in Tag Scheme). *Suppose LNP22 is knowledge sound. Then our tag scheme in Construction 3.12 is non-frameable if $\text{MLWE}_{\mathcal{R}, n, m, q, \tilde{\chi}}$ assumption for $\tilde{\chi} := \mathcal{U}(\mathbb{S}_v)$ and $\text{MSIS}_{\mathcal{R}, n', (m+n'), \hat{q}, \hat{p}}$ assumption for $\beta = \max(\hat{q}/\hat{p} - 1, v)$ are hard.*

4 Lether: Account-Based Private Blockchain Payments from Lattices

In this section, we provide the full details of Lether. Due to space constraints, we defer the formal definitions of Lether to Appendix D. 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 3.12, and AmtGen is instantiated by mmEnc of the Ref-AH mmPKE in Construction 3.2.

The remaining algorithms are straightforward. The Register algorithm is from LNP22.V. The RollOver algorithm consists of mmExt and the additively-homomorphic evaluation of Ref-AH mmPKE from Construction 3.2. The LinkTag algorithm is from the tag scheme in Construction 3.12. 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 3.9, which can then be publicly verified by the system.

As in other account-based private blockchain payment systems, including (Anonymous) Zether [7, 11] and Pride CT [24], we set $k := 32$ as the bit-length of the maximum value supported in the payment system, i.e., $\text{MAX} = 2^{32} - 1$. As noted in Zether [7], a larger range of values can be represented 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 supports values larger than MAX.

Algorithm 3 Setup(1^λ) λ is the security parameter

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

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

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

$$R_{\text{pk}} = \left\{ (s || \mathbf{b} - \mathbf{A}^\top \mathbf{s}) \in \{0, 1\}^{(m+n)d} \right\}. \quad (4.1)$$

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

Algorithm 5 generates an anonymous transaction. In the anonymous transfer, the spender uses an anonymity set to hide the identity of herself and the recipient. Thus, each account in the anonymity set should be treated as the spender and recipient. We

begin by defining the proof relation R_{an} along with its statement and witness as follows,

$$R_{\text{an}} = \left\{ \begin{array}{l} \text{stat} = (\mathbf{A}, \mathbf{A}_H, (\mathbf{b}_i, \mathbf{u}_i, v_i)_{i \in [N]}, (\mathbf{c}, (c_i)_{i \in [N]}), \mathbf{v}_H); \\ \text{wit} = (s_s, \mathbf{r}, \mathbf{b}^{(s)}, \mathbf{b}^{(r)}, \hat{m}, \mathbf{b}_{m'}, bt_{m'}, \mathbf{e}_H, \mathbf{b}_h); \\ \mathbf{b}^{(s)}, \mathbf{b}^{(r)} \in \{0, 1\}^N \subseteq \mathbb{Z}_q^N, \sum_{i \in [N]} b_i^{(s)} = 1, \sum_{i \in [N]} b_i^{(r)} = 1, \\ \text{bin} \in \{0, 1\}^{(m+n+2n'+o+o')d}, \\ \left\| \begin{array}{l} \mathbf{r} \\ \mathbf{c} - \mathbf{A}\mathbf{r} \end{array} \right\|_\infty \leq \psi \cdot \beta_0, \\ \left\| \begin{array}{l} c_1 - \langle \mathbf{b}_1, \mathbf{r} \rangle - \lfloor \frac{q}{2T+2} \rfloor (b_1^{(r)} - b_1^{(s)}) \hat{m} \\ \vdots \\ c_N - \langle \mathbf{b}_N, \mathbf{r} \rangle - \lfloor \frac{q}{2T+2} \rfloor (b_N^{(r)} - b_N^{(s)}) \hat{m} \\ (2 \cdot \mathbf{v}_H + \mathbf{e}_H - \mathbf{A}_H \cdot \mathbf{s}_{\hat{0}}) / \hat{q} \end{array} \right\|_\infty \leq \psi \cdot \beta_1, \\ \langle \vec{0}^{d-k} || \vec{2}^k, \hat{m}' - \vec{T}_d \rangle = \langle \vec{0}^{d-k} || \vec{2}^k, \hat{b}_{t_{m'}} \rangle, \\ \sum_{i \in [N]} b_i^{(s)} (v_i + c_i - \langle \mathbf{u}_i + \mathbf{c}, \mathbf{s}_{\hat{0}} \rangle) - \lfloor q/(2T+2) \rfloor \cdot \hat{m}' = h \end{array} \right\}. \quad (4.2)$$

where $h := \sum_{i \in [o]} \delta_i \cdot b_h^{(i)}$, $\hat{m}' := \sum_{i \in [o']} \delta'_i \cdot b_{m'}^{(i)}$ are the same as in Equation (3.3) and

$$\text{bin} := (s_s || \sum_{i \in [N]} b_i^{(s)} \mathbf{b}_i - \mathbf{A}^\top \mathbf{s}_{\hat{0}} || \hat{m} || bt_{m'} || \mathbf{e}_H || \mathbf{b}_{m'} || \mathbf{b}_h).$$

The statement includes the public parameters $\mathbf{A}, \mathbf{A}_H := \mathcal{H}(\text{Lether} || H)$ for the current epoch H , the anonymity 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 s_s of the spender;
- the randomness $\mathbf{r}$ used in the amount ciphertexts $(\mathbf{c}, (c_i)_{i \in [N]})$;
- the binary integer vector $\mathbf{b}^{(s)}, \mathbf{b}^{(r)}$ indicating the indices of the spender and recipient, respectively, where only $b_s^{(s)} = 1, b_r^{(r)} = 1$, and all other elements are 0;
- a binary polynomial $\hat{m} \leftarrow \text{BinPoly}(m)$ whose coefficients represent the binary decomposition of the integer amount m ;
- a polynomial vector $\mathbf{b}_{m'} \leftarrow \text{Bin}(\hat{m}')$ representing the coefficient-wise binary decomposition of the polynomial-encoded balance $\hat{m}' \in \mathcal{R}_{2T+2}$ in the spender's account *after* the transaction;
- a binary polynomial $bt_{m'} \leftarrow \text{BinPoly}(m')$ whose coefficients are the binary decomposition of the integer-encoded balance $m' := \langle \vec{0}^{d-k} || \vec{2}^k, \hat{m}' \rangle$ in the spender's account *after* the transaction;
- the rounding error $\mathbf{e}_H$ related to the linkable tag $\mathbf{v}_H$;
- a polynomial vector $\mathbf{b}_h \leftarrow \text{Bin}(h)$ which is the coefficient-wise binary decomposition of the decryption error h during the decryption of the spender's account *after* the transaction.

The proof relation consists of four parts, detailed below.

First, to guarantee the uniqueness of spender and recipient, we must ensure that both $\mathbf{b}^{(s)}$ and $\mathbf{b}^{(r)}$ are binary *integer* vectors whose entries sum to 1. We use $\mathbf{b}^{(s)}$ as an example: we first show that $\mathbf{b}^{(s)} \in \{0, 1\}^N \subseteq \mathbb{Z}_q^N$, and then prove that $\sum_{i \in [N]} b_i^{(s)} = 1$.

Second, we prove that a concatenation of several vectors, specifically, $\mathbf{b}_{m'}, \mathbf{b}_h, \hat{m}, bt_{m'}, \mathbf{e}_H, s_s$, and $\mathbf{e}_s := (\mathbf{b}_s - \mathbf{A}^\top \mathbf{s}_s)$, forms a polynomial vector with binary coefficients. This not only proves knowledge of the spender's private key $(s_s, \mathbf{e}_s)$, but also guarantees that the transaction amount $\hat{m}$, the spender's balance $(\mathbf{b}_{m'}, bt_{m'})$, the decryption error $\mathbf{b}_h$, and the rounding error in the linkable tag

$\mathbf{e}_H$ each lie within a suitable range. Specifically, we choose parameters such that $\hat{q}/\hat{p} = 2$, $v = 1$, and $\bar{v} = 2$, which guarantees that the rounding 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 batch the proofs of the well-formedness of the multi-recipient ciphertext in Equation (3.1) and the linkable tag in Equation (3.4) to improve efficiency. This ensures that the decrease in the spender's amount equals the increase in the recipient's amount, while the amounts of all decoy accounts are zero. Specifically, we leverage the binary strings $\mathbf{b}^{(s)}$ and $\mathbf{b}^{(r)}$ to implicitly encode the messages in the ciphertexts as $\hat{m}_i := (b_i^{(r)} - b_i^{(s)}) \cdot \hat{m}$. For decoy accounts, $b_i^{(s)} = b_i^{(r)} = 0$, so $\hat{m}_i = 0$; for the spender, $b_s^{(s)} = 1$, $b_r^{(r)} = 0$, thus $\hat{m}_s = -\hat{m}$; and for the recipient, $\hat{m}_r = \hat{m}$. Our protocol also supports the special case where the spender and recipient are the same one, i.e., $s = r$, in which case $\hat{m}_s = \hat{m}_r = 0$, 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 $\beta = \sqrt{n'd} \cdot (dm\hat{q}v + \hat{q}/\hat{p} - 1 + \hat{q}/\hat{p} \cdot \hat{p})/\hat{q}$ to accommodate the batched proof.

Fourth, we use verifiable decryption to show that the decrypted balance of the spender's account after the transaction is non-negative (i.e., the spender is not *overdrawn*). Since the message space for balances is shifted from $\{-T, \dots, T+1\}$ to $\{0, \dots, 2T+1\}$ by adding the constant offset T_d , we need to shift it back by subtracting T_d during the range proof.

Algorithm 4 AddrGen()

```

1: (pk := b, sk := s) ← mmKGen()
2:  $\pi \leftarrow \text{LNP22.P}(R_{\text{pk}}, (\mathbf{A}, \mathbf{b}), \mathbf{s})$  where  $R_{\text{pk}}$  is defined in Equation (4.1)
3: return (reg := (pk,  $\pi$ ), sk)

```

Algorithm 5 AnTransfer

Input: $(\text{pk}_i = \mathbf{b}_i)_{i \in [N]}$, s, r , $\text{sk}_s = \mathbf{s}_s$, $\text{amt} = m, \mathbb{S}$

```

1:  $H \leftarrow \mathbb{S}$ 
2:  $\mathcal{H} \leftarrow \text{pp}$ 
3:  $\mathbf{A}_H := \mathcal{H}(\text{Lether}||H)$ 
4:  $\mathbf{v}_H := \lfloor \mathbf{A}_H \cdot \mathbf{s}_s \bmod \hat{q} \rfloor \hat{p}$ 
5:  $\mathbf{e}_H := \mathbf{A}_H \cdot \mathbf{s} - \hat{q}/p \cdot \mathbf{v}_H \bmod \hat{q}$ 
6:  $(\mathbf{r}, \mathbf{e}_u) \leftarrow \mathcal{D}_{\sigma_0}^n \times \mathcal{D}_{\sigma_0}^m$ 
7:  $(y_i)_{i \in [N]} \leftarrow \mathcal{D}_{\sigma_1}^N$ 
8:  $\hat{m} \leftarrow \text{BinPoly}(m)$ 
9:  $\hat{m}_s := -\hat{m}$ ,  $\hat{m}_r := \hat{m}$ ,  $\hat{m}_i := 0$  for  $i \in [N] \setminus \{s, r\}$ 
10:  $(\mathbf{c}, (\mathbf{c}_i)_{i \in [N]}) \leftarrow \text{mmEnc}(\mathbf{A}, (\mathbf{b}_i)_{i \in [N]}, (\hat{m}_i)_{i \in [N]}, (\mathbf{r}, \mathbf{e}_u), (y_i)_{i \in [N]})$ 
11:  $\mathbf{b}^{(s)} := (b_1^{(s)}, \dots, b_N^{(s)}) \in \{0, 1\}^N$ , where  $b_s^{(s)} := 1$ ,  $b_i^{(s)} := 0$  for  $i \in [N] \setminus \{s\}$ 
12:  $\mathbf{b}^{(r)} := (b_1^{(r)}, \dots, b_N^{(r)}) \in \{0, 1\}^N$ , where  $b_r^{(r)} := 1$ ,  $b_i^{(r)} := 0$  for  $i \in [N] \setminus \{r\}$ 
13:  $\text{acc} \leftarrow \mathbb{S}$ 
14: for  $i \in [N]$ , do  $(\mathbf{u}_i, v_i) \leftarrow \text{acc}[\text{pk}_i]$ 
15:  $(\mathbf{u}'_s, v'_s) := (\mathbf{u}_s + \mathbf{c}, v_s + c_s)$ 
16:  $\hat{m}' := \lfloor v'_s - \langle \mathbf{u}'_s, \mathbf{s}_s \rangle \rfloor_{2T+2}$ 
17:  $h := v'_s - \langle \mathbf{u}'_s, \mathbf{s}_s \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m}'$ 
18:  $\mathbf{b}_{m'} \leftarrow \text{Bit}(\hat{m}')$ 
19:  $\mathbf{b}_h \leftarrow \text{Bit}(h)$ 
20:  $m' := \langle \vec{0}^{d-k} || \vec{2}^k, \vec{m}' - \vec{T}_d \rangle$ 
21:  $\mathbf{b}_{m'} \leftarrow \text{BinPoly}(m')$ 
22:  $\Pi \leftarrow \text{LNP22.P}(R_{\text{an}}, \text{stat}, \text{wit})$  where  $R_{\text{an}}$  is defined in Equation (4.2)
23: return tx =  $((\text{pk}_i := \mathbf{b}_i)_{i \in [N]}, \mathbf{ct} := (\mathbf{c}, (\mathbf{c}_i)_{i \in [N]}), \text{tag} := \mathbf{v}_H, \Pi)$ 

```

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  $((\text{pk}_i = \mathbf{b}_i)_{i \in [N]}, \mathbf{ct} = (\mathbf{c}, (\mathbf{c}_i)_{i \in [N]}), \text{tag} = \mathbf{v}_H, \Pi) \leftarrow \text{tx}$ 
2:  $H \leftarrow \mathbb{S}$ 
3:  $\mathcal{H} \leftarrow \text{pp}$ 
4:  $\mathbf{A}_H := \mathcal{H}(\text{Lether}||H)$ 
5:  $\text{acc} \leftarrow \mathbb{S}$ 
6: for  $i \in [N]$ ,  $(\mathbf{u}_i, v_i) \leftarrow \text{acc}[\text{pk}_i]$ 
7: return 0/1  $\leftarrow \text{LNP22.V}(R_{\text{an}}, \text{stat}, \Pi)$  where  $R_{\text{an}}$  is defined in Equ. (4.2)

```

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

Denial of Service. We further consider 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. We note that such an attack reduces to the non-frameability property of the tag scheme.

4.1 Parameter Setting

We first present the parameter setting for Lether. At a high level, the parameters must satisfy not only the requirements of the verifiable Ref-AH mmPKE and the tag scheme (as discussed in Section 3), but also the conditions required by LNP22 when these components are combined. We summarize the requirements as follows:

- The Ref-AH mmPKE scheme must be $\text{mmIND-CPA}^{\text{KOSK}}$ secure, support T -level additive homomorphism, and achieve ζ -correctness for $\zeta \leq 2^{-128}$
- Verifiable multi-encryption, verifiable decryption, and verifiable refresh must satisfy completeness, soundness (or unforgeability), and simulatability.
- The tag scheme must satisfy event-oriented linkability, multi-tag anonymity, and non-frameability.
- LNP22 must satisfy completeness, knowledge soundness and simulatability.

Following prior work [12, 13, 15, 31], we use the Lattice Estimator (a.k.a. LWE Estimator [1]) to estimate the practical hardness of the lattice problems appearing in our Ref-AH mmPKE and tag scheme. These include $\text{MLWE}_{\mathcal{R}, n, m, q, \chi}$ for $\chi := \mathcal{U}(\mathbb{S}_v)$, $\text{MatrixHint-MLWE}_{\mathcal{R}, m+N, n, q, \chi_0}^{N, \chi_1, S}$, $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$, $\text{MLWR}_{\mathcal{R}, n, n', \hat{q}, \hat{p}, \chi}$ for $\chi = \mathcal{U}(\mathbb{S}_v)$, and $\text{MSIS}_{\mathcal{R}, n', (m+n'), \hat{q}, \hat{p}}$ for $\beta = \max(\hat{q}/\hat{p} - 1, v)$, all targeting the 128-bit security level.

In addition, following the parameter selection strategy of the basic mmPKE [44] and LNP22 [31, 35], we choose parameters to guarantee sufficient correctness of Ref-AH mmPKE, as well as completeness, simulatability, and soundness of LNP22 as used in Lether. We also carefully select parameters to minimize the amortized cost of refresh operations. We defer the detailed parameter selection to Appendix A.1 and summarize the concrete parameters below.

Specifically, we set $k := 32$ for Lether. For Ref-AH mmPKE, we choose $q \approx 2^{46}$, $d = 64$, $N = 16$, $m = 30$, $n = 26$, $T = 60$, $v = 1$, $\bar{v} = 2$, $\sigma_0 = 15.9$, and $\sigma_1 = 30560$. For the tag scheme, we set $d = 64$, $\hat{q} \approx 2^{11}$, $\hat{p} \approx 2^{10}$ with $\hat{q}/\hat{p} = 2$, and $n' = 2$. Note that the modulus q and degree d are also used in LNP22 to maintain compatibility across components.

4.2 Implementation and Evaluation

We implement Lether in C on a standard desktop machine equipped with an Intel i7-11850H CPU running at 2.50 GHz. Our implementation is based on the LaZer library [35].

Although the LaZer library provides a user-friendly Python interface, it only supports proving simple linear relations in a fixed manner, and the moduli used by the NIZK system and the target relations do not match. Therefore, as discussed in Section 1.3 and Section 2, to make Lether practical and efficiently prove the required relations, we optimize the LNP22 implementation in the LaZer library to support: (i) using the same modulus as the proof relations; (ii) batching proofs (e.g., verifiable multi-encryption and tag proofs); (iii) aggregating integer proofs (e.g., those arising in ring signatures); and (iv) proving quadratic relations (e.g., those arising in balance proofs). To realize these optimizations, we extensively restructure internal functions and parameter generation scripts. The newly added or modified code exceeds 3,000 lines. Implementation details are deferred to Appendix A.2.

Results for typical Lether transactions appear in Table 1. Figure 1 and Figure 2 show the communication size of transactions $|tx|$ and the associated proof size $|\Pi|$, together with the proving time t_p and verification time t_v , for anonymity set sizes N ranging from 2 to 64. Each runtime value is averaged over 100 runs. We observe that all costs scale linearly with the anonymity set size N , which is consistent with (Anonymous) Zether [7, 11] and Pride CT [24]. These results demonstrate that Lether remains efficient for $N \leq 64$.

Table 5 reports the results of the registration phase in Lether. The communication cost for each user during registration is about 37 KB, and the computation cost is only a few milliseconds. Moreover, these costs are *one-time* for each user.

Anony.	Eval. Level	Register			
		$ reg $	$ \pi $	t_p	t_v
N	T				
16	60	37.2	27.5	0.12	0.08

Table 5: Results of the Lether registration phase. Sizes are reported in KB and times in seconds. We report the communication size $|reg|$ (including proofs $|\pi|$), the proving time t_p , and the verification time t_v . The anonymity set size is $N := 16$ (Anony.), and the evaluation level is $T := 60$ (Eval. Level).

Acknowledgments

This work was partially supported by an Australian Research Council Discovery Grants DP250100229 and DP260100245. Siu-Ming Yiu was supported by Shenzhen-Hong Kong-Macao Science and Technology Plan Project (Category C Project: SGDX20210823103537030), and the Theme-based Research Scheme of the Research Grants Council of Hong Kong (T35-710/20-R). We thank the anonymous reviewers for their helpful comments.

Open Science

Artifact URL: <https://github.com/dingdangdangwhx/Lether-artifact>.

We provide self-contained C implementations of the Lether payment system. The artifact also contains scripts for generating the

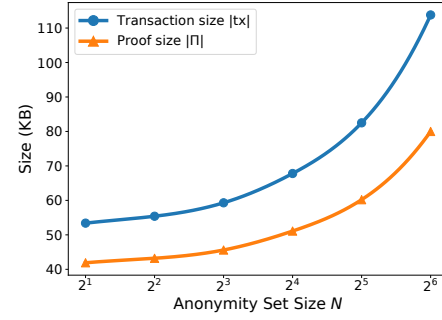


Figure 1: Transaction and proof size growth with anonymity set size. The evaluation level is fixed at $T := 60$.

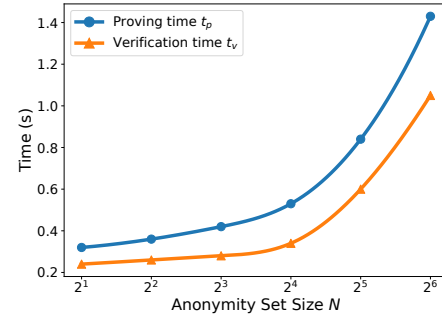


Figure 2: Proving and verification time growth with anonymity set size. The evaluation level is fixed at $T := 60$.

header files used by the LaZer library and for generating the witness-statement pairs for transactions.

References

- [1] Martin R Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology* 9, 3 (2015), 169–203.
- [2] Diego F Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerdand Silde. 2023. 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*. 1467–1481.
- [3] Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. 2013. Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theoretical Computer Science* 469 (2013), 1–14.
- [4] Elaine Barker, Lily Chen, Sharon Keller, Allen Roginsky, Apostol Vassilev, and Richard Davis. 2017. *Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography*. Technical Report. National Institute of Standards and Technology.
- [5] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. 2002. Randomness Re-use in Multi-recipient Encryption Schemes. In *Public Key Cryptography – PKC 2003*, Yvo G. Desmedt (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 85–99.
- [6] Jonathan Bootle, Kaoutar Elkhayoui, Julia Hesse, and Yacov Manevich. 2022. DualDory: logarithmic-verifier linkable ring signatures through preprocessing. In *European Symposium on Research in Computer Security*. Springer, 427–446.
- [7] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. 2020. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*. Springer, 423–443.
- [8] Melissa Chase and Anna Lysyanskaya. 2006. 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, 78–96.
- [9] Rafael del Pino and Shuichi Katsumata. 2022. A New Framework for More Efficient Round-Optimal Lattice-Based (Partially) Blind Signature via Trapdoor Sampling. In *Advances in Cryptology – CRYPTO 2022*, Yevgeniy Dodis and Thomas

- Shrimpton (Eds.). Springer Nature Switzerland, Cham, 306–336.
- [10] Michel H Devoret and Robert J Schoelkopf. 2013. Superconducting circuits for quantum information: an outlook. *Science* 339, 6124 (2013), 1169–1174.
 - [11] Benjamin E Diamond. 2021. Many-out-of-many proofs and applications to anonymous zether. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1800–1817.
 - [12] Muhammed F Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. 2023. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. In *Annual International Cryptology Conference*. Springer, 484–517.
 - [13] Muhammed F Esgin, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. 2019. Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications. In *Annual International Cryptology Conference*. Springer, 115–146.
 - [14] Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. 2022. Efficient verifiable partially-decryptable commitments from lattices and applications. In *IACR International Conference on Public-Key Cryptography*. Springer, 317–348.
 - [15] Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. 2022. MatRiCT+: More efficient post-quantum private blockchain payments. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1281–1298.
 - [16] Muhammed F Esgin, Raymond K Zhao, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. 2019. MatRiCT: efficient, scalable and post-quantum blockchain confidential transactions protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 567–584.
 - [17] Thomas Espitau, Guilhem Niot, and Thomas Prest. 2024. Flood and Submerse: Distributed Key Generation and Robust Threshold Signature from Lattices. In *Advances in Cryptology – CRYPTO 2024*, Leonid Reyzin and Douglas Stebila (Eds.). Springer Nature Switzerland, Cham, 425–458.
 - [18] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. 2019. 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, 649–678.
 - [19] Amos Fiat and Adi Shamir. 1986. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*. Springer, 186–194.
 - [20] Ethereum Foundation. 2025. *Introducing ZKnox*. <https://x.com/ethereumfndn/status/1896592240228893072> Posted on X (formerly Twitter), March 26, 2025.
 - [21] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 169–178.
 - [22] Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter Rønne, and Tjerand Silde. 2022. Verifiable decryption in the head. In *Australasian Conference on Information Security and Privacy*. Springer, 355–374.
 - [23] Jens Groth, Rafail Ostrovsky, and Amit Sahai. 2006. Perfect Non-interactive Zero Knowledge for NP. In *Advances in Cryptology - EUROCRYPT 2006*, Serge Vaudenay (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 339–358.
 - [24] Yue Guo, Harish Karthikeyan, Antigoni Polychroniadou, and Chaddy Huussin. 2024. Pride ct: Towards public consensus, private transactions, and forward secrecy in decentralized payments. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3904–3922.
 - [25] Abida Haque, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. 2022. Logarithmic-size (linkable) threshold ring signatures in the plain model. In *IACR International Conference on Public-Key Cryptography*. Springer, 437–467.
 - [26] Kaoru Kurosawa. 2002. Multi-recipient Public-Key Encryption with Shortened Ciphertext. In *Public Key Cryptography*, David Naccache and Pascal Paillier (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 48–63.
 - [27] Russell WF Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. 2019. Omniring: Scaling private payments without trusted setup. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 31–48.
 - [28] Zeyu Liu, Katerina Sotiraki, Eran Tromer, and Yunhao Wang. 2025. Lattice-based Multi-message Multi-recipient KEM/PKE with Malicious Security. *Cryptology ePrint Archive* (2025).
 - [29] Vadim Lyubashevsky. 2012. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 738–755.
 - [30] Vadim Lyubashevsky and Gregory Neven. 2017. 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, 293–323.
 - [31] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. 2022. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In *Annual International Cryptology Conference*. Springer, 71–101.
 - [32] Vadim Lyubashevsky, Ngoc Khanh Nguyen, Maxime Plançon, and Gregor Seiler. 2021. 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, 218–248.
 - [33] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. 2021. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *IACR International Conference on Public-Key Cryptography*. Springer, 215–241.
 - [34] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. 2021. SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions. In *Annual International Cryptology Conference*. Springer, 611–640.
 - [35] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. 2024. The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. 3125–3137.
 - [36] Varun Madathil and Alessandra Scauro. 2025. Pri FHE te: Achieving Full-Privacy in Account-Based Cryptocurrencies is Possible. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 199–229.
 - [37] Shen Noether, Adam Mackenzie, et al. 2016. Ring confidential transactions. *Ledger* 1 (2016), 1–18. Source code available at: <https://github.com/monero-project/monero> (accessed on branch release-v0.18).
 - [38] Amirreza Sarencheh, Hamidreza Khoshakhlagh, Alireza Kavousi, and Aggelos Kiayias. 2025. DART: Decentralized, Anonymous, and Regulation-friendly Tokenization. *Cryptology ePrint Archive* (2025).
 - [39] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*. IEEE, 459–474.
 - [40] Tjerand Silde. 2022. Short Paper: Verifiable Decryption for BGV. In *International Conference on Financial Cryptography and Data Security*. Springer, 381–390.
 - [41] Shi-Feng Sun, Man Ho Au, Joseph K Liu, and Tsz Hon Yuen. 2017. 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, 456–474.
 - [42] Yongli Tang, Deng Pan, Qing Ye, Yuanhong Li, and Jinxia Yu. 2024. Event-oriented linkable group signature from lattice. *IEEE Transactions on Consumer Electronics* (2024).
 - [43] Patrick P Tsang, Victor K Wei, Tony K Chan, Man Ho Au, Joseph K Liu, and Duncan S Wong. 2004. Separable linkable threshold ring signatures. In *International Conference on Cryptology in India*. Springer, 384–398.
 - [44] Hongxiao Wang, Ron Steinfeld, Markku-Juhani O. Saarinen, Muhammed F. Esgin, and Siu-Ming Yiu. 2026. mmCipher: Batching Post-Quantum Public Key Encryption Made Bandwidth-Optimal. In *35th USENIX Security Symposium, USENIX Security 2026, Baltimore, MD, USA, August 12–14, 2026*. USENIX Association, (to appear). <https://eprint.iacr.org/2025/1000>
 - [45] Nan Wang, Qianhui Wang, Dongxi Liu, Muhammed F. Esgin, and Alsharif Abuadba. 2025. BulletCT: towards more scalable ring confidential transactions under transparent setup. In *Proceedings of the 34th USENIX Conference on Security Symposium (Seattle, WA, USA) (USENIX Security 2025)*. USENIX Association, USA, Article 172, 20 pages.
 - [46] Yuxi Xue, Xingye Lu, Man Ho Au, and Chengru Zhang. 2024. Efficient linkable ring signatures: new framework and post-quantum instantiations. In *European symposium on research in computer security*. Springer, 435–456.
 - [47] Tsz Hon Yuen, Muhammed F Esgin, Joseph K Liu, Man Ho Au, and Zhimin Ding. 2021. DualRing: generic construction of ring signatures with efficient instantiations. In *Annual International Cryptology Conference*. Springer, 251–281.
 - [48] Tsz Hon Yuen, Shi-feng Sun, Joseph K Liu, Man Ho Au, Muhammed F Esgin, Qingzhao Zhang, and Dawu Gu. 2020. 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, 464–483.
 - [49] Tianyu Zheng, Shang Gao, Yubo Song, and Bin Xiao. 2023. Leaking Arbitrarily Many Secrets: Any-out-of-Many Proofs and Applications to RingCT Protocols. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2533–2550.

A Detailed Parameter Settings and Implementations

In this section, we detail the parameter settings and implementation aspects of Lether.

A.1 Detailed Parameter Setting

We demonstrate how to choose the parameters for the Lether system, including the verifiable Ref-AH mmpKE, the event-oriented linkable tag scheme, and LNP22, as follows.

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

- The Ref-AH mmPKE scheme must be $\text{mmIND-CPA}^{\text{KOSK}}$ secure, support T -level additive homomorphism, and achieve ζ -correctness for $\zeta \leq 2^{-128}$.
- 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.

Following a similar strategy to [44], we describe the parameter selection step by step below.

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

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

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

where $\|(\mathbf{r}, \mathbf{e}_u)\|_\infty \leq \psi \cdot \sqrt{(m+n)d} \cdot \tau\sigma_0$ and $\|y_i\|_\infty \leq \psi \cdot \sqrt{Nd} \cdot \tau\sigma_1$. Therefore, 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 $\lfloor q/(4T+4) \rfloor$. That is, the following inequality should hold with overwhelming probability:

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

Third, we fix the parameters as $d = 64$, $N = 16$, and $\psi = 189$ (see Appendix B.3). Following Lemma B.1, we fix $\tau = 1.6$ to bound the Gaussian tail. To ensure the security of our Ref-AH mmPKE and its verifiability, we use the parameter-generation script from [44] to instantiate the hardness assumptions $\text{MLWE}_{\mathcal{R}, n, m, q, \bar{\chi}}$ for $\bar{\chi} := \mathcal{U}(\mathbb{S}_v)$, $\text{MatrixHint-MLWE}_{\mathcal{R}, m+N, n, q, \chi_0}^{N, \chi_1, S}$, and $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ at the 128-bit security level. This yields the concrete parameters $\sigma_0 = 15.9$, $\sigma_1 = 30560$, $q \approx 2^{46}$, $m = 30$, and $n = 26$ when $T = 60$.

Finally, we verify that the above parameters also satisfy the soundness conditions of LNP22 when the modulus of LNP22 matches that of our Ref-AH mmPKE, as detailed later.

Plug-and-Play Event-Oriented Linkable Tag. To parameterize our tag scheme, in addition to the requirements already specified for our verifiable Ref-AH mmPKE, we require the following additional assumptions to hold:

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

We briefly outline our parameter selection as follows.

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

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 work [12, 13, 15, 31], we use the Lattice Estimator (a.k.a. LWE Estimator [1]) to estimate the practical hardness of MLWR and MSIS at the 128-bit security level, which yields $n' := 2$.

In what follows, we show how to choose the parameters for LNP22 that enable the integration of the tag proof and the verifiable Ref-AH mmPKE into the transaction proof.

LNP22 in Lether. We further summarize the LNP22 requirements as follows:

- **Switched modulus condition:** As discussed in Section 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} (Equation (4.2)). This leads to the requirement that

$$q \geq (\hat{q}/\hat{p} \cdot \hat{p} + md \cdot \hat{q}v + \hat{q}/\hat{p} - 1) + \hat{q} \cdot \beta_1.$$

- **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 [31, 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 $q \geq 2^k \cdot (2T + 1)$.

After verifying that the above requirements are satisfied, we obtain all parameters for LNP22 in Lether using our modified parameter selection scripts from [29, 35]. Overall, fixing $N := 16$ and $T := 60$, the resulting sizes are as follows.

For Ref-AH mmPKE, we have $|\mathbf{b}_i| = 9.34$ KB, $|\mathbf{c}| = 10.78$ KB, $|c_i| = 0.36$ KB, and $|\mathbf{ct}| = 16.53$ KB. For the tag scheme, the tag size is $|\mathbf{v}_H| = 0.16$ KB. For LNP22, the proof size for R_{pk} (Equation (4.1)) in the registration phase is about $|\pi| = 27.5$ KB, the proof size for R_{ref} (Equation (3.3)) in the refresh phase is about $|\pi'| = 43.8$ KB, and the proof size for R_{an} (Equation (4.2)) in the transaction phase is about $|\Pi| = 51.1$ KB.

The total communication cost of a *verifiable refresh* is $|\text{ref}| = |\pi'| + |\mathbf{c}| + |c_i| = 54.9$ KB. The total communication cost of *registration* is $|\text{reg}| = |\pi| + |\mathbf{b}_i| = 37.2$ KB, which is a one-time cost for each user. The total communication cost of a *transaction* is $|\text{tx}| = |\Pi| + |\mathbf{ct}| + |\mathbf{v}_H| = 67.8$ KB.

Following the above strategies, we further present the results of Lether under different homomorphic evaluation levels $T \in \{1, \dots, 1000\}$ in Table 6. We observe that as T increases, both the transaction and refresh costs increase accordingly, mainly due to larger parameter sizes required by Ref-AH mmPKE and LNP22. However, the amortized cost is minimized when $T = 60$, which is why we set $T := 60$ in our implementation.

A.2 Detailed Implementation

We first recall the workflow of the implementations of LNP22 in the LaZer library [35]. The user first executes the *code-generation* scripts

Eval. Level	Transaction		Refresh		Amortization
	tx	Π	ref	π'	
T					tx + ref /T
1	62.3	48.2	49.9	40.7	112.2
20	66.7	50.4	53.5	42.9	69.3
40	67.8	51.1	54.9	43.8	69.0
60	67.8	51.1	54.9	43.8	68.7
80	68.7	51.8	55.7	44.3	69.4
100	70.7	53.1	57.6	45.6	71.3
1000	80.1	58.1	65.3	50.3	80.2

Table 6: Additional Lether results for different evaluation levels $T \in \{1, 20, 40, 60, 80, 100, 1000\}$ (Eval. Level) for additively homomorphic operations on fresh or refreshed accounts. Sizes are in KB. For the Transaction and Refresh phases, we report the communication sizes (|tx|, |ref|), including the associated proofs (| Π |, | π' |), as well as the amortized communication per level, computed as |tx| + |ref|/T. The anonymity set size is fixed at $N := 16$.

to obtain a *header file* containing the parameters required by the LaZer library, given a description of the proving relation. Next, the user develops a *front-end* program (in C or Python) that generates or loads the witness and statement, and invokes the generic proving and verification functions provided by the LaZer library to produce or verify a proof. When these functions are invoked by the front-end, the LaZer library calls the LNP22 C implementation with the parameters specified in the header file, together with the statement and witness supplied by the front-end.

To implement the Lether scheme, we first modify the original *code-generation* scripts. Specifically, we fix the modulus q of LNP22 to match that of the statement (i.e., that of Ref-AH mmPKE), and predefine $\lceil \log q \rceil$ and the statement dimensions m, n (as discussed above and evaluated for security and soundness). Following the standard LaZer workflow, we then execute the modified code-generation scripts to determine the concrete modulus q and other parameters required by LNP22 to generate the header file, where the LWE estimator [1] is invoked as a subroutine to ensure 128-bit security.

With the resulting modulus q , we use SageMath scripts to generate the witness and statement, and employ a front-end C program to load them and invoke the generic proving and verification functions. Since the original LNP22 implementation in the LaZer library supports only linear relations, we reconstruct its C library to extend its functionality. Specifically, we introduce three ARPs (for binary proofs, verifiable multi-encryption, and other relations, respectively, as described above) and implement both quadratic and linear relations over $\mathbb{Z}_q$ and $\mathcal{R}_q$ to support verifiable decryption, balance proofs, sign-bit proofs, integer proofs, and Hamming-weight proofs for spender–recipient indicator bit strings.

We now describe how to aggregate the integer proofs for the spender–recipient indicator bit strings. Specifically, we set the number of repetitions to $M := \lceil \lambda / \log q \rceil$ to ensure that the soundness error of the aggregation is negligible (i.e., $(1/q)^M \leq 2^{-\lambda}$). We then aggregate the spender–recipient indicator bit strings $\mathbf{b}^{(s)} = \{b_i^{(s)}\}_{i \in [N]}$ and $\mathbf{b}^{(r)} = \{b_i^{(r)}\}_{i \in [N]}$ using integer challenges $\{\gamma_{i,u} \in$

$\mathbb{Z}_q\}_{i \in [2N], u \in [M]}$, derived via a hash function following the Fiat-Shamir transform [19], into M aggregated integers $\{b_u\}_{u \in [M]}$, defined as

$$b_u := \sum_{i=0}^{N-1} \gamma_{i,u} \cdot b_i^{(s)} + \sum_{i=0}^{N-1} \gamma_{i+N,u} \cdot b_i^{(r)}.$$

Next, we prove that each b_u is an integer via LNP22. This optimization reduces the cost of integer proofs from $2N$ instances to a small constant M (where $2N := 32$, $M := 3$ in our implementations), reducing the runtime in practice from a few seconds to a fraction of a second. After establishing that $\mathbf{b}^{(s)}$ and $\mathbf{b}^{(r)}$ are integer vectors, we further apply binary proofs to show that $\mathbf{b}^{(s)}, \mathbf{b}^{(r)} \in \{0, 1\}^N$, and linear proofs to show that $\sum_{i \in [N]} b_i^{(s)} = \sum_{i \in [N]} b_i^{(r)} = 1$. Together, these proofs establish the well-formedness of the spender–recipient indicator vectors.

B Preliminaries

B.1 Notation

Let $\lambda \in \mathbb{N}$ denote the security parameter. For a positive integer n , we denote the set $\{1, \dots, n\}$ by $[n]$. For positive integers b, t , we denote the integer vector $(b^0, b^1, \dots, b^{t-1})$ 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 v , we write $\mathbb{S}_v$ to denote the set of polynomials in $\mathcal{R}_q$ with infinity norm bounded by v . The size of the $\mathbb{S}_v$ coefficient support is denoted $\bar{v} \leq 2v + 1$; for example $v = 1$, $\bar{v} = 2$ 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/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 $\lfloor \cdot \rfloor$, e.g., $\lfloor a \rfloor$ 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 $x \leftarrow A(y)$ 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 $\mathcal{U}(S)$. We denote $\mathcal{D}_\sigma$ as a discrete Gaussian distribution with Gaussian width σ , where $\sigma := \sqrt{2\pi} \cdot s$ corresponds to the standard deviation 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.

Here, we define some useful functions and lemmas as follows:

- $\hat{m} \in \mathcal{R}_2 \leftarrow \text{BinPoly}(m \in \mathbb{Z}_q)$: Given an integer $m \in \mathbb{Z}_q$ with $m < 2^d$, this function outputs a binary polynomial $\hat{m} \in \mathcal{R}_2$ whose coefficients represent the binary decomposition of m .
- $\mathbf{b}_h \in \mathcal{R}_2^m \leftarrow \text{Bit}(h \in \mathcal{R}_q)$: 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 $\delta = (\delta_1, \delta_2, \dots, \delta_m) = (1, 2, \dots, \beta + 1 - 2^{\lfloor \log \beta \rfloor})$.

Lemma B.1 (Gaussian tail bound [29, Lemma 4.4]). *For any $\tau > 0$, $\Pr[\|z\| > \tau \sqrt{n}; z \leftarrow \mathcal{D}_j^n] < \tau^n e^{\frac{n}{2}(1-\tau^2)}$, where $\mathcal{D}_j^n$ denotes the discrete Gaussian distribution over $\mathbb{Z}^n$ with standard deviation s and τ is Gaussian tail bound factor.*

B.2 Non-Interactive Zero Knowledge Protocol

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

Definition B.2 (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 $\hat{\Pi}$ is defined as follows.

- $\text{pp}_{\hat{\Pi}} \leftarrow \hat{\Pi}.\text{Setup}(1^\lambda)$: Input a security parameter 1^λ , it outputs a public parameter $\text{pp}_{\hat{\Pi}}$.
- $\pi \leftarrow \hat{\Pi}.\text{Prove}(\text{pp}_{\hat{\Pi}}, x, w)$: Input the public parameter $\text{pp}_{\hat{\Pi}}$, a statement x and a witness w such that $(x, w) \in R_{\mathcal{L}}$, it outputs a proof π .
- $0/1 \leftarrow \hat{\Pi}.\text{Verify}(\text{pp}_{\hat{\Pi}}, x, \pi)$: Input the public parameter $\text{pp}_{\hat{\Pi}}$, 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 [12, 23].

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

$$\Pr \left[\hat{\Pi}.\text{Verify}(\text{pp}_{\hat{\Pi}}, x, \pi) = 1 \mid \begin{array}{l} \text{pp}_{\hat{\Pi}} \leftarrow \hat{\Pi}.\text{Setup}(1^\lambda); \\ \pi \leftarrow \hat{\Pi}.\text{Prove}(\text{pp}_{\hat{\Pi}}, x, w) \end{array} \right] = 1.$$

Computational Zero Knowledge. A NIZK argument system $\hat{\Pi}$ is computational zero knowledge if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\hat{\Pi}.\text{Sim} = (\text{Sim}_0, \text{Sim}_1)$ such that the following is negligible with λ ,

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

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

$$\Pr \left[\begin{array}{l} \hat{\Pi}.\text{Verify}(\text{pp}_{\hat{\Pi}}, x, \pi) = 1 \\ \wedge (x, \tilde{w}) \notin R_{\mathcal{L}} \end{array} \mid \begin{array}{l} \text{pp}_{\hat{\Pi}} \leftarrow \hat{\Pi}.\text{Setup}(1^\lambda); \\ (x, \pi) \leftarrow \mathcal{A}(\text{pp}_{\hat{\Pi}}); \\ \tilde{w} \leftarrow \hat{\Pi}.\mathcal{E}(\text{pp}_{\hat{\Pi}}, \mathcal{A}, x, \pi) \end{array} \right] = \text{negl}(\lambda).$$

B.3 NIZK Protocol in LNP22 [31]

We recall one of the most efficient lattice-based NIZK protocols proposed in [31], denoted as 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 LNP22 as follows:

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

For clarity, we omit the public parameters of LNP22 and their relation to the witness in $R_{\mathcal{L}}$. The witness in $R_{\mathcal{L}}$ is a vector 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 -norm bounds on linear functions,
- a set Ω of approximate ℓ_2 -norm bounds with relaxation factor $\psi \approx 189\gamma$ where γ is a rejection sampling constant. As a typical choice, in this paper, we set $\gamma := 1$ and omit it in subsequent expressions for simplicity.

Notably, LNP22 can also support:

- Proving an (integer) 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 $i \in \{1, \dots, d-1\}$ 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 $\langle \vec{a}, \vec{1}^d \rangle = 0$ over $\mathbb{Z}$;
- Proving an approximate ℓ_∞ -norm bound via the corresponding ℓ_2 -norm, i.e., $\|Ds - u\|_\infty \leq \|Ds - u\|_2 \leq \psi \cdot \beta$.

We define the construction of LNP22 as follows.

Construction B.3 (LNP22 from [31]). Let λ be the security parameter. The LNP22 protocol consists of the following algorithms:

- $\text{pp}_{\text{LNP}} \leftarrow \text{LNP22}.\text{Setup}(1^\lambda)$: Given the security parameter λ , output the public parameters pp_{LNP} .
- $\pi \leftarrow \text{LNP22}.\text{P}(\text{pp}_{\text{LNP}}, R_{\mathcal{L}}, \text{stat}, \text{wit})$: Given pp_{LNP} , a proof relation $R_{\mathcal{L}}$, statement stat , and witness wit as defined in Equation (B.1), output a proof π .
- $0/1 \leftarrow \text{LNP22}.\text{V}(\text{pp}_{\text{LNP}}, R_{\mathcal{L}}, \text{stat}, \pi)$: Given pp_{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 LNP22 are analyzed in detail in [31]; we refer the reader to the original paper for further information.

B.4 Lattice Preliminaries

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

Definition B.4 (MLWE Problem). Let $m, n > 0$ be positive integers. Let χ be an error distribution over $\mathcal{R}$. The MLWE problem, denoted by $\text{MLWE}_{\mathcal{R}, m, n, q, \chi}$, asks an adversary $\mathcal{A}$ to distinguish the following two case: (1) $(A, Ar + e)$ for $A \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $r \leftarrow \chi^n$, $e \leftarrow \chi^m$, and (2) (A, u) for $A \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $u \leftarrow \mathcal{U}(\mathcal{R}_q^m)$.

Definition B.5 (Matrix Hint-MLWE Problem [17, 44]). Let $m, n, \ell > 0$ be positive integers. Let $\mathcal{S}$ be a distribution over $\mathcal{R}^{\ell \times (m+n)}$ and let χ_0, χ_1 be distributions over $\mathcal{R}^{m+n}$ and $\mathcal{R}^\ell$, respectively. The Matrix Hint-MLWE problem, denoted by $\text{MatrixHint-MLWE}_{\mathcal{R}, m, n, q, \chi_0}^{\ell, \chi_1, \mathcal{S}}$, asks an adversary $\mathcal{A}$ to distinguish between the following two cases: (1) $(A, [I_m \mid A]r, R, h)$, where $A \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $r \leftarrow \chi_0$, $R \leftarrow \mathcal{S}$,

$\mathbf{y} \leftarrow \chi_1$, and $\mathbf{h} := \mathbf{R}\mathbf{r} + \mathbf{y}$; (2) $(\mathbf{A}, \mathbf{u}, \mathbf{R}, \mathbf{h})$, where $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $\mathbf{u} \leftarrow \mathcal{U}(\mathcal{R}_q^m)$, $\mathbf{r} \leftarrow \chi_0$, $\mathbf{R} \leftarrow \mathcal{S}$, $\mathbf{y} \leftarrow \chi_1$, and $\mathbf{h} := \mathbf{R}\mathbf{r} + \mathbf{y}$.

We note that there exists a reduction from standard MLWE to Matrix Hint-MLWE with appropriate parameters.

Definition B.6 (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 $\text{MLWR}_{\mathcal{R}, m, n, q, p, \chi}$, asks an adversary $\mathcal{A}$ to distinguish the following two case: (1) $(\mathbf{A}, [\mathbf{A}\mathbf{r}]_p)$ for $\mathbf{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{m \times n})$, $\mathbf{r} \leftarrow \chi^n$, and (2) $(\mathbf{A}, [\mathbf{u}]_p)$ 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 , $[\mathbf{u}]_p$ is itself uniform over $\mathcal{R}_p^m$.

Definition B.7 (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 $\text{MSIS}_{\mathcal{R}, m, n, q, \beta}$, 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 $\|\mathbf{v}\|_\infty \leq \beta$.

C Definitions and Proofs for Building Blocks

In this section, we present the formal definitions of our building blocks, including verifiable Ref-AH mmPKE and an event-oriented linkable tag scheme, and provide the security proofs for these constructions.

For convenience, we summarize the notations used in the Ref-AH mmPKE construction (and Lether) in Table 7 and restate the syntax of Ref-AH mmPKE and the event-oriented linkable tag scheme.

Table 7: Summary of main notations used in Lether, including Ref-AH mmPKE and tag scheme.

Notation	Description
λ	security parameter
ζ	correctness parameter
N	# of recipients, size of anonymity set
m, n	# of rows of $\mathbf{A}$, # of columns of $\mathbf{A}$
n'	dimension of linkable tag $\mathbf{v}_H$
q	modulus in Ref-AH mmPKE and LNP22
d	ring dimension of $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$
ν	ℓ_∞ -norm bound on private key (s_i, e_i)
$\bar{\nu}$	support size $\bar{\nu} \leq 2\nu + 1$ of private key (s_i, 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 integer message and the amount.

C.1 Verifiable Ref-AH mmPKE

Definition C.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 $\mathcal{C}_s$ consists of the following algorithms.

- $\text{pp}_{\text{Enc}} \leftarrow \text{mmSetup}(1^\lambda, N)$: On input a security parameter 1^λ and a recipient number N , it outputs a public parameter pp_{Enc} (which is an implicit input to all remaining algorithms).
- $(\text{pk}, \text{sk}) \leftarrow \text{mmKGen}()$: It outputs a public-private key pair $(\text{pk}, \text{sk}) \in \mathcal{K}$.
- $\text{ct} := (\hat{\text{ct}}, (\text{ct}_i)_{i \in [N]}) \leftarrow \text{mmEnc}((\text{pk}_i)_{i \in [N]}, (\mathbf{m}_i)_{i \in [N]}; \mathbf{r}, (\mathbf{r}_i)_{i \in [N]})$: On input N public keys $(\text{pk}_i)_{i \in [N]}$, N messages $(\mathbf{m}_i)_{i \in [N]}$,

$(N + 1)$ randomnesses $\mathbf{r}, (\mathbf{r}_i)_{i \in [N]}$, it outputs the multi-recipient ciphertext $\text{ct} := (\hat{\text{ct}}, (\text{ct}_i)_{i \in [N]})$.

- $\text{ct}_i := (\hat{\text{ct}}, \text{ct}_i) / \perp \leftarrow \text{mmExt}(i, \text{ct})$: On input a multi-recipient ciphertext $\text{ct} \in \mathcal{C}$, and an index $i \in [N]$, it deterministically outputs the individual ciphertext $\text{ct}_i \in \mathcal{C}_s$ or a symbol $\perp$ to indicate extraction failure.
- $\text{m} / \perp \leftarrow \text{mmDec}(\text{sk}, \text{ct})$: On input a private key sk , and an individual ciphertext $\text{ct} \in \mathcal{C}_s$, it outputs a message $\text{m} \in \mathcal{M}$ or a symbol $\perp$ to indicate decryption failure.
- $\text{ct}' / \perp \leftarrow \text{mmRef}(\text{pk}, \text{sk}, \text{ct})$: On input a public-private key pair (pk, sk) , and an individual ciphertext $\text{ct} \in \mathcal{C}_s$, it outputs a refreshed individual ciphertext $\text{ct}' \in \mathcal{C}_s$ or a symbol $\perp$ to indicate refresh failure.

Remark C.2 (Correctness). Let $\zeta : \mathbb{N} \rightarrow [0, 1]$. We say a Ref-AH mmPKE scheme is ζ -correct, if for all $\lambda, N \in \mathbb{N}$ and $i \in [N]$, message $\text{m}_i \in \mathcal{M}$, the following probability is at most $\zeta(\lambda)$,

$$\Pr \left[\begin{array}{c} \exists i \in [N] : \\ \text{mmDec}(\text{pp}, \text{sk}_i, \text{ct}_i) \neq \text{m}_i \end{array} \mid \begin{array}{c} \text{pp} \leftarrow \text{mmSetup}(1^\lambda, N); \\ \forall i \in [N] : (\text{pk}_i, \text{sk}_i) \leftarrow \text{mmKGen}(\text{pp}); \\ \text{ct} \leftarrow \text{mmEnc}(\text{pp}, (\text{pk}_i)_{i \in [N]}, (\mathbf{m}_i)_{i \in [N]}); \\ \text{ct}_i \leftarrow \text{mmExt}(\text{pp}, i, \text{ct}) \end{array} \right].$$

Remark C.3 (Security). Let RA-mmPKE be a Ref-AH mmPKE scheme, let $N \in \mathbb{N}$, and let λ denote the security parameter. We define the chosen-plaintext attack (CPA) security of Ref-AH mmPKE under the *knowledge-of-secret-key* (KOSK) assumption and the standard CPA security via the $\text{mmIND-CPA}^{\text{KOSK}}$ and mmIND-CPA security games, respectively, as shown in Figure 3.

For all PPT adversaries $\mathcal{A}$, we say that RA-mmPKE is $\text{mmIND-CPA}^{\text{KOSK}}$ secure if the advantage

$$\text{Adv}_{\text{RA-mmPKE}, N, \mathcal{A}}^{\text{mmIND-CPA}^{\text{KOSK}}}(\lambda) := \left| \Pr \left[\text{GAME}_{\text{RA-mmPKE}, N, \mathcal{A}}^{\text{mmIND-CPA}^{\text{KOSK}}}(\lambda) = 1 \right] - \frac{1}{2} \right|$$

is negligible in λ . Similarly, we say that RA-mmPKE is mmIND-CPA secure if the advantage

$$\text{Adv}_{\text{RA-mmPKE}, N, \mathcal{A}}^{\text{mmIND-CPA}}(\lambda) := \left| \Pr \left[\text{GAME}_{\text{RA-mmPKE}, N, \mathcal{A}}^{\text{mmIND-CPA}}(\lambda) = 1 \right] - \frac{1}{2} \right|$$

is negligible in λ .

We note that, using the KOSK compiler from [44] (i.e., requiring each recipient to prove knowledge of the corresponding private key), an $\text{mmIND-CPA}^{\text{KOSK}}$ secure (Ref-AH) mmPKE can be generically transformed into an mmIND-CPA secure scheme.

Looking ahead to the construction of Ref-AH mmPKE in Construction 3.2 and Lether system, here the KOSK assumption can be removed by requiring each recipient to prove knowledge of its private key during a *registration* phase, specifically by 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 incurring for exponential soundness degradation, as discussed in [9, 44].

Remark C.4 (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 $\text{pp}_{\text{Enc}} \leftarrow \text{mmSetup}(1^\lambda)$, any $(\text{pk}_i, \text{sk}_i) \leftarrow \text{mmKGen}()$ and $\text{m}_i^{(j)} \in \mathcal{M}$ for all $i \in [N]$, $j \in [T]$, the following holds,

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

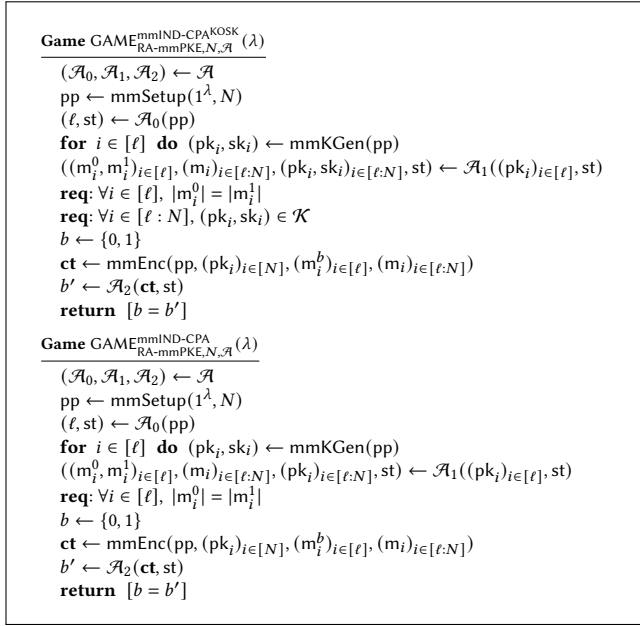


Figure 3: The CPA security games for Ref-AH mmPKE.

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

Remark C.5 (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 $\text{pp}_{\text{Enc}} \leftarrow \text{mmSetup}(1^\lambda)$, any $(\text{pk}, \text{sk}) \leftarrow \text{mmKGen}()$, any individual ciphertext $\text{ct} \in \mathcal{C}_s$ encrypted under pk and evaluated additively at most T times, there exists a PPT algorithm $\text{ct}' \leftarrow \text{mmRef}((\text{pk}, \text{sk}), \text{ct})$ such that $\text{mmDec}(\text{sk}, \text{ct}') = \text{mmDec}(\text{sk}, \text{ct})$ and ct' is a fresh ciphertext that supports at least T further additive homomorphic evaluations with other fresh or refreshed ciphertexts.

Definition C.6 (Soundness in Verifiable Multi-Encryption). Let $\text{mmPKE} = (\text{mmSetup}, \text{mmKGen}, \text{mmEnc}, \text{mmDec}, \text{mmExt})$ be a (Ref-AH) mmPKE scheme for message space $\mathcal{M}$. Let $\hat{\Pi} = (\hat{\Pi}.\text{Setup}, \hat{\Pi}.\text{Prove}, \hat{\Pi}.\text{Verify})$ be a verifiable multi-encryption scheme for mmPKE.

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

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

Definition C.7 (Unforgeability in Verifiable Decryption). Let $\text{mmPKE} = (\text{mmSetup}, \text{mmKGen}, \text{mmEnc}, \text{mmDec}, \text{mmExt})$ be a (Ref-AH) mmPKE scheme. Let $\hat{\Pi} = (\hat{\Pi}.\text{Setup}, \hat{\Pi}.\text{Prove}, \hat{\Pi}.\text{Verify})$ be a verifiable decryption scheme for mmPKE.

We say that $\hat{\Pi}$ is unforgeable if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $\text{pp}_{\text{Enc}} \leftarrow \text{mmSetup}(1^\lambda, N)$, any $\text{pp}_{\hat{\Pi}} \leftarrow \hat{\Pi}.\text{Setup}(1^\lambda)$,

the following probability is negligible in λ ,

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

Definition C.8 (Unforgeability in Verifiable Refresh). Let $\text{mmPKE} = (\text{mmSetup}, \text{mmKGen}, \text{mmEnc}, \text{mmDec}, \text{mmExt})$ be a (Ref-AH) mmPKE scheme. Let $\hat{\Pi} = (\hat{\Pi}.\text{Setup}, \hat{\Pi}.\text{Prove}, \hat{\Pi}.\text{Verify})$ be a verifiable refresh scheme for mmPKE.

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

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

Lemma C.9 (Soundness in Verifiable Multi-Encryption). Suppose LNP22 is knowledge sound. Then, our verifiable multi-encryption in Construction 3.5 is sound if the following probability is negligible,

$$\sum_{i \in [N]} \Pr[\|\langle \mathbf{e}_i, \bar{\mathbf{r}} \rangle + \bar{y}_i - \langle \mathbf{s}_i, \bar{\mathbf{e}}_u \rangle\|_\infty \geq \lfloor q/(4T+4) \rfloor] \quad (\text{C.1})$$

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

PROOF. If LNP22 is knowledge sound, there exists a PPT extractor that can extract the witness $(\bar{\mathbf{r}}, (\bar{\mathbf{m}}_i)_{i \in [N]})$ satisfying the relation R_{enc} in Equation (3.1).

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

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

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

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

Therefore, the decryption will fail if the decryption error satisfies $\|\langle \mathbf{e}_i, \bar{\mathbf{r}} \rangle + \bar{y}_i - \langle \mathbf{s}_i, \bar{\mathbf{e}}_u \rangle\|_\infty \geq \lfloor q/(4T+4) \rfloor$. $\square$

Lemma C.10 (Unforgeability in Verifiable Decryption). Suppose LNP22 is knowledge sound. Then, our verifiable decryption in Construction 3.7 is unforgeable if $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard.

PROOF. Suppose there exists an adversary $\mathcal{A}$ that can generate a valid proof π for a different message $\hat{\mathbf{m}}'$, where $\hat{\mathbf{m}} \neq \hat{\mathbf{m}}'$ for the original message $\hat{\mathbf{m}}$ encrypted by the challenger. If 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 (3.2).

Denote $\bar{\mathbf{e}} := \mathbf{b} - \mathbf{A}^\top \bar{\mathbf{s}}$ and $\mathbf{e} := \mathbf{b} - \mathbf{A}^\top \mathbf{s}$, where $\mathbf{s}$ is the correct private key of $\mathbf{b}$ honestly generated by the challenger. We first argue that $\bar{\mathbf{s}} = \mathbf{s}$. Otherwise, the adversary can break the $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption for the instance of $[\mathbf{I}|\mathbf{A}]$ with the solution $[(\bar{\mathbf{e}} - \mathbf{e})(\bar{\mathbf{s}} - \mathbf{s})]^\top$.

Denote $\bar{h} := \sum_{i=1}^o \delta_i \cdot \bar{b}_h^{(i)}$ for $\bar{b}_h \in \{0, 1\}^{od}$, and

$$h := c - \langle \mathbf{c}, \bar{\mathbf{s}}' \rangle - \lfloor q/(2T+2) \rfloor \cdot \bar{\mathbf{m}}, \quad (\text{C.2})$$

where the ciphertext (c, e) is honestly generated by the challenger. Thus, we have $\tilde{h}, h < \lfloor q/(4T+4) \rfloor$ and $\|\tilde{h} - h\|_\infty < \lfloor q/(2T+2) \rfloor$. Next, we can obtain from R_{dec} :

$$c - \langle c, \tilde{s} \rangle - \lfloor q/(2T+2) \rfloor \cdot \hat{m} = \tilde{h}, \quad (\text{C.3})$$

After subtracting Equation (C.3) and Equation (C.2), we can obtain $\lfloor q/(2T+2) \rfloor \cdot (\hat{m} - \hat{m}') = h - \tilde{h}$. Since $\hat{m} \neq \hat{m}'$ and $\|\lfloor q/(2T+2) \rfloor \cdot (\hat{m} - \hat{m}')\|_\infty \geq \lfloor q/(2T+2) \rfloor$, it contradicts the soundness of LNP22. $\square$

Lemma C.11 (Unforgeability in Verifiable Refresh). *Suppose LNP22 is knowledge sound. Then, our verifiable refresh in Construction 3.9 is unforgeable if $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard and the following probability is negligible,*

$$\Pr[\|\langle e, \tilde{r}' \rangle + \tilde{y}' - \langle s, \tilde{e}'_u \rangle\| \geq \lfloor q/(4T+4) \rfloor] \quad (\text{C.4})$$

where (e, s) is the private key in mmPKE and $(\tilde{r}', \tilde{m}')$ is the extracted witness along with $\tilde{e}'_u := c' - \text{Ar}', \tilde{y}' := c' - \langle b, \tilde{r}' \rangle - \lfloor q/(2T+2) \rfloor \cdot \tilde{m}'$.

PROOF SKETCH. The proof combines the arguments from Lemma C.9 and Lemma C.10. Briefly, 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 $\langle \tilde{0}^{d-k} \|\tilde{2}^k, \tilde{m} - \tilde{T}_d \rangle = \langle \tilde{0}^{d-k} \|\tilde{2}^k, \tilde{m}' \rangle$ ensures the consistency between the messages in the evaluated and refreshed ciphertexts. $\square$

C.2 Event-Oriented Linkable Tag

Definition C.12 (Tag Scheme). A tag scheme with a public-private key pair space $\mathcal{K}$, a tag space $\mathcal{T}$ consists of the following algorithms.

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

Definition C.13 (Event-Oriented Linkability). Let Tag be a tag scheme. We say that Tag is event-oriented linkable if for any PPT adversary $\mathcal{A}$, any $(\text{pp}_{\text{tag}}, \text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$, the following probability is negligible in λ ,

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

Definition C.14 (Multi-Tag Anonymity). Let Tag be a tag scheme. We say that Tag is multi-tag anonymous if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $(\text{pp}_{\text{tag}}, \text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$, any $\ell \in \mathbb{N}$, the following probability is negligible in λ ,

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

Definition C.15 (Non-Frameability). Let Tag be a tag scheme. We say that Tag is non-frameable if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, any $(\text{pp}_{\text{tag}}, \text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda)$, the following probability is negligible in λ ,

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

Lemma C.16 (Event-Oriented Linkability in Tag Scheme). *Suppose LNP22 is knowledge sound. Then our tag scheme in Construction 3.12 is event-oriented linkable if $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard.*

PROOF. Suppose there exists an adversary $\mathcal{A}$ can generate two valid proofs π_0, π_1 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 $(\tilde{s}_0, \tilde{\mathbf{e}}_H^0), (\tilde{s}_1, \tilde{\mathbf{e}}_H^1)$ satisfying the relation R_{tag} in Equation (3.4).

Similar to Lemma C.10, we can argue that $\tilde{s}_0 = \tilde{s}_1$ holds if the $\text{MSIS}_{\mathcal{R}, m, (m+n), q, v}$ assumption is hard.

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

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

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

$$\mathbf{A}_H \cdot \tilde{\mathbf{s}}_i - \tilde{\mathbf{e}}_H^i - \hat{q}/\hat{p} \cdot \mathbf{v}_H^i \bmod \hat{q} \equiv 0 \bmod \hat{q}. \quad (\text{C.5})$$

Later, we subtract Equation (C.5) for $i = \{0, 1\}$ to get

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

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

Lemma C.17 (Multi-Tag-Anonymity in Tag Scheme). *Our tag scheme in Construction 3.12 is multi-tag-anonymous if $\text{MLWR}_{\mathcal{R}, n, n', \hat{q}, \hat{p}, \chi}$ assumption for $\chi = \mathcal{U}(\mathbb{S}_v)$ is hard.*

PROOF SKETCH. The tags are generated as $\mathbf{v}_H^i := [\mathbf{A}_H^i \cdot \mathbf{s} \bmod \hat{q}]_{\hat{p}}$ for $i \in [\ell]$, which are instances of the MLWR assumption. Therefore, as demonstrated in [12], 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. $\square$

Lemma C.18 (Non-Frameability in Tag Scheme). *Suppose LNP22 is knowledge sound. Then our tag scheme in Construction 3.12 is non-frameable if $\text{MLWE}_{\mathcal{R}, n, m, q, \tilde{\chi}}$ assumption for $\tilde{\chi} := \mathcal{U}(\mathbb{S}_v)$ and $\text{MSIS}_{\mathcal{R}, n', (m+n'), \hat{q}, \beta}$ assumption for $\beta = \max(\hat{q}/\hat{p} - 1, v)$ 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 $(\tilde{\mathbf{s}}', \tilde{\mathbf{e}}_H')$ satisfying the relation in Equation (3.4).

Therefore, we have

$$\mathbf{v}_H \bmod \hat{q} \equiv \mathbf{A}_H \cdot \mathbf{s} + \mathbf{e}_H \bmod \hat{q} \quad (\text{C.6})$$

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

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

$$\mathbf{v}_H \bmod \hat{q} \equiv \mathbf{A}_H \cdot \mathbf{s}' + \mathbf{e}'_H \quad (\text{C.7})$$

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

Then, we subtract Equation (C.7) from Equation (C.6) to obtain a short solution $[\mathbf{s} - \tilde{\mathbf{s}} \mid \mathbf{e}_H - \tilde{\mathbf{e}}_H]$ to the MSIS $_{\mathcal{R}, m, (m+n), \hat{q}, \beta}$ instance defined by $[\mathbf{A}_H \mid \mathbf{I}_{n'}]^\top$ for $\beta = \max(\hat{q}/\hat{p} - 1, \nu)$. If the solution $[\mathbf{s} - \tilde{\mathbf{s}} \mid \mathbf{e}_H - \tilde{\mathbf{e}}_H]$ is zero, then the adversary can break the MLWE $_{\mathcal{R}, n, m, q, \tilde{\chi}}$ instance defined by $\mathbf{A}^\top \mathbf{s} + \mathbf{e}$. $\square$

D 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 captures Anonymous Zether-like protocols, e.g., [7, 11]. We also highlight the key differences between our model and RingCT-like protocols used in UTXO-based blockchains, such as [15, 16, 27, 41, 45, 48, 49].

The main advantage of our definition is it explicitly captures the inherent stateful nature of blockchain environments and incorporates the notion of epochs, which is not considered in [11].

Similar to [11], here we focus primarily on transfer security; for the security of funding and withdrawal (burn), we refer the reader to [7].

Definition D.1 (Account-Based Private Blockchain Payment). We first introduce the notation used specifically for the security model in Table 8. The blockchain state $\mathbb{S}$ consists of: (1) a table $\text{acc}[\cdot]$ of registered accounts, where each public key pk indexes an account as $\text{acc}[\text{pk}]$; (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:

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: On input a security parameter λ , it outputs the system public parameters pp . We assume pp is an implicit input to all remaining algorithms.
- $(\text{pk}, \text{sk}, \pi) \leftarrow \text{AddrGen}()$: It generates a public-private key pair (pk, sk) and a proof π of correct key generation. The public key pk serves as the account *address*.
- $\mathbb{S} \leftarrow \text{Register}(\text{pk}, \pi, \mathbb{S})$: On input a public key pk and a proof π , it updates the state by initializing $\text{acc}[\text{pk}]$ if the proof is valid.
- $\text{tag}_H \leftarrow \text{TagGen}(\text{sk}, H)$: On input a private key sk and the current epoch H , it outputs a linkable tag tag_H .
- $\mathbf{ct} \leftarrow \text{AmtGen}((\text{amt}_i)_{i \in [N]}, (\text{pk}_i)_{i \in [N]})$: On input a set of amounts $(\text{amt}_i)_{i \in [N]}$ and corresponding public keys $(\text{pk}_i)_{i \in [N]}$, it outputs a set of amount ciphertexts $\mathbf{ct}$.
- $\mathbb{S} \leftarrow \text{RollOver}((\text{pk}_i)_{i \in [N]}, \mathbf{ct}, \mathbb{S})$: On input a set of addresses $(\text{pk}_i)_{i \in [N]}$, a set of amount ciphertexts $\mathbf{ct}$, and the blockchain state $\mathbb{S}$, it updates the state by setting $\text{acc}[\text{pk}_i] := \text{ct}_i \oplus \text{acc}[\text{pk}_i]$ where $\oplus$ denotes the additively-homomorphic evaluation.
- $\text{tx} \leftarrow \text{AnTransfer}((\text{pk}_i)_{i \in [N]}, s, r, \text{sk}_s, \text{amt}, \mathbb{S})$: On input an anonymity set of public keys $(\text{pk}_i)_{i \in [N]}$, spender index s , recipient index r , the spender's private key sk_s , the transfer amount amt , and the current state $\mathbb{S}$, it generates amount ciphertexts $\mathbf{ct} \leftarrow \text{AmtGen}((\text{amt}_i)_{i \in [N]}, (\text{pk}_i)_{i \in [N]})$ such that $\text{amt}_s = -\text{amt}$, $\text{amt}_r =$

amt , and $\text{amt}_i = 0$ for all $i \in [N] \setminus \{s, r\}$, a tag $\text{tag}_H \leftarrow \text{TagGen}(\text{sk}_s, H)$ and a proof Π . It outputs the transaction $\text{tx} = ((\text{pk}_i)_{i \in [N]}, \mathbf{ct}, \text{tag}_H, \Pi)$.

- $0/1 \leftarrow \text{LinkTag}(\text{tag}, \mathbb{S})$: On input a tag tag and the blockchain state $\mathbb{S}$, it outputs 1 if the tag already appears in $\mathbb{S}$, and 0 otherwise.
- $0/\mathbb{S} \leftarrow \text{Verify}(\text{tx}, \mathbb{S})$: On input a transaction tx and the blockchain state $\mathbb{S}$, if the proof Π is valid and $\text{LinkTag}(\text{Tag}, \mathbb{S}) = 0$, it updates the state by recording tx and running $\text{RollOver}((\text{pk}_i)_{i \in [N]}, \mathbf{ct}, \mathbb{S})$ with the associated public keys and ciphertexts¹⁰; otherwise, output 0.
- $m \leftarrow \text{ReadBalance}((\text{pk}, \text{sk}), \mathbb{S})$: On input a public-private key pair (pk, sk) and the blockchain state $\mathbb{S}$, it outputs the balance m of the account $\text{acc}[\text{pk}]$.

Table 8: Notations for the account-based private blockchain payment formal model.

Symbol	Description
$\mathbb{S}$	the blockchain state
$\text{acc}[\text{pk}]$	an account indexed by the address (public key) pk
N	# of public keys in the anonymity set
$(\text{pk}_i)_{i \in [N]}$	anonymity set of public keys with $ (\text{pk}_i)_{i \in [N]} = N$
$\mathbf{ct}$	set of ciphertext with $ \mathbf{ct} = N$
s, r	the indices to indicate the spender 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 $\text{tx} = ((\text{pk}_i)_{i \in [N]}, \mathbf{ct}, \text{tag}_H, \Pi)$
V	a set of all valid balances/amounts, $V \subseteq [0, \dots, \text{MAX}]$

Like [16], 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.

We highlight the main differences between (Anonymous Zether-like) account-based private blockchain payments and UTXO-based RingCT-like 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 $\text{tag}_H \leftarrow \text{TagGen}(\text{sk}, H)$ 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*, thereby allowing a non-negative transaction amount to be “subtracted” from the spender’s balance when the ciphertext is added to the account. In UTXO-based RingCT, the input to the Mint algorithm 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 $(\text{acc}[\text{pk}_i])_{i \in [N]}$. In UTXO-based

¹⁰Similar to [11], we assume for simplicity that each transaction takes effect immediately.

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 (Anonymous Zether-like) account-based private blockchain payment is linear in the size of the anonymity set. As stated in (Anonymous) Zether [7, 11], 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 spender 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 [49], which requires publishing auxiliary data for all coins in the anonymity set to hide the number of input coins.

D.1 Security Definition

We define the list $\mathcal{U}$ in Table 9. Following [16], the list $\mathcal{U}$ can be viewed as a database. A public key pk , a secret key sk , and a linkable tag tag_H in $\mathcal{U}$ can serve as unique identifiers for rows to retrieve associated information. For example, $\mathcal{U}[pk].acc$ denotes the account associated with the public key pk . tag_H denotes the linkable tag of the account for the current epoch H . $PreBal$ 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.

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

Table 9: Structure of the list $\mathcal{U}$ used in the security model

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

- **AdGen(i):** On input a query number i , it runs $(pk_i, sk_i, \pi_i) \leftarrow \text{AddrGen}()$ and outputs pk_i . It adds (pk_i, 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):** On 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 $\mathcal{U}[pk].sk$, $\mathcal{U}[pk].acc$ and $\mathcal{U}[pk].balance$.
- **Transfer($((pk_i)_{i \in [N]}, s, r, amt, \mathbb{S})$):** On input an anonymity set $(pk_i)_{i \in [N]}$ including the spender public key pk_s , the recipient public key pk_r , a transaction amount amt , and the blockchain state $\mathbb{S}$. It first retrieve sk_s from $\mathcal{U}$, runs $tx \leftarrow \text{AnTransfer}((pk_i)_{i \in [N]}, s, r, sk_s, amt, \mathbb{S})$ and $B \leftarrow \text{Verify}(tx, \mathbb{S})$. If $B = 0$ indicating the verification fails, it outputs $\perp$. Otherwise, it sets $\mathcal{U}[pk_i].acc := \mathcal{U}[pk_i].acc + ct_i$, $\mathcal{U}[pk_i].PreBal := \mathcal{U}[pk_i].CurBal$ for all $i \in [N]$, and sets $\mathcal{U}[pk_s].CurBal := \mathcal{U}[pk_s].CurBal - amt$, $\mathcal{U}[pk_r].CurBal := \mathcal{U}[pk_r].CurBal + amt$. 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 protocol is said to be ϵ -correct if for any $pp \leftarrow \text{Setup}(1^\lambda)$, any $N \in \mathbb{Z}^+$, any blockchain state $\mathbb{S}$, any $(pk_s, sk_s, \pi_s) \leftarrow \text{AddrGen}()$ that has

been appropriately funded or received transferred funds for $m \leftarrow \text{ReadBalance}(sk_s, \mathbb{S})$, at any epoch $H \leftarrow \mathbb{S}$, any amount $amt \in \mathbb{V}$ for $amt \leq m$, and any set $(pk_i)_{i \in [N] \setminus \{s\}}$ of arbitrarily registered decoy public keys along with the recipient’s public key pk_r , the following probability holds

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

If $\epsilon = 0$, then the protocol is said to be *perfectly correct*. If $\epsilon = \text{negl}(\lambda)$, 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 spender and the recipient are hidden among uncorrupted decoy accounts. Therefore, we adapt the anonymity security model from [16] to the account-based blockchain setting and divide it into two cases as follows.

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

Definition D.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 $pp \leftarrow \text{Setup}(1^\lambda)$

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

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

where the game Exp_{AN} is defined as follows:

- (1) $((pk_i)_{i \in [N]}, s^0, s^1, r^0, r^1, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pp)$: $\mathcal{A}$ is given pp and access to all oracles, and then it outputs a set of registered public keys $(pk_i)_{i \in [N]}$, the target spender indices $s^0, s^1 \in [N]$, the target recipient indices $r^0, r^1 \in [N]$, and some state information st to be used by $\mathcal{A}$ in the next stage.
- (2) $amt \leftarrow \mathbb{V}$: The challenger samples an amount from set $\mathbb{V}$.
- (3) $tx_{i,j} \leftarrow \text{AnTransfer}((pk_i)_{i \in [N]}, s^i, r^j, sk_{s^i}, amt, \mathbb{S})$ for $i, j \in \{0, 1\}$: Both s^0 and s^1 are spent to both r^0 and r^1 respectively, where sk_{s^0}, sk_{s^1} are retrieved from $\mathcal{U}$. If $\text{Verify}(tx_{i,j}, \mathbb{S}) = 0$ for some $i, j \in \{0, 1\}$, then set all $tx_{i,j} := \perp$.
- (4) $b_s, b_r \leftarrow \{0, 1\}$.
- (5) $b'_s, b'_r \leftarrow \mathcal{A}^{\mathcal{O}}(tx_{b_s, b_r}, st, \mathbb{S})$: $\mathcal{A}$ 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 spender and recipient of tx_{b_s, b_r} .

$\mathcal{A}$ wins the game $\text{Exp}_{\text{AN}}(\mathbb{S})$ if $b'_s = b_s$ and wins the game $\text{Exp}_{\text{AN}}(\mathbb{R})$ if $b'_r = b_r$ if the following conditions hold:

- all public keys in $(pk_i)_{i \in [N]}$ are honestly generated and verified in register algorithm,
- all accounts of pk_{s^i}, pk_{r^i} for $i \in \{0, 1\}$ are not be corrupted (i.e. not queried to *Corrupt*),
- $tx_{i,j} \neq \perp$ for all $i, j \in \{0, 1\}$,
- pk_{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 spender is hidden among all the uncorrupted addresses in the anonymity set.

Definition D.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 $\text{pp} \leftarrow \text{Setup}(1^\lambda)$

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

where the game Exp_{AR} is defined as follows:

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

$\mathcal{A}$ wins the game Exp_{AR} if the following holds:

- all public keys in $(pk_i)_{i \in [N]}$ are honestly generated and verified in register algorithm,
- both accounts of pk_{s^i} for $i \in \{0, 1\}$ are not be corrupted,
- $\text{tx}_i \neq \perp$ for all $i \in \{0, 1\}$,
- pk_{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 spender and the recipient can learn the transaction amount, even if the identities of the spender and recipient are revealed.

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

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

where the game Exp_{CON} is defined as follows:

- (1) $((pk_i)_{i \in [N]}, \text{amt}^0, \text{amt}^1, s, r, \text{st}) \leftarrow \mathcal{A}^O(\text{pp})$: $\mathcal{A}$ is given pp and access to all oracles, and then it outputs a set of registered public keys $(pk_i)_{i \in [N]}$, the target amounts $\text{amt}^0, \text{amt}^1 \in V$, a spender index $s \in [N]$, a recipient index $r \in [N]$, and some state information st to be used by $\mathcal{A}$ in the next stage.
- (2) $\text{tx}_i \leftarrow \text{AnTransfer}((pk_i)_{i \in [N]}, s, r, \text{sk}_s, \text{amt}_i^i, \mathbb{S})$ for $i \in \{0, 1\}$: amt^0 and amt^1 are transferred from s to r respectively, where sk_s is retrieved from $\mathcal{U}$. If $\text{Verify}(\text{tx}_i, \mathbb{S}) = 0$ for some $i \in \{0, 1\}$, then set all $\text{tx}_i = \perp$.
- (3) $b \xleftarrow{\$} \{0, 1\}$

- (4) $b' \leftarrow \mathcal{A}^O(\text{tx}_b, \text{st}, \mathbb{S})$: $\mathcal{A}$ 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 bit b' for the real amount of tx_b .

$\mathcal{A}$ wins the game Exp_{CON} if the following holds:

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

Balance. Here, we adapt the balance security model from [16, 48] 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 anonymity set;
- (3) overdraft her own account;
- (4) double spend her *own* account within the same epoch.

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

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

where the game Exp_{BAL} is defined as follows:

- (1) $(\text{tx}^1, \dots, \text{tx}^t) \leftarrow \mathcal{A}^O(\text{pp})$: the adversary $\mathcal{A}$ is given access to all the oracles together with pp and outputs a set of T transactions $(\text{tx}^1, \dots, \text{tx}^t)$, where $\text{tx}^i = ((pk_j^i)_{j \in [N]}, \text{ct}^i, \text{tag}^i, \Pi^i)$. Without loss of generality, we suppose the T transactions are all in the same epoch.¹¹
- (2) $B^i \leftarrow \text{Verify}(\text{tx}^i, \mathbb{S})$ for $i = 1, \dots, t$.

$\mathcal{A}$ wins the game Exp_{BAL} if the following holds

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

¹¹As (Anonymous) Zether [7, 11] 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.

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 spender 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 spender's and recipient's amounts (including amounts associated with other accounts in the anonymity 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 spender's available balance. In other words, the transaction causes at least one account in the anonymity set to have 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 during the same epoch by generating distinct linkable tags. This is captured by Case 4.

Like [15, 16], 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 [11] only allows the adversary to produce a single transaction.

In our formal definition, we explicitly state our modeling assumptions to facilitate future extensions that relax them. One possible enhancement is to remove the assumption in Exp_{BAL} that all accounts in the anonymity set correspond to well-formed and correctly maintained states. In the current model, this assumption follows from the correctness of the registration phase and the validity of previous transactions from earlier epochs. Relaxing this assumption would require substantially more involved balance analyses and is left for future work. We note that a similar assumption is made, either explicitly or implicitly, in prior account-based private payment systems such as (Anonymous) Zether [7, 11].

E Security Proofs for Lether

E.1 Anonymity

Theorem E.1 (Anonymity against Non-Recipients). *Lether in Section 4 is anonymous against non-recipients if LNP22 is simulatable, tag scheme is tag-anonymous, and Ref-AH mmPKE is mmIND-CPA^{KOSK} secure. More precisely, for any PPT adversary $\mathcal{A}$ against Exp_{AN} , there exists PPT adversaries $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$ against simulatability of LNP22, tag-anonymity of tag scheme, and mmIND-CPA^{KOSK} security of Ref-AH mmPKE, such that*

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

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

Game₀: This is identical to Exp_{AN} . Thus, we have $\text{Adv}_{\mathcal{A}}^{\text{Game}_0} = \text{Adv}_{\mathcal{A}}^{\text{Exp}_{\text{AN}}}$.

Game₁: 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

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

Game₂: 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 spender public keys pk_s^0 and pk_s^1 . Thus, there exists a PPT adversary $\mathcal{B}_1$ whose running time is approximately that of $\mathcal{A}$ such that

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

Game₃: This game is the same as Game_2 , except that the challenger modifies the amount ciphertext **ct**.

Specifically, the challenger changes the ciphertext $\mathbf{ct} = (c, (c_i)_{i \in [N]})$ 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 (i.e., $i \in \{s^0, s^1, r^0, r^1\}$), each $v_i \leftarrow \mathcal{U}(\mathcal{R}_q)$ is also uniformly sampled. For the other account (i.e., possible corrupted accounts), the ciphertext is reproduced as $v_i := \langle \mathbf{u}, \mathbf{s}_i \rangle + h_i + \lfloor q/(2T+2) \rfloor \cdot m_i$ where $h_i := \langle -\mathbf{s}_i || \mathbf{e}_i, \mathbf{e}_u || \mathbf{r} \rangle + y_i$ is decryption error and $m_i = 0$ because the corrupted accounts are all decoys. Here, the challenger can obtain each private key $(\mathbf{s}_i, \mathbf{e}_i)$ under the KOSK assumption. We note that the KOSK assumption can be removed if each user is required to provide a proof of knowledge of the private key during the registration phase (as demonstrated in [44] and Remark C.3).

Therefore, based on the security of (Ref-AH) mmPKE established in [44], there exists a PPT adversary $\mathcal{B}_2$ whose running time is approximately that of $\mathcal{A}$ such that

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

Note that the ciphertexts for the accounts of the challenge senders and recipients $\text{pk}_s^i, \text{pk}_r^i$ for $i \in \{0, 1\}$ are uniformly random. Furthermore, the output of AnTransfer is independent of pk_s^i and pk_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. $\square$

Theorem E.2 (Anonymity against Recipient). *Lether in Section 4 is anonymous against recipient if LNP22 is simulatable, tag scheme is tag-anonymous, and Ref-AH mmPKE is mmIND-CPA^{KOSK} secure. More precisely, for any PPT adversary $\mathcal{A}$ against Exp_{AR} , there exists PPT adversaries $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$ against simulatability of LNP22, tag-anonymity of tag scheme, and mmIND-CPA^{KOSK} security of Ref-AH mmPKE, such that*

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

PROOF SKETCH. The proof is analogous to Theorem E.1; so we provide only a sketch. **Game₀** is identical to Exp_{AR} . **Game₁**, **Game₂**, and **Game₃** are the same as those in Theorem E.1, except that in **Game₃**, since the recipient is corrupted, the challenger constructs its ciphertext as $v_i = \langle \mathbf{u}, \mathbf{s}_r \rangle + h_i + \lfloor q/(2T+2) \rfloor m$. Thus, in **Game₃**,

the adversary $\mathcal{A}$ has a success probability of $1/2$ in outputting $b' = b$. $\square$

E.2 Confidentiality

Theorem E.3 (Confidentiality). *Lether in Section 4 is confidential if LNP22 is simulatable and Ref-AH mmPKE is mmIND-CPA^{KOSK} secure. More precisely, for any PPT adversary $\mathcal{A}$ against Exp_{CON} , there exists PPT adversaries $\mathcal{B}_0, \mathcal{B}_1$ against simulatability of LNP22, and mmIND-CPA^{KOSK} security of Ref-AH mmPKE, such that*

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

PROOF SKETCH. The proof is analogous to Theorem E.1, so we provide only a sketch. **Game₀** is identical to Exp_{CON} . **Game₁** is the same as **Game₁** in Theorem E.1. **Game₂** corresponds to **Game₃** in Theorem E.1. Thus, in **Game₂**, the adversary $\mathcal{A}$ has a success probability of $1/2$ in outputting $b' = b$. $\square$

E.3 Balance

Theorem E.4 (Balance). *Lether in Section 4 is balanced if LNP22 is sound, tag scheme is event-oriented linkable, and MLWE_{R,n,m,q,\tilde{\chi}} assumption for $\tilde{\chi} := \mathcal{U}(\mathbb{S}_v)$ is hard.*

PROOF. Let $\mathcal{A}$ be a PPT adversary against the balance property in the game Exp_{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 $j^* \in \{1, \dots, t\}$, and for all $i \in [N]$, there exists $\text{amt}_i^{j^*} < 0$, where $\text{amt}_i^{j^*} := \mathcal{U}[\text{pk}_i^{j^*}].\text{CurBal} - \mathcal{U}[\text{pk}_i^{j^*}].\text{PreBal}$, and either $\mathcal{U}[\text{pk}_i^{j^*}].\text{IsCrpt} = 0$ or $\text{pk}_i^{j^*} \neq \mathcal{U}[\text{tag}^{j^*}].\text{pk}$.

Since $\mathcal{A}$ is able to generate a valid proof Π , the extractability of LNP22 implies the existence of an extractor that can extract a witness satisfying R_{an} in Equation (4.2).

For the first situation (i.e., $\mathcal{U}[\text{pk}_i^{j^*}].\text{IsCrpt} = 0$), from the soundness of verifiable encryption for mmPKE (Lemma C.9), we obtain that each amount in the ciphertext satisfies $\text{amt}_i^{j^*} = (b_i^{(r)} - b_i^{(s)}) \cdot \hat{m}$, where $\langle \vec{0}^{d-k} | \vec{2}^k, \vec{m} \rangle \geq 0$. Therefore, for some $i^* \in [N]$ such that $\text{amt}_{i^*}^{j^*} < 0$, it must hold that $b_{i^*}^{(s)} = 1$. The extracted private key $\tilde{s}_s$ must satisfy $\|(\tilde{s}_s | \sum_{i \in [N]} b_i^{(s)} \cdot \mathbf{b}_i - \mathbf{A}^\top \tilde{s}_s)\|_\infty \leq v$. Given that $\mathcal{U}[\text{pk}_{i^*}^{j^*}].\text{IsCrpt} = 0$, i.e., $\mathcal{A}$ does not possess the private key corresponding to $\mathbf{b}_{i^*}$, and $b_{i^*}^{(s)} = 1$, $b_i^{(s)} = 0$ for all $i \in [N] \setminus \{i^*\}$, the adversary can solve the MLWE_{R,n,m,q,\tilde{\chi}} instance on input $([\mathbf{I} | \mathbf{A}], \mathbf{b}_{i^*})$, where $\tilde{\chi} := \mathcal{U}(\mathbb{S}_v)$ —leading to a contradiction.

In the second situation (i.e., $\text{pk}_{i^*}^{j^*} \neq \mathcal{U}[\text{tag}^{j^*}].\text{pk}$), it reduces to breaking the event-oriented linkability property of the tag scheme, as formalized in Lemma C.16. Specifically, for some $i^* \in [N]$ such that $\text{amt}_{i^*}^{j^*} < 0$ and $\text{pk}_{i^*}^{j^*} \neq \mathcal{U}[\text{tag}^{j^*}].\text{pk}$, it implies that $\mathcal{A}$ is able to use the private key corresponding to $\text{pk}_{i^*}^{j^*}$ to generate a valid tag that is unlinkable to $\text{pk}_{i^*}^{j^*}$, 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 $j^* \in \{1, \dots, t\}$ satisfying $\sum_{i \in [N]} \text{amt}_i^{j^*} \neq 0$, where $\text{amt}_i^{j^*} := \mathcal{U}[\text{pk}_i^{j^*}].\text{CurBal} - \mathcal{U}[\text{pk}_i^{j^*}].\text{PreBal}$.

Since the adversary $\mathcal{A}$ is able to generate a valid proof Π , the extractability of LNP22 ensures that there exists an extractor that can extract a witness satisfying R_{an} as defined in Equation (4.2).

From the soundness of verifiable encryption in mmPKE (Lemma C.9), we know that each amount in the ciphertext satisfies $\text{amt}_i^{j^*} = (b_i^{(r)} - b_i^{(s)}) \cdot \hat{m}$, where $\langle \vec{0}^{d-k} | \vec{2}^k, \vec{m} \rangle \geq 0$. Summing over all $i \in [N]$, we obtain:

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

Since both $\sum_{i \in [N]} b_i^{(r)} = 1$ and $\sum_{i \in [N]} b_i^{(s)} = 1$, we have $\sum_{i \in [N]} \text{amt}_i^{j^*} = (1 - 1) \cdot \hat{m} = 0$. This contradicts the assumption that the sum is non-zero and thus violates the soundness of 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 $j^* \in \{1, \dots, t\}$ satisfying $\mathcal{U}[\text{pk}_i^{j^*}].\text{CurBal} < 0$ for some $i \in [N]$.

Since $\mathcal{A}$ can produce a valid proof Π , the extractability of LNP22 ensures that an extractor can recover a witness satisfying R_{an} as defined in Equation (4.2).

From the soundness of verifiable encryption in mmPKE (Lemma C.9), each amount in the ciphertext satisfies $\text{amt}_i^{j^*} = (b_i^{(r)} - b_i^{(s)}) \cdot \hat{m}$, with $\langle \vec{0}^{d-k} | \vec{2}^k, \vec{m} \rangle \geq 0$. Let $s := i^*$ be the index such that $b_{i^*}^{(s)} = 1$. Then, only the balance of account $\text{acc}[\text{pk}_{i^*}^{j^*}]$ 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 $\text{acc}[\text{pk}_{i^*}^{j^*}]$ remains non-negative.

From the unforgeability of verifiable decryption in mmPKE (Lemma C.10), the decrypted balance satisfies $\langle \vec{0}^{d-k} | \vec{2}^k, \vec{m}' - \vec{T}_d \rangle \in \{0, \dots, 2^k - 1\}$ which implies that the balance is non-negative. This contradicts the assumption in $\mathcal{E}_{\text{overdraft}}$ and therefore violates the soundness of 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 $j^* \in \{1, \dots, t\}$ satisfying $\text{tag}^{j^*} \notin (\mathcal{U}[\text{pk}_i^{j^*}].\text{tag})_{i \in [N]}$.

Since $\mathcal{A}$ can produce a valid proof Π , the extractability of LNP22 ensures that an extractor can recover a witness satisfying R_{an} as defined in Equation (4.2). In particular, the extracted binary vector $\mathbf{b}^{(s)}$ has Hamming weight 1. Let $s := i^*$ be the unique index such that $b_{i^*}^{(s)} = 1$.

Hence, the adversary effectively generates a valid tag tag^{j^*} using the private key of $\text{pk}_{i^*}^{j^*}$ 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 C.16. $\square$