cryptography

MDPI

Article

LACT+: Practical Post-Quantum Scalable Confidential Transactions

Jayamine Alupotha *, Xavier Boyen and Matthew McKague

School of Computer Science, Queensland University of Technology, Brisbane 4000, Australia
* Correspondence: alupotha@qut.edu.au

Abstract: A "*confidential monetary value*" carries information about the real monetary value but does not disclose it. Post-quantum private blockchains with confidential monetary values—large-sized blockchains with large verification times—have the least scalability because they need to save and verify more information than those with "*plain-text monetary values*". High scalability is an essential *security* requirement for decentralized blockchain payment systems because the more honest peers who can afford to verify the blockchain copies are, the higher the security. We propose a quantum-safe transaction protocol for confidential monetary blockchains, LACT+ (*Lattice-based Aggregable Confidential Transactions*), which is more scalable than previous post-quantum confidential blockchains, i.e., many input/output transactions with logarithmic sized complexity.

Keywords: blockchains; confidential transactions; post-quantum cryptography; scalability; lattice-based cryptography

Citation: Alupotha, J.; Boyen, X.; McKague, M. LACT+: Practical Post-Quantum Scalable Confidential Transactions. *Cryptography* **2023**, *7*, 24. https://doi.org/10.3390/cryptography7020024

Academic Editor: Licheng Wang

Received: 17 February 2023
Revised: 12 April 2023
Accepted: 24 April 2023
Published: 8 May 2023

1. Introduction

Blockchain-based cryptocurrencies have become popular due to the security of decentralization, i.e., there is no single point of failure and no single authority in control. However, decentralization has a downside because it requires many honest peers to keep a copy of the blockchain and constantly verify it. Otherwise, malicious peers can alter the cash system for their gain. The number of honest peers mainly depends on the system's scalability (size and verification time) because peers have to spend their resources like space and computational power to save and verify blockchains. Despite the security and privacy, "quantum-safe" and "confidential" cryptocurrencies have the least scalability because they need special computations and algorithms that require more space and more computational power. This paper introduces a more scalable quantum-safe confidential blockchain transaction method, LACT+, where we only need 5.7 KB to store a confidential monetary value, whereas refs. [1,2] need >30 KB and 9.6 KB, respectively.

Confidential Transactions. Cryptocurrencies are public records accessible by anyone. Even though pseudonyms are commonly used, refs. [3–9] show that these identities can be linked to real identities. Therefore, maintaining monetary amounts in "plain text" is a major privacy concern. Confidential transactions (CT) solve this problem by keeping monetary values confidential yet providing tools to verify that hidden coin amounts are not negative, stolen, or double spent. However, this enhanced privacy comes at the cost of scalability because these tools generate proofs that should be stored in the blockchain, and verification of these proofs takes more computational power than simply checking plain-text numbers.

Aggregable Transactions. A transaction takes a set of current coins, changes their ownership, and updates the blockchain with that information. Hence, blockchain verifiers need to know "who has coins? Furthermore, are they only spending what they have?". Hence, they mainly need to know about current coins and who *has* them, not who *had* them. As the name implies, aggregable transactions aggregate multiple transactions into a smaller

transaction, and this aggregated transaction shows "who has coins after those transactions". For example, if $\mathcal{P}_1$ sends coins to $\mathcal{P}_2$ and $\mathcal{P}_3$, and $\mathcal{P}_2$ sends their coins to $\mathcal{P}_4$, the aggregated transaction shows that $\mathcal{P}_3$ and $\mathcal{P}_4$ have coins, and forgets that $\mathcal{P}_1$ and $\mathcal{P}_2$ had coins. Hence, an aggregable blockchain is a single aggregated transaction of all previous transactions. This forgotten information reduces the size and verification time and improves scalability significantly. However, previous aggregable blockchains refs. [1,2,10–14] cannot prove that the aggregated blockchain is the aggregated version of the "right" or the consensus accepted blockchain (more theoretically, previous aggregated blockchains are not "immutable"). Hence, despite the high scalability, "peers" have to download and verify the complete blockchain.

Quantum Safety. Recent developments in quantum computing, e.g., IBM's roadmap ref. [15], show that quantum adversaries can be a real threat even to the current standardized security level in the near future. Post-quantum (we use "post-quantum", "quantum-safe", or "quantum-resistant" to describe plausibly quantum-secure protocols) payments started receiving attention because many established decentralized payments are based on the discrete-logarithmic problem (DLP), which is vulnerable to quantum adversaries. Not only transparent payments, but many aggregable confidential transactions refs. [10–14] are also based on DLP and equally vulnerable to quantum adversaries. Ref. [1,2] introduced two quantum-safe aggregable confidential transactions that are based on the short integer solution problem (SIS) and the modular SIS. However, ref. [1] has an inefficient proof system and the transactions of ref. [2] are limited to a maximum two inputs and two outputs.

This paper proposes an immutable aggregable confidential transaction protocol, LACT+ (Lattice-based Aggregable Confidential Transactions), based on approximate modular SIS (approximate SIS), to provide post-quantum security with practical scalability.

1.1. Related Work

Let us explain the key components of decentralized aggregable payments that lead to LACT+. Note that here, we sketch a common protocol that describes refs. [1,2,10–14] generally regardless of their "hardness assumption", e.g., they can be based on DLP or SIS.

Digital Coins from Cryptographic View. Decentralized payment systems store monetary values as digital coins where each coin represents some amount of the smallest unit (e.g., the number of cents). These digital coins are stored in the public domain, where anyone has access. Therefore, coins need a security mechanism to prevent theft. Let there be a digital signature scheme that signs messages when its key card (secret signing key) is presented and the public key of the key card tells anyone whether it produced the signed message or not. Users attach these public keys to their digital coin(s) and securely keep the key card(s) with them. Wherever they want to spend the coins, they create a signed receipt, and they send the signed receipt to the network. Because the public key verifies the signed receipts without key cards, the whole network can verify that the legitimate owner is spending the coin. However, once the coins are sent, the new owner attaches a new public key to the coins where only he/she has a key card to it. In that way, after sending the coins, previous owners cannot take coins back because they do not have new key cards. The signed receipts are known as digital signatures.

Transparent Digital Coin Transactions. Users create transactions when they want to send digital coins or receive them. A transaction turns a set of current unspent coins (inputs) into spent coins and produces new unspent coins (outputs). However, a transaction should not illegally generate coins—the input coins should be equal to the output coins—or allow coins to be stolen. Therefore, each transaction has a **header** that proves the total coin amount summation (**1. summation proof**) and the digital signatures of coins (**2. ownership proofs**). In transparent coin transactions, the summation proof is simple and only contains the number of supplied coins (e.g., mining reward) and transaction fee because the verifiers can add the numbers and can check that input coins and the supplied coins are equal to the output coins and the transaction fee. However, the ownership proof is linear in the number

of unique public keys of inputs, i.e., the ownership proof of the transaction is the set of signatures of all unique input public keys refs. [16,17].

Confidential Digital Coins. Instead of plain-text coins, confidential payment systems keep coins' value hidden. We call them "*confidential coins*". A confidential coin is a commitment for the coin amount and a range proof to prove that the hidden coin amount is in the accepted range, e.g., $[0, 2^{64})$. A commitment can be viewed as a safe box that hides coins and is locked with a secret key (a.k.a., blinding key or the masking key). Yet, anyone can verify that the hidden coin amount is in the accepted range using the cryptographic range proof (even though proving the range of physical coins is impossible without opening the safe-box, it is actually possible with digital coin commitments!). These range proofs are called "zero-knowledge proofs" because they do not reveal anything other than the designated knowledge statement. In this case, the proofs only show that values are in the accepted range.

Confidential Transactions (CT). There are two categories of CTs; (1) Ring CTs which obfuscate the senders and receivers with shadow participants refs. [18–22] and (2) Aggregable CTs which allow aggregating multiple transactions into one transaction by removing spent coin records refs. [1,2,10–14]. In general, confidential transactions are similar to the transparent transactions but with the following main differences,

- they transact confidential coins, and
- their headers contain zero-knowledge summation proofs to show that hidden input coins are equal to hidden output coins, along with plain-text supplied coins and fees.

However, Ring CTs and aggregable CTs have *different* ownership proofs. Ring CTs' ownership proofs are threshold signatures or one/many-out-of-many proofs that are either linear or logarithmic in the number of unique participants, whereas, aggregable CTs' **reuse** their summation proofs as ownership proofs ref. [10]. Due to this reuse and removing spent coin records, aggregable CTs provide better scalability than Ring CTs. From now on, we use CT for aggregable CTs.

Ownership + Summation Proofs. Let there be an additively homomorphic commitment $u = \texttt{com}(v, k)$ that hides and binds (v, k) when v is the coin amount and k is the secret masking key. Here, "binding" means that it is computationally impossible to find another (v', k') pair for the same u, and "additively homomorphic" denotes that $\sum_{i=0}^{*} \texttt{com}(v_i, k_i) = \texttt{com}(\sum_{i=0}^{*} v_i, \sum_{i=0}^{*} k_i)$. A confidential coin is a commitment and a range proof of that commitment. For example, $C = (u, \pi = \texttt{range}_{2^L}(u))$, and π verifies that u's hidden coin amount is in $[0, 2^L)$.

Assume that a set of participants want to spend confidential coins $[u_i = \texttt{com}(v_i, k_i)]_{i=0}^{*}$ and create new confidential coins $[u'_i = \texttt{com}(v'_i, k'_i)]_{i=0}^{*}$ for the supplied coins s and the transaction fee f. Then they compute a public key out of commitments such that

$$pk = \sum_{i=0}^{*} u'_i - \sum_{i=0}^{*} u_i + \texttt{com}(f - s, 0) = \texttt{com}(0, \sum_{i=0}^{*} k'_i - \sum_{i=0}^{*} k_i)$$

because $\sum_{i=0}^{*} v'_i - \sum_{i=0}^{*} v_i + f - s = 0$ if the owners do not illegally generate coins.

Aggregable CTs' digital signature schemes use commitments as their public keys. However, a valid signature can be created *if and only if* the commitments' value is zero refs. [1,10–14]. In the previous equation, if participants do not generate coins illegally, they can create digital signatures for pk using the aggregated secret key $\sum_{i=0}^{*} k'_i - \sum_{i=0}^{*} k_i$. Hence a signature of pk can prove (1) the summation and (2) the ownership because a digital signature can only be created if all the secret keys are known.

Lattice-based Ownership+Summation Proofs. Ref. [1] introduced the first quantum-safe aggregable CT protocol. However, ref. [1] is inefficient leading to >30 KB commitments. Ref. [2] proposed the first practical aggregable post-quantum CT protocol, LACT, based on the Module-Ring Short Integer Problem (MSIS). LACT significantly improves efficiency by storing coins in their binary forms such that $u = \texttt{com}([b_0, ..., b_{63}], k)$ instead of the full integer, when $v = \sum_{i=0}^{63} 2^i b_i$. LACT's confidential integers are binary commitments and

a range proof for all bits, i.e., $C = (u, \pi = \texttt{range}_2(u))$ when π verifies that the bits of u's hidden coin amount are in $[0, 1]$. More concretely, LACT's commitments are only 9.6 KB long for coins in $[0, 2^{64})$.

However, summation does not work anymore because adding two binary commitments does not output a binary commitment due to carries. Therefore, LACT use carry proofs to fix the summation. Each carry proof has a commitment of carries and a range proof to show that each carry is either 0 or 1. Therefore, a LACT header contains supplied coins, transaction fee, a digital signature and a carry proof to prove the ownership and the summation of hidden coins.

Unprovable Immutability. In blockchains refs. [16,17], transaction headers are securely stored with consensus proofs to show that the community has accepted the that version of the blockchain, which is the "right blockchain". If somebody changes the right blockchain, the consensus proofs will indicate that there was an altercation, i.e., **immutability**. However, current aggregable CT protocols refs. [2,10–14] do not provide proper immutability proofs for aggregated blockchains, only for complete blockchains. Therefore, despite the aggregation, peers have to keep all coin records for the verification, e.g., Grin ref. [23] and Beam ref. [24]—which use Mimblewimble aggregable transactions refs. [10–13] keep all transactions to provide immutability. Therefore, aggregable CTs do not reduce the overall blockchain size nor provide trustless verification for aggregable CTs in current settings.

1.2. Our Contribution

We introduce LACT+, a more efficient Lattice-based Aggregable Confidential Transaction protocol with provable immutability. Furthermore, we implement a public LACT+ C library ref. [25] and provide formal security proofs.

Approx-SIS. LACT+ uses the Approximate Module Short Integer Problem (Approx-SIS) ref. [26] which can be tightly reduced to LWE (Learning with Errors ref. [27]) or SIS ref. [28]. Approx-SIS creates shorter proofs by dropping low bits of each polynomial coefficient without affecting security when the dropped bits are smaller than some norm, e.g., each LACT+ commitment drops 14 bits from 256×6 coefficients and reduces the size from 8.4 KB to 5.7 KB.

More than 2 inputs/outputs. LACT's main constraint was that a maximum of two inputs or outputs could be included in a transaction because the carries must be in [0, 1]. If there are more than two inputs/outputs, they had to be separated into multiple transactions, and each transaction includes an additional carry proof. LACT+ transactions allow having more than two inputs or outputs at a logarithmic cost, i.e., $\mathcal{O}(\log(x))$ for x number of in/outputs. Hence, LACT+ does not unnecessarily increase the size of the header.

Trustless Immutability. Previous aggregable CTs' headers are malleable, or the owners of the transactions can use the same header set for different unspent coins. This allows the owners to forge many chains for the same header with different unspent coins or spent coins refs. [10,13]. Hence, preserving headers is not sufficient to provide immutability, and block creators hash the whole transaction and save the transaction hash through the consensus mechanism. Therefore, the peers must download the complete chain (note that we do not consider users who depend on "trusted" peers **without** verifying the blockchain as peers because they are replicas of the those "trusted" peers). Hence, except for the efficient ownership + summation proofs, the aggregation has no use to the peers who do not blindly trust other peers.

As a solution, LACT+ headers contain aggregable activity proofs ref. [29] to prevent the forging of different blockchains for the consensus accepted headers, i.e., each transaction header contains an activity proof for that transaction, and by giving the same input and output coin records, the activity proof verifies that they existed. More importantly, an aggregated activity proof can verify the activities of an aggregated CT; hence, chains' unspent coin records cannot be forged. Not only that, activity proofs are constant sizes for any number of inputs and outputs, e.g., 49 bytes in LACT+. Hence, activity proofs do not

break transaction aggregation, but at a tiny cost, peers can download a small aggregated chain and verify the existence of coins without blindly trusting other peers.

Table 1 compares LACT+ with other CT protocols.

Table 1. Comparison of Confidential Transactions.

Implementation	Post Quantum	Security	Aggregable	Immutable Headers
Ring CT ref. [30]	✗	DLP	✗	-
Maxwell's CT ref. [31]	✗	DLP	✗	-
Mimblewimble ref. [10]	✗	DLP	✓	✗
Ring CT v2 ref. [19]	✗	DLP	✗	-
Fuchsbauer et al. ref. [13]	✗	DLP	✓	✗
Lattice Ring CT ref. [21]	✓	SIS	✗	-
MATRiCT ref. [20]	✓	MSIS	✗	-
Zhang et al. ref. [1]	✓	SIS	✓	✗
MATRiCT+ ref. [22]	✓	MSIS/LWE	✗	-
LACT ref. [2]	✓	SIS/MSIS	✓	✗
Ours (LACT+)	✓	Approx-SIS	✓	✓

2. Preliminary

An integer ring is $\mathbb{Z}/q\mathbb{Z} = \mathbb{Z}_q$ which has integers in $[-\frac{q-1}{2}, \frac{q-1}{2}]$ for an odd prime q. A polynomial ring is $\mathbb{Z}[X]/(X^N+1) = \mathcal{R}$ with $N = 2^k$ for some integer $k > 0$. A fully splitting ring is $\mathbb{Z}_q[X]/(X^N+1) = \mathcal{R}_q$ where each polynomial coefficient is in $\mathbb{Z}_q$ for a prime $q = 1 \pmod{2N}$. A polynomial is denoted with an upper arrow, e.g., $\vec{a}$ that is in $\mathcal{R}$ or $\mathcal{R}_q$. We denote vectors of n elements in bold letters ($\boldsymbol{a} = [a_i]_{i=0}^{n}$) and matrices of m vectors in capital bold letters ($\boldsymbol{A} = [\boldsymbol{a}_i]_{i=0}^{m}$). Similarly, polynomial vectors are denoted as $\vec{\boldsymbol{a}} = [\vec{a}_i]_{i=0}^{n}$, and matrices are denoted as $\vec{\boldsymbol{A}} = [\vec{\boldsymbol{a}}_i]_{i=0}^{m}$. $\vec{a}\vec{b}$ denotes the polynomial multiplication of $\vec{a}$ and $\vec{b}$. $x\vec{a}$ is a scalar multiplication where each coefficient is multiplied by scalar x.

Uniform sampling of some set S is denoted as $s \xleftarrow{\$} S$. A negligible function of security parameter λ is denoted as $\epsilon(\lambda) = 1/o(\lambda^c)$ for some natural number c. The security parameter and public parameters are commonly referred as λ and $pp_{(\lambda)}$. We only use the following norms: the infinity norm $\|\vec{a}\| = \texttt{max}([|a_i|]_{i=0}^{N-1})$ and the level 1 norm $\|\vec{a}\|_1 = \sum_{i=0}^{N-1} |a_i|$. $\boldsymbol{0}$ and $\vec{\boldsymbol{0}}$ are a vector of all zero values and a zero polynomial of all zero coefficients, respectively. $\texttt{binary}(v)$ either outputs $\boldsymbol{b}$ of L elements or $\vec{b} \in \mathcal{R}_q$ such that $v = \sum_{i=0}^{L} 2^i b_i$ according to the context. $\texttt{rot}(v_i, i)$ outputs polynomial $\vec{a}$ such that the i^{th} coefficient of $\vec{a}$ is v_i, and other coefficients are zeros s.t. $[\vec{a}_j = 0]_{j=0, j\neq i}^{N}$. $\texttt{highbits}_p(v)$ and $\texttt{lowbits}_p(v)$ denote high bits and low bits of $v \in \mathbb{Z}_q$ such that $\texttt{highbits}_p(v) \cdot 2^p + \texttt{lowbits}_p(v) = v$. Furthermore, $\texttt{up}_p(v) = vs. \cdot 2^p$. We use these functions for polynomial vectors in an element-wise manner, e.g., $\texttt{highbits}_p(\vec{\boldsymbol{a}})$ to denote high-bit polynomial vectors. Note that we use polynomial rotations to change the locations of bits. For example,

$$(j,i) \geq 0; j{+}i < N : \texttt{rot}(v,j)\texttt{rot}(1,i){=}\texttt{rot}(v,j{+}i) \in \mathcal{R}$$

$$(j,i){\geq}0; j{-}i{>}0; i{<}N : \texttt{rot}(v,j)\texttt{rot}(-1,N{-}i) = \texttt{rot}(v,j{-}i)$$

2.1. Common Security Properties

First, we start with the common security definitions of confidential coins, carry proofs, and transactions, i.e., the hiding property, zero-knowledge, and knowledge soundness.

Confidential coins and carry proofs should be hiding, i.e., they should not reveal the committed coin value or the carry vector. We define hiding or indistinguishability of a generic protocol $\texttt{P}$'s outputs below.

Definition 1 (Statistical Hiding)**.** *Let $\mathcal{D}_1$ be the output distribution of protocol $\mathtt{P}(pp_\lambda)$, and $\mathcal{D}_0$ be the simulated distribution created by a simulator $\mathcal{S}(pp_\lambda, \mathcal{T})$ with a trapdoor $\mathcal{T}$ of public parameter pp_λ. For any $\mathcal{A}$, $\mathtt{P}$'s outputs are hiding if*

$$2\left|\tfrac{1}{2} - Pr\left[b \stackrel{?}{=} b' | b \xleftarrow{\$} [0,1]; \theta \xleftarrow{\$} D_b; b' \leftarrow \mathcal{A}(pp_\lambda, \theta)\right]\right| \leq \epsilon(\lambda).$$

Coin and carry proofs contain range proofs. A range proof should not reveal anything about the committed value other than the range. Similarly, summation proofs of transactions should not disclose anything about the coin values other than that input coins are equal to output coins. We define these requirements for a generic protocol $\mathtt{P}$ and statement(s) $\mathcal{L}$. For example, in range proofs, $\mathtt{range}(L, \boldsymbol{b})$ outputs 1 if all $b_i \in [0,1]$ of $i \in [0, L)$. The statement $\mathcal{L}$ of a range proof is "$\mathtt{range}(L, \boldsymbol{b})$ is 1 for hidden $\boldsymbol{b}$". Assume the generic protocol $\mathtt{P}$ reveals a statement $\mathcal{L}$ about the outputs but nothing else. Then, $\mathtt{P}$'s outputs are zero-knowledge except for $\mathcal{L}$. We state the zero-knowledge of $\mathtt{P}$ below.

Definition 2 (Zero-Knowledge Argument)**.** *Let $\mathcal{D}_0$ be the simulated distribution of a simulator $\mathcal{S}(pp_\lambda, \mathcal{L}, \mathcal{T})$ with a trapdoor $\mathcal{T}$ of public parameter pp_λ for $\mathcal{L}$, and $\mathcal{D}_1$ be the real output distribution of protocol $\mathtt{P}(pp_\lambda)$ for statements $\mathcal{L}$. For any p.p.t. $\mathcal{A}$, $\mathtt{P}$'s outputs are zero-knowledge if*

$$2\left|\tfrac{1}{2} - Pr\left[b \stackrel{?}{=} b' | b \xleftarrow{\$} [0,1]; \theta \xleftarrow{\$} D_b; b' \leftarrow \mathcal{A}(pp_\lambda, \theta, \mathcal{L})\right]\right| \leq \epsilon(\lambda).$$

Even though coins, carry proofs, and transactions are hiding and zero-knowledge, they should be knowledge sound such that no adversary can create valid proofs for invalid hidden coin amounts or invalid carry vectors. We capture this property by using a simulator $\mathcal{K}$ with a trapdoor $\mathcal{T}$ who extracts hidden values from the proofs. This is a stronger version of knowledge soundness, called *simulated witness extractability*. If a generic protocol is simulated witness extractable, the simulator $\mathcal{K}$ can always extract valid hidden amounts and carries that satisfy the required statement(s). We state the simulated witness extraction of a generic protocol $\mathtt{P}$ for statement(s) $\mathcal{L}$ below.

Definition 3 (Simulated Witness Extractability)**.** *Let $\mathtt{P}$ be a generic zero-knowledge protocol for statement $\mathcal{L}$ such that $\mathtt{P}(pp_\lambda, \mathcal{L}, \Pi)$ verifies whether hidden values of Π satisfies $\mathcal{L}$ or not. $\mathtt{P}$ is simulated witness extractable if*

$$Pr\left[\begin{matrix} \mathtt{P}(pp_\lambda, \mathcal{L}, \Pi) \stackrel{?}{=} 1 \wedge \\ \mathcal{L}(pp_\lambda, \boldsymbol{w}) \stackrel{?}{=} 1 \end{matrix} \middle| \begin{matrix} \Pi \leftarrow \mathcal{A}(pp_\lambda) \\ \boldsymbol{w} \leftarrow \mathcal{K}(pp_\lambda, \mathcal{T}, \mathcal{L}, \Pi) \end{matrix}\right] \leq \epsilon(\lambda).$$

Technically, these extractors do not exist outside the simulated world because public parameters are securely generated to avoid unintentional trapdoors.

2.2. Hardness Assumption

LACT+ depends on the approximate SIS problem ref. [26].

Definition 4 (Approximate Inhomogeneous Module Short Integer Solution Problem (Approx-SIS))**.** *The advantage of an algorithm $\mathcal{A}$ solving Approx-SIS of $(n, m, q, \gamma, \gamma', N)$ after one execution is given by,*

$$Pr\left[\begin{matrix} \|\vec{\boldsymbol{s}}\| \leq \gamma \wedge \\ \|\vec{\boldsymbol{H}}\vec{\boldsymbol{s}} - \vec{\boldsymbol{y}}\| \leq \gamma' \end{matrix} \middle| \begin{matrix} \vec{\boldsymbol{H}} \xleftarrow{\$} \mathcal{R}_q^{n \times m}; \vec{\boldsymbol{y}} \xleftarrow{\$} \mathcal{R}_q^n \\ \vec{\boldsymbol{s}} \leftarrow \mathcal{A}(pp_\lambda, \vec{\boldsymbol{y}}, \vec{\boldsymbol{H}}) \end{matrix}\right]$$

2.3. Activity Proofs

Let $\mathbb{H}$ be a multiplicative group of prime order $q_2 > 2^{3\lambda}$ and $\mathtt{H} : \{0,1\}^* \to \mathbb{H}$ be a secure collision-resistant hash function family. Casually, we use $\mathtt{H}$ for one randomly chosen but commonly known member of a collision-resistant hash function family for a given λ where the key of $\mathtt{H}$ is random, fixed, and publicly known.

Let there be a non-empty *set* of data entries $\boldsymbol{s}$ that will be updated into a new *set* $\boldsymbol{s}'$. The proof δ of this activity is,

$$\delta = \texttt{activity}(\boldsymbol{s}, \boldsymbol{s}') : \prod_{i=0}^{*} \texttt{H}(s_i') \cdot (\prod_{i=0}^{*} \texttt{H}(s_i))^{-1} \in \mathbb{H}.$$

Definition 5 (Immutability)**.** *An activity proof of* $(\texttt{H}, q')$ *is immutable if the following is less than or equal to* $\epsilon(\lambda)$*.*

$$Pr\left[\begin{array}{r} (s_1, s_1') \stackrel{?}{\neq} (s_2, s_2') \wedge \\ \texttt{activity}(s_1, s_1') \stackrel{?}{=} \texttt{activity}(s_2, s_2') \end{array} \middle| \begin{pmatrix} s_1, s_1' \\ s_2, s_2' \end{pmatrix} \leftarrow \mathcal{A}(\texttt{H}, q_2)\right]$$

2.4. Hints

We frequently multiply high-bit polynomials with short challenge polynomials and check equality. Because the multiplication creates large errors that may propagate into higher bits, we use "hints", a $\{-1, 0, 1\}$ polynomial vector that holds at most χ recovery bits. The way of creating and using hints is stated below.

$\texttt{hints}(\chi, \vec{\boldsymbol{a}} \in \mathcal{R}_q^n, \vec{\boldsymbol{b}} \in \mathcal{R}_q^n)$:
$\vec{\boldsymbol{h}} = \vec{\boldsymbol{a}} - \vec{\boldsymbol{b}} \in \mathcal{R}_q^n$
if $\|\vec{\boldsymbol{h}}\| > 1 \wedge \|\vec{\boldsymbol{h}}\|_1 > \chi$: return $\perp$
return $\vec{\boldsymbol{h}}$

$\texttt{use\_hints}(\chi, \vec{\boldsymbol{a}} \in \mathcal{R}_q^n, \vec{\boldsymbol{h}} \in \mathcal{R}_q^n)$:
if $\|\vec{\boldsymbol{h}}\| > 1 \wedge \|\vec{\boldsymbol{h}}\|_1 > \chi$:
return $\perp$
return $\vec{\boldsymbol{b}} = \vec{\boldsymbol{a}} - \vec{\boldsymbol{h}} \in \mathcal{R}_q^n$

2.5. Public Parameters

We use the following public parameters (Algorithm 1) throughout the remainder of the paper.

Algorithm 1 Public Parameter Generation

Assign $\mathcal{C}_\beta^N, L', p_1, p_2, \chi, \alpha, \tau, \tau_1, \tau_2, \tau_3 \in \mathbb{N}$ such that
$L'L \leq N$ ▷ Maximum number of inputs/outputs is $2^{L'}$ because we only used one polynomial for carry commitments. When using l number of polynomials, this constraint should be $L'L \leq Nl$.
▷ for negligible soundness error
$\log_2 \binom{N}{\beta} + \beta \geq 2\lambda$,
▷ for smaller errors targeting different summations
$p_1 \cdot 2^{16} < 2^{\gamma'}$ and $p_2 \cdot 2^8 < 2^{\gamma'}$
▷ Norm 1 limit for hint polynomials is heuristically chosen according to p_1 and p_2 aiming to create hints within a reasonable time
$\chi \leftarrow \texttt{heuristic}(p_1, p_2)$ s.t. $\|\texttt{hint}(\cdot) \in \mathcal{R}_q^n\|_1 \leq \chi$
▷ for rejection sampling in range proofs
$0 < 2 \ll \alpha,\ 0 < \beta^2\tau + \beta\tau_1 \ll \tau_2 \leq \gamma$,
▷ for rejection sampling in aggregate signatures
$0 < (2^{L'} + 1)\beta\tau \ll \tau_3$ and $2^{L'}\tau_3 \leq \gamma$,
▷ for computational hiding of coin amounts
$(m-3)N\log_2(2\tau) \geq 6\lambda$,
$L\alpha \ll \gamma$
Get $\vec{\boldsymbol{H}} \stackrel{\$}{\leftarrow} \mathcal{R}_q^{n \times m}$ s.t. $\texttt{Approx} - \texttt{SIS}_{n,m,q,N,\gamma,\gamma'}$ is hard.
$\mathcal{C}_\beta^N = [x \in \mathcal{R}_q \text{ s.t. } \|x\| = 1, \|x\|_1 = \beta]$
return $pp_{(\lambda,L)} = (n, m, q, N, \mathcal{C}_\beta^N, L', p_1, p_2, \chi, \alpha, \tau, \tau_1, \tau_2, \tau_3, \gamma, \gamma', \vec{\boldsymbol{H}})$

2.6. Fiat–Shamir Signatures

We define a digital signature scheme based on Fiat–Shamir challenges similar to refs. [32–36] in Algorithm 2. This signature scheme is the Approx-SIS version of ref. [36]

with (1) a varying key space to allow aggregate keys and (2) beginning zero polynomials to align with commitments.

SIG provides completeness and strong EUF-CMA (Existential Unforgeability under Chosen Message Attack) due to the security of ref. [36] and the hardness of Approx-SIS.

We define the security properties below.

Definition 6. *SIG is complete and strong EUF-CMA if*

$$Pr\left[\mathit{SIG.ver}(pp_\lambda, pk, m, \mathtt{sig}, c) \stackrel{?}{=} 1 \middle| \begin{matrix}(k, pk) = \mathit{SIG.kgen}(pp_\lambda, c) \\ \mathtt{sig} = \mathit{SIG.sign}(pp_\lambda, k, m, c)\end{matrix}\right] \geq 1 - \epsilon(\lambda)$$

$$Adv^{EUF,\mathcal{A}}_{SIG,pp_\lambda} := Pr\left[Game^{EUF,\mathcal{A}}_{SIG,pp_\lambda}()\right] \leq \epsilon(\lambda).$$

$\underline{\mathsf{Game}^{\mathsf{EUF},\mathcal{A}}_{\mathsf{SIG},pp_\lambda}:}$
$c \leftarrow \mathcal{A}_{step1}(pp_\lambda)$; **if** $c > 2^{L'}$: return $\perp$
$(k, pk) = \mathsf{SIG.kgen}(pp_\lambda, c)$; $(m', \mathtt{sig}') \leftarrow \mathcal{A}^{\mathtt{sign}_k}_{step2}(pp_\lambda, pk)$
return $(m', \mathtt{sig}') \notin \mathbf{Q} \wedge \mathsf{SIG.ver}(pp_\lambda, pk, m', \mathtt{sig}', c)$

$\underline{\mathsf{Oracle}\ \mathtt{sign}_k(m, c): \mathtt{sig} = \mathsf{SIG.sign}(pp_\lambda, k, m, c)}$
$\mathbf{Q} = \mathbf{Q} \cup \{(m, \mathtt{sig})\}$; return sig

Theorem 1. SIG *is complete and EUF-CMA when ref. [36] is complete and EUF-CMA, and solving Approx-SIS of* $(n, m-3, q, \gamma, \gamma', N)$ *is hard.*

Algorithm 2 Digital Signatures

1: ▷ Here, the key spaces change according to $c \leq 2^{L'}$
2: ▷ Recall that $\tau_2 \gg (2^{L'}+1)\tau\beta$ and $\gamma \leq 2^{L'}\tau_3$
3: $\underline{\mathsf{SIG.kgen}(pp_\lambda, c):}$ ▷ $\vec{k}$ is the secret signing key
4: $\vec{k} \xleftarrow{\$} [-c\tau, c\tau]^{(m-3)\times N}$; $\vec{pk} = \mathtt{highbits}_{p_1}(\vec{H}[\mathbf{0}^3, \vec{r}]) \in \mathcal{R}_q^n$
5: return $(\vec{k}, \vec{pk})$

6: $\underline{\mathsf{SIG.sign}(pp_\lambda, \vec{k}, m, c):}$ if $c > 2^{L'}$: return $\perp$
7: $\vec{pk} = \mathtt{highbits}_{p_1}(\vec{H}[\mathbf{0}^3, \vec{r}]) \in \mathcal{R}_q^n$
8: $\vec{r}_0 \xleftarrow{\$} [-c\tau_3, c\tau_3]^{(m-3)\times N}$
9: $\vec{y} = \mathtt{highbits}_{\gamma'}(\vec{pk}) \in \mathcal{R}_q^n$
10: $\vec{x} = \mathtt{hash}(\vec{pk}, \vec{y}, m) \in \mathcal{C}_\beta^N$
11: $\vec{\sigma} = \vec{r}_0 + \vec{x}\vec{k} \in \mathcal{R}^{m-3}$
12: **if** $\|\vec{\sigma}\| > (c\tau_3 - c\beta\tau)$: go to Step 8
13: $\vec{y}' = \mathtt{highbits}_{\gamma'}(\vec{H}[\mathbf{0}^3, \vec{\sigma}] - \vec{x}\mathrm{up}_{p_1}(\vec{pk})) \in \mathcal{R}_q^n$
14: $\vec{h} = \mathtt{hints}(\chi, \vec{y}', \vec{y})$; **if** $\vec{h} = \perp$: go to Step 8
15: return $(\vec{\sigma}, \vec{h}, \vec{x})$

16: $\underline{\mathsf{SIG.ver}(pp_\lambda, \vec{pk}, m, \vec{\sigma}, \vec{x}, c):}$
17: if $c > 2^{L'}$: return 0
18: $\vec{y}' = \mathtt{highbits}_{\gamma'}(\vec{H}[\mathbf{0}^3, \vec{\sigma}] - \vec{x}\mathrm{up}_{p_1}(\vec{pk})) \in \mathcal{R}_q^n$
19: $\vec{y} = \mathtt{hints}(\chi, \vec{y}', \vec{h})$; **if** $\vec{y} = \perp$: return 0
20: return $\|\vec{\sigma}\| < \gamma \wedge \vec{x} \stackrel{?}{=} \mathtt{hash}(\vec{pk}, \vec{y}, m)$

3. LACT+ Protocol

A LACT+ transaction has input confidential coins, output confidential coins, and a header (signature, carry proof, and activity proof), as shown in Figure 1. The input confidential coins are in the blockchain's set of unspent coins. During the aggregation, they will be removed from the unspent coin set, and output confidential coins will be added to the unspent coin set (see Figure 2) where $\texttt{coin(s)}$ was removed). Therefore, an aggregated blockchain can be seen as a big transaction that contains all unspent coins. In this section, we explain confidential coins, carry proofs of the headers, and finally the complete aggregable transactions.

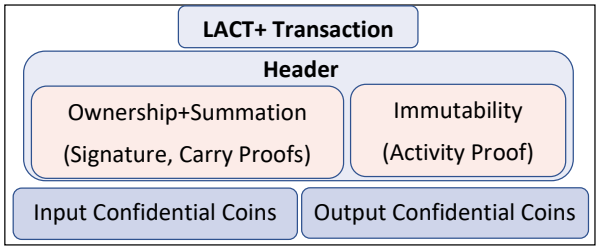

Figure 1. LACT+ transaction structure.

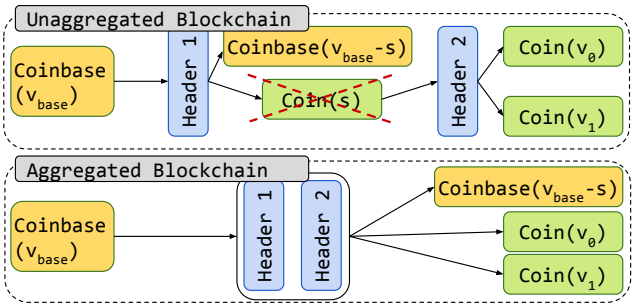

Figure 2. Transaction aggregation.

3.1. Confidential Coins

We begin by building confidential coin records where we can hide coin values but can also verify that the hidden coin amounts are in some range $[0, 2^L)$.

Let there be a confidential coin scheme $\texttt{COIN}$ that supports the following functionalities.

- $\texttt{COIN.gen}(pp_{(\lambda,L)}, v)$: returns $(k, \texttt{coin})$ if v is in $[0, 2^L - 1]$; otherwise returns $\perp$. Here, k is a secret key, a.k.a., blinding key, and L is defined in $pp_{(\lambda,L)}$.
- $\texttt{COIN.open}(pp_{(\lambda,L)}, v, k, \texttt{coin})$: returns 1 if $\texttt{coin}$ is generated for $(v \in [0, 2^L), k)$; otherwise, returns 0.
- $\texttt{COIN.ver}(pp_{(\lambda,L)}, \texttt{coin})$: returns 1 if the hidden coin amount is in $[0, 2^L)$ otherwise, returns 0.

Protocol Sketch

First, we sketch an interactive version of a 1-bit range proof. Let there be a value b which should be either 0 or 1. For some mask $a \in [-\alpha, \alpha]$ and random challenge $x \in [-1, 0, 1]$, take $z = bx + a$ if $z \in [-\alpha + 1, \alpha - 1]$. Otherwise, the process is repeated until a valid z is found. Then $z(z - x) = xa(2b - 1) + a^2$ because $x^2 b(b - 1)$ should be zero. We use this method to prove the range of a bit.

Let $\mathcal{P}$ be a prover and $\mathcal{V}$ be a verifier. $\mathcal{V}$ has a commitment $u = \texttt{com}(b, 0, k)$, and $\mathcal{P}$ who knows $k \in [-\tau, \tau]$ wants to show that the bit b is either 1 or 0 without revealing it. The prover uses commitments to prove the range of b. First, $\mathcal{P}$ creates commitment $t_1 = \texttt{com}(b, a(2b - 1), r_1)$ and sends it to $\mathcal{V}$. After receiving a challenge x_1 from $\mathcal{V}$, $\mathcal{P}$ replies with $t_2 = \texttt{com}(xa, a^2, r_2)$. Then, $\mathcal{V}$ sends the second challenge x_2. After that, $\mathcal{P}$ computes $z = (bx + a)$ and $r = x_2(x_1 k + r_1) + r_2$. If z is in $[-\alpha+1, \alpha-1]$ and r is in

$[-\tau+\tau_1+\tau_2, \tau-\tau_1-\tau_2]$, they statistically hide b, a, k, r_1, and r_2. Hence, $\mathcal{P}$ sends z and r to $\mathcal{V}$. If z and r are not in the secure ranges, $\mathcal{P}$ rejects them and restarts the protocol. Finally, the verifier $\mathcal{V}$ checks that $\texttt{com}(x_1 z, z(z-x_2), r) \stackrel{?}{=} x_2 x_1 u + x_2 t_1 + t_2$. $\mathcal{V}$ accepts that b is either 0 or 1 if they are equal.

We use this technique to prove the range of the coin by repeating it for all bits. Furthermore, we increase the range of challenges into $2^{2\lambda}$ so that $\mathcal{P}$ cannot cheat. The complete non-interactive protocol `COIN` is given in Algorithm 3, and a graphical overview is provided in Figure 3. Note that all LACT+ commitments have the form of $\texttt{com}(*,*,*,*)$, not $\texttt{com}(*,*,*)$. Hence, the verifier will check $\texttt{com}(x_1 z, z(z-x_2), 0, r) \stackrel{?}{=} x_2 x_1 u + x_2 t_1 + t_2$ with a fixed 0 value. This extra value space is used for carry proofs which will be explained in the next section.

One-bit Range Proof Sketch (Interactive)

$\mathcal{P}$		$\mathcal{V}_2$
$a \xleftarrow{\$} [-\alpha, \alpha]; r_1 \xleftarrow{\$} [-\tau_1, \tau_1]$		
$t_1 = \texttt{com}(b, a(2b-1), r_1)$	$\xrightarrow{t_1}$	
$r_2 \xleftarrow{\$} [-\tau_2, \tau_2]$	$\xleftarrow{x_1}$	$x_1 \xleftarrow{\$} [-1, 1]$
$t_2 = \texttt{com}(xa, a^2, r_2)$	$\xrightarrow{t_2}$	
$z = (bx + a) \in [-\alpha+1, \alpha-1]$	$\xleftarrow{x_2}$	$x_2 \xleftarrow{\$} [-1, 1]$
$r = x_2(x_1 k + r_1) + r_2$ *s.t.*		
$r \in [-\tau+\tau_1+\tau_2, \tau-\tau_1-\tau_2]$	$\xrightarrow{z, r}$	$\texttt{com}(x_1 z, z(z-x_2), r) \stackrel{?}{=}$
		$x_2 x_1 u + x_2 t_1 + t_2$

$$\vec{H}\underbrace{\begin{bmatrix} \vec{x}_1 \sum_{i=0}^{L} \vec{z}_i \\ \hat{\hat{z}} \\ \vec{0} \\ \vec{r} \end{bmatrix}}_{\texttt{set\_c}(\vec{s})} = \vec{x}_2 \vec{x}_1 \vec{H} \underbrace{\begin{bmatrix} \hat{\hat{b}} \\ \vec{0} \\ \vec{0} \\ \vec{k} \end{bmatrix}}_{\vec{u}} + \vec{x}_2 \vec{H} \underbrace{\begin{bmatrix} \vec{0} \\ \sum_{i=0}^{L} \vec{a}_i \times \texttt{rot}(2b_i - 1, i) \\ \vec{0} \\ \vec{r}_1 \end{bmatrix}}_{\vec{t}_1} + \vec{H} \underbrace{\begin{bmatrix} \vec{x}_1 \sum_{i=0}^{L} \vec{a}_i \\ \sum_{j=0}^{L-1} \vec{a}_j^2 \\ \vec{0} \\ \vec{r}_2 \end{bmatrix}}_{\vec{t}_2}$$

Figure 3. A graphical overview of confidential coins (Algorithm 3).

Algorithm 3 New Confidential Coins

▷ Creates coin records for v coins
$\texttt{COIN.gen}(pp_{(\lambda,L)}, v)$:
▷ Short vector for the commitment
$\vec{b} = \texttt{binary}(L, v)$ s.t. $v = \sum_{i=0}^{L-1} 2^i b_i$
$\vec{k} \xleftarrow{\$} [-\tau, \tau]^{(m-3)\times N}$
$\vec{s} = [\vec{b}, \vec{0}, \vec{0}, \vec{k}] \in \mathcal{R}_q^m$
$\vec{u} = \texttt{highbits}_{p_1}(\vec{H}\vec{s}) \in \mathcal{R}_q^n$ ▷ Commitment
▷ Get masking values for bits from Lazy Sampling
$[\vec{a}_i \xleftarrow{\$} [-(\alpha - (1 - b_i)), (\alpha - (1 - b_i))]^N]_{i=0}^L$
▷ Short key vectors for range proof commitments
$\vec{r}_1 \xleftarrow{\$} [-\tau_1, \tau_1]^{(m-3)\times N}; \vec{r}_2 \xleftarrow{\$} [-\tau_2, \tau_2]^{(m-3)\times N}$
$\vec{s}_1 = [\vec{0}, \sum_{i=0}^{L} \vec{a}_i \texttt{rot}(2b_i - 1, i), \vec{0}, \vec{r}_1] \in \mathcal{R}^m$
$\vec{t}_1 = \texttt{highbits}_{p_2}(\vec{H}\vec{s}_1) \in \mathcal{R}_q^n$
$\vec{x}_1 = \texttt{hash}(\vec{u}, \vec{t}_1) \in \mathcal{C}_\beta^N$
$\vec{s}_2 = [\vec{x}_1 \sum_{i=0}^{L} \vec{a}_i, \sum_{i=0}^{L} \vec{a}_i \vec{a}_i, \vec{0}, \vec{r}_2] \in \mathcal{R}^m$
$\vec{t}_2 = \texttt{highbits}_{\gamma'}(\vec{H}\vec{s}_2) \in \mathcal{R}_q^n$
$\vec{x}_2 = \texttt{hash}(\vec{u}, \vec{t}_1, \vec{t}_2) \in \mathcal{C}_\beta^N$
$[\vec{z}_i = \vec{x}_2 \texttt{rot}(b_i, i) + \vec{a}_i]_{i=0}^L \in \mathcal{R}^L$
$\vec{r} = \vec{x}_2(\vec{x}_1\vec{k} + \vec{r}_1) + \vec{r}_2 \in \mathcal{R}^{m-3}$
if $\|\vec{z}\| > (\alpha - 1) \vee \|\vec{r}\| > (\tau_2 - \beta^2\tau - \beta\tau_1)$:
  go to Step 5
▷ Check norms
$\hat{z} = \sum_{i=0}^{L} \vec{z}_i(\vec{z}_i - \vec{x}_2 \texttt{rot}(1, i)) \in \mathcal{R}$
if $\|\hat{z}\| > \gamma$: go to Step 5
▷ Create hints for $\vec{t}_2$
$\vec{s} = [\vec{x}_1 \sum_{i=0}^{L} \vec{z}_i,\ \hat{z},\ \vec{0},\ \vec{r}] \in \mathcal{R}^m$
$\vec{t}_2' = \texttt{highbits}_{\gamma'}(\vec{H}\vec{s} - \vec{x}_2(\vec{x}_1 \texttt{up}_{p_1}(\vec{u}) + \texttt{up}_{p_2}(\vec{t}_1))) \in \mathcal{R}_q^n$
$\vec{h} = \texttt{hints}(\chi, \vec{t}_2', \vec{t}_2)$
if $\vec{h} = \perp$: go to Step 5
return $\vec{k}$ and $\texttt{coin} = (\vec{z}, \vec{r}, \vec{u}, \vec{t}_1, \vec{h}, \vec{x}_2)$

▷ Open coin records of v coins and key $\vec{k}$
$\texttt{COIN.open}(pp_{(\lambda,L)}, v, \vec{k}, \texttt{coin})$:
$(\vec{z}, \vec{r}, \vec{u}, \vec{t}_1, \vec{h}, \vec{x}_2) = \texttt{coin}$
$\vec{s} = [\texttt{binary}(v), \vec{0}, \vec{0}, \vec{k}] \in \mathcal{R}^m$
return $\|\vec{s}\| \overset{?}{\leq} \alpha \wedge \vec{u} \overset{?}{=} \texttt{highbits}_{p_1}(\vec{H}\vec{s}) \in \mathcal{R}_q^n$

▷ Verify coin records
$\texttt{COIN.ver}(pp_{(\lambda,L)}, \texttt{coin})$:
$(\vec{z}, \vec{r}, \vec{u}, \vec{t}_1, \vec{h}, \vec{x}_2) = \texttt{coin}$
$\vec{x}_1 = \texttt{hash}(\vec{u}, \vec{t}_1)$
$\hat{z} = \sum_{i=0}^{L} \vec{z}_i(\vec{z}_i - \vec{x}_2 \texttt{rot}(1, i)) \in \mathcal{R}$
$\vec{s} = [\vec{x}_1 \sum_{i=0}^{L} \vec{z}_i,\ \hat{z},\ \vec{0},\ \vec{r}] \in \mathcal{R}^m$
▷ Use hints to recompute $\vec{t}_2$
$\vec{t}_2' = \texttt{highbits}_{\gamma'}(\vec{H}\vec{s} - \vec{x}_2(\vec{x}_1 \texttt{up}_{p_1}(\vec{u}) + \texttt{up}_{p_2}(\vec{t}_1))) \in \mathcal{R}_q^n$
$\vec{t}_2 = \texttt{use\_hints}(\chi, \vec{t}_2', \vec{h})$
if $\vec{t}_2 = \perp$: return 0
return $\vec{x}_2 \overset{?}{=} \texttt{hash}(\vec{u}, \vec{t}_1, \vec{t}_2) \wedge \|\vec{s}\| \leq \gamma$

We want COIN to be complete, hiding, binding, zero-knowledge, and knowledge sound. Here, we state the undefined security properties of COIN; completeness and binding. Other properties, namely, hiding, zero-knowledge, and knowledge soundness, defined in Section 2.1 are valid for COIN as well when the statement $\mathcal{L}$ is

"all L bits of $\boldsymbol{b} = \texttt{binary}(L, v)$ are either 0 or 1".

Definition 7 (Completeness)**.** COIN *is complete if honestly generated proofs can be verified for the correct range, or the following is greater than or equal to* $1 - \epsilon(\lambda)$.

$$Pr\left[\texttt{COIN.ver}(pp_{(\lambda,L)}, \texttt{coin}) \;\middle|\; \begin{matrix} vs. \in [0, 2^L - 1] \\ (k, \texttt{coin}) = \texttt{COIN.gen}(pp_{(\lambda,L)}, v) \end{matrix}\right]$$

Once a coin record has been published, no one, including the creator who knows the secret key, should be able to claim another coin value for the same coin record. Otherwise, after sending a transaction, owners can illegally generate coins. We capture this security property below.

Definition 8 (Binding)**.** COIN *is binding if*

$$Pr\left[\begin{matrix} (v,k) \stackrel{?}{\neq} (v',k') \wedge \\ \texttt{COIN.open}(pp_{(\lambda,L)}, v, k, \texttt{coin}) \wedge \\ \texttt{COIN.open}(pp_{(\lambda,L)}, v', k', \texttt{coin}) \end{matrix} \middle| \begin{pmatrix} \texttt{coin} \\ v, k \\ v', k' \end{pmatrix} \leftarrow \mathcal{A}(pp_{(\lambda,L)}) \right] \leq \epsilon(\lambda)$$

Theorem 2. COIN *is complete, computationally binding, and computational knowledge sound, and zero-knowledge if approx-SIS of* $(n, m, q, \gamma, \gamma', N)$ *is hard (see the security proof in Appendix A).*

3.2. Logarithmic-Sized Carry Proofs

Binary commitments are smaller than full integer commitments. However, we cannot directly check summations of transactions' in/outputs and chains' total unspent coin amount when they are in binary forms. Therefore, LACT ref. [2] uses **static carries** with maximum two inputs and two outputs and a **static supply**.

Unlike traditional blockchains, LACT maintains an unspent coin record (with a zero key) for the coinbase (coin supplier) where the coinbase holds future supply, and those coins do not belong to anyone until they are awarded. The idea is to maintain a *constant coin amount* on the blockchain all the time. For instance, the coinbase holds the maximum $v_{\texttt{base}} = S$ at the beginning and keeps reducing the amount $v_{\texttt{base}} = v_{\texttt{base}} - s^{(t)}$ with each rewarded coin $s^{(t)}$ of transaction t. Therefore, the blockchain always has S coins. When the coin amount is constant, the following equations can be used to check the summation of individual transactions and the blockchain.

Let the maximum coin amount be in $[0, 2^L)$. For transaction t, let the inputs be $\boldsymbol{c}$ with coins $[v_i]_{i=0}^{*}$ such that $\texttt{binary}(v_i) = [b_{i,j}]_{j=0}^{L}$ and outputs be $\boldsymbol{c}'$ with coins $[v'_i]_{i=0}^{*}$ such that $\texttt{binary}(v'_i) = [b'_{i,j}]_{j=0}^{L}$. Then, the input carries $[c_j]_{j=0}^{L+1}$ are computed as

$$\left[0, \left(\sum_{i=1}^{|\boldsymbol{c}|} b_{i,0} + c_0\right) \div 2, ..., \left(\sum_{i=1}^{|\boldsymbol{c}|} b_{i,L-2} + c_{L-2}\right) \div 2, 0\right]_{j=0}^{L+1}$$

where c_0 and c_L are zeros, and other c_js are either 0 or 1. Similarly, the output carries $[c'_j]_{j=0}^{L+1}$ are

$$\left[0, \left(\sum_{i=1}^{|\boldsymbol{c}'|} b'_{i,0} + c'_0\right) \div 2, ..., \left(\sum_{i=1}^{|\boldsymbol{c}'|} b'_{i,L-2} + c'_{L-2}\right) \div 2, 0\right]_{j=0}^{L+1}$$

such that c'_0 and c'_L are zeros, and c'_js are 0 or 1. Note that $\div$ is the integer division and $|\cdot|$ is the size of a set/vector.

Then, these binary vectors satisfy Equation (1) if inputs coins are equal to output coins.

$$\left[\sum_{i=1}^{|c'|} b'_{i,j} - \sum_{i=1}^{|c|} b_{i,j} + (c_j'^{(t)} - 2c_{j+1}'^{(t)} - c_j^{(t)} + 2c_{j+1}^{(t)}) \stackrel{?}{=} 0\right]_{j=0}^{L} \tag{1}$$

Furthermore, due to the constant coin amount, Equation (2) can be used to verify that unspent coins (`ucoins`) are equal to the total supply S of headers (`headers`) $\texttt{binary}(S) = [S_j]_{j=0}^{L}$.

$$\left[\sum_{i=1}^{|\texttt{ucoins}|} b_{i,j} + \sum_{t=1}^{|\texttt{headers}|} \left(c_j'^{(t)} - 2c_{j+1}'^{(t)} - c_j^{(t)} + 2c_{j+1}^{(t)}\right) \stackrel{?}{=} S_j\right]_{j=0}^{L} \tag{2}$$

Therefore, LACT headers contain carry

$$\texttt{com}([c_j'^{(t)} - 2c_{j+1}'^{(t)} - c_j^{(t)} + 2c_{j+1}^{(t)}]_{j=0}^{L-1}, *)$$

and range proofs of in/output carries (we call them carry proofs) when in/output size is **2**. Note that when there is only one in/output, the carries are always zero, in which case no carry proofs are needed. These carries' range proofs are similar to coins' carry proofs even though a functional output of the carries is committed. However, a normal transaction has to be separated into multiple LACT transactions if there are more than two inputs or outputs. For example, a transaction of $(2 \Rightarrow 3)$ can be separated into two transactions; $(2 \Rightarrow 2)$ and $(1 \Rightarrow 2)$ with two carry proofs.

We generalize this concept of static carry proofs for more than two input and output transactions at a logarithmic cost to reduce the size impact of carry proofs in LACT+.

Let there be a carry proof protocol, `CARRY` that commits and provides range proofs for maximum $2^{L'}$ number of inputs or outputs.

- $\texttt{CARRY.gen}(pp_{(\lambda,L)}, [v_i]_{i=0}^{|\texttt{in}|}, [v_i]_{i=0}^{|\texttt{out}|})$: returns $(k, \texttt{carry})$ if all v and v are in $[0, 2^L - 1]$; otherwise returns $\perp$. Here, k is the secret key of carry commitment, $|\texttt{in}| \leq 2^{L'}$ and $|\texttt{out}| \leq 2^{L'}$. At this point, we do not check $\sum v = \sum v'$, which will be performed in transaction creation. Here, $|\texttt{in}|$ and $|\texttt{out}|$ are the input and output sizes.
- $\texttt{CARRY.ver}(pp_{(\lambda,L)}, \texttt{carry})$: returns one if the hidden carries are in $[0, 2^{L'})$ and the carries are committed according to Equation (1); otherwise, it returns zero.

Protocol Sketch

Carry proofs contain a commitment and a range proof to show that each carry is in a proper range, which is $[0, 2^{L'})$ for maximum $2^{L'} \geq 2$ inputs and outputs. Let there be $2^{L_{\texttt{in}}}$ number of inputs and $2^{L_{\texttt{out}}}$ number of outputs. A LACT+ carry commitment is $\texttt{com}(c_j'^{(t)} - 2c_{j+1}'^{(t)} - c_j^{(t)} + 2c_{j+1}^{(t)}, 0, 0, k)$ when $\boldsymbol{c}$ and $\boldsymbol{c}'$ are input and output carries. Here, except for the 0^{th} and $(L-1)^{\text{th}}$ carries, the output carries $[c'_j]_{j=1}^{L-2}$ are in $[0, 2^{L_{\texttt{out}}})$, and other input carries $[c_j]_{j=1}^{L-2}$ are in $[0, 2^{L_{\texttt{in}}})$. Hence, LACT+ creates range proofs for the following vectors

$$[\texttt{binary}(L_{\texttt{in}}, c_1), ..., \texttt{binary}(L_{\texttt{in}}, c_{L-2})] \in \mathcal{R} \text{ and}$$
$$[\texttt{binary}(L_{\texttt{out}}, c'_1), ..., \texttt{binary}(L_{\texttt{out}}, c'_{L-2})] \in \mathcal{R}.$$

We define the new carry proof protocol in Algorithm 4. Furthermore, we provide a graphical view of the carry proofs in Figure 4 using the same notation as Algorithm 4.

Algorithm 4 New Confidential Carry Proofs

▷ $|\mathtt{in}|, |\mathtt{out}|$ ($\leq 2^{L'}$) are the number of in/outputs, including supplied coins and transaction fee.
$\mathtt{CARRY.gen}(pp_{(\lambda,L)}, [v_i]_{i=0}^{|\mathtt{in}|}, [v_i]_{i=0}^{|\mathtt{out}|})$:
if $|\mathtt{in}| > 2^{L'} \wedge |\mathtt{out}| > 2^{L'}$:
    return $\perp$
$L_{\mathtt{in}} = \lceil \log_2(\lfloor (2|\mathtt{in}| - 1) \rfloor) \rceil$
$L_{\mathtt{out}} = \lceil \log_2(\lfloor (2|\mathtt{out}| - 1) \rfloor) \rceil$
$[b_i = \mathtt{binary}(v^{(i)})]_{i=1}^{|\mathtt{in}|}, [b'_i = \mathtt{binary}(v'^{(i)})]_{i=1}^{|\mathtt{out}|}$
▷ Carries when $\div$ is the integer division
$\boldsymbol{c}_0 = \left[0, [c_{0,j+1} = \left(\sum_{i=1}^{|\mathtt{in}|} b_{i,j} + c_{0,j}\right) \div 2]_{j=0}^{L-1}, 0\right]$
$\boldsymbol{c}_1 = \left[0, [c_{1,j+1} = \left(\sum_{i=1}^{|\mathtt{out}|} b'_{i,j} + c_{1,j}\right) \div 2]_{j=0}^{L-1}, 0\right]$
▷ Bits of input carries
$\boldsymbol{c}'_0 = [\mathtt{binary}_{L_{\mathtt{in}}}(c_{0,j})]_{j=0}^{L+1} \in \mathcal{R}^{(L+1)\times L'}$
▷ Bits of output carries
$\boldsymbol{c}'_1 = [\mathtt{binary}_{L_{\mathtt{out}}}(c_{1,j})]_{j=0}^{L+1} \in \mathcal{R}^{(L+1)\times L'}$
▷ Compute random masks for carries
▷ Lazy Sampling
$[[\vec{\boldsymbol{d}}_{0,i,l} \xleftarrow{\$} [-(\alpha - (1 - c'_{0,i,l})), (\alpha - (1 - c'_{0,i,l}))]^N]_{i=1}^{L}]_{l=0}^{L_{\mathtt{in}}}$
Set $\vec{\boldsymbol{d}}_{0,0} = \vec{\mathbf{0}}$ and $\vec{\boldsymbol{d}}_{0,L} = \vec{\mathbf{0}}$
$[[\vec{\boldsymbol{d}}_{1,i,l} \xleftarrow{\$} [-(\alpha - (1 - c'_{1,i,l})), (\alpha - (1 - c'_{1,i,l}))^N]_{i=1}^{L}]_{l=0}^{L_{\mathtt{out}}}$
Set $\vec{\boldsymbol{d}}_{1,0} = \vec{\mathbf{0}}$ and $\vec{\boldsymbol{d}}_{1,L} = \vec{\mathbf{0}}$
$\hat{\vec{c}}_0 = \sum_{l=0}^{L_{\mathtt{in}}} \sum_{j=0}^{L} \vec{\boldsymbol{d}}_{0,j,l} \mathtt{rot}(2c'_{0,j,l} - 1, jL_{\mathtt{in}} + l) \in \mathcal{R}$
$\hat{\vec{c}}_1 = \sum_{l=0}^{L_{\mathtt{out}}} \sum_{j=0}^{L} \vec{\boldsymbol{d}}_{1,j} \mathtt{rot}(2c'_{1,j,l} - 1, jL_{\mathtt{out}} + l) \in \mathcal{R}$
$\hat{\vec{c}}' = [(c_{1,j} - c_{0,j}) - 2(c_{1,j+1} - c_{0,j+1})]_{j=0}^{L}, \mathbf{0}^{N-L}] \in \mathcal{R}$
$t_{\mathtt{in}} = N - j \cdot (L_{\mathtt{in}} - 1) - l$
$t'_{\mathtt{in}} = N - (j+1) \cdot (L_{\mathtt{in}} - 1) - l - 1$
$\hat{\vec{d}}'_0 = \sum_{l=0}^{L_{\mathtt{in}}} \sum_{j=0}^{L} (\vec{\boldsymbol{d}}_{0,j,l} \mathtt{rot}(-2^l, t_{\mathtt{in}}) - 2\vec{\boldsymbol{d}}_{0,j+1,l} \mathtt{rot}(-2^l, t'_{\mathtt{in}}))$
$t_{\mathtt{out}} = N - j \cdot (L_{\mathtt{out}} - 1) - l$
$t'_{\mathtt{out}} = N - (j+1) \cdot (L_{\mathtt{out}} - 1) - l - 1$
$\hat{\vec{d}}'_1 = \sum_{l=0}^{L_{\mathtt{out}}} \sum_{j=0}^{L} (\vec{\boldsymbol{d}}_{1,j,l} \mathtt{rot}(-2^l, t_{\mathtt{out}}) - 2\vec{\boldsymbol{d}}_{1,j+1,l} \mathtt{rot}(-2^l, t'_{\mathtt{out}}))$
▷ Pick random masks
$\vec{\boldsymbol{k}} \xleftarrow{\$} [-\tau, \tau]^{(m-3)\times N}$
$\vec{\boldsymbol{r}}_1 \xleftarrow{\$} [-\tau_1, \tau_1]^{(m-3)\times N}; \vec{\boldsymbol{r}}_2 \xleftarrow{\$} [-\tau_2, \tau_2]^{(m-3)\times N}$
▷ Carry commitment
$\vec{\boldsymbol{u}} = \mathtt{highbits}_{p_1}(\vec{\boldsymbol{H}}[\hat{\vec{c}}', \vec{0}, \vec{0}, \vec{\boldsymbol{k}}]) \in \mathcal{R}_q^n$
$\vec{\boldsymbol{t}}_1 = \mathtt{highbits}_{p_2}(\vec{\boldsymbol{H}}[\vec{0}, \hat{\vec{c}}_1, \hat{\vec{c}}_0, \vec{\boldsymbol{r}}_1]) \in \mathcal{R}_q^n$
$\vec{x}_1 = \mathtt{hash}(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}}_1) \in \mathcal{C}_\beta^N$
$\vec{\boldsymbol{s}}_2 = [\vec{x}_1(\hat{\vec{d}}'_1 - \hat{\vec{d}}'_0), \sum_{l=0}^{L_{\mathtt{out}}} \sum_{j=0}^{L} \vec{\boldsymbol{d}}^2_{1,j,l}, \sum_{l=0}^{L_{\mathtt{in}}} \sum_{j=0}^{L} \vec{\boldsymbol{d}}^2_{0,j,l}, \vec{\boldsymbol{r}}_2]$
$\vec{\boldsymbol{t}}_2 = \mathtt{highbits}_{\gamma'}(\vec{\boldsymbol{H}}\vec{\boldsymbol{s}}_2) \in \mathcal{R}_q^n$
$\vec{x}_2 = \mathtt{hash}(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}}_1, \vec{\boldsymbol{t}}_2) \in \mathcal{C}_\beta^N$
$\vec{\boldsymbol{z}}_0 = [[\vec{x}_2 \mathtt{rot}(c_{0,j,l}, jL_{\mathtt{in}} + l) + \vec{\boldsymbol{d}}_{0,j,l}]_{j=1}^{L}]_{l=0}^{L_{\mathtt{in}}} \in \mathcal{R}^{L_{\mathtt{in}} \times L}$
$\vec{\boldsymbol{z}}_1 = [[\vec{x}_2 \mathtt{rot}(c_{1,j,l}, jL_{\mathtt{out}} + l) + \vec{\boldsymbol{d}}_{1,j,l}]_{j=1}^{L}]_{l=0}^{L_{\mathtt{in}}} \in \mathcal{R}^{L_{\mathtt{out}} \times L}$
if $\|\vec{\boldsymbol{z}}_0\| > (\alpha - 1) \vee \|\vec{\boldsymbol{z}}_1\| > (\alpha - 1)$:
    go to Step 15
$\vec{\boldsymbol{r}} = \vec{x}_2(\vec{x}_1 \vec{\boldsymbol{k}} + \vec{\boldsymbol{r}}_1) + \vec{\boldsymbol{r}}_2 \in \mathcal{R}^{m-3}$
if $\|\vec{\boldsymbol{r}}\| > (\tau_2 - \beta^2\tau - \beta\tau_1)$:
    go to Step 15
$\hat{\vec{z}}'_0 = \vec{x}_1 \sum_{l=0}^{L_{\mathtt{in}}} \sum_{j=0}^{L} (\vec{\boldsymbol{z}}_{0,j,l} \mathtt{rot}(-2^l, t_{\mathtt{in}}) - 2\vec{\boldsymbol{z}}_{0,j+1,l} \mathtt{rot}(-2^l, t'_{\mathtt{in}})) \in \mathcal{R}$
$\hat{\vec{z}}'_1 = \vec{x}_1 \sum_{l=0}^{L_{\mathtt{out}}} \sum_{j=0}^{L} (\vec{\boldsymbol{z}}_{1,j,l} \mathtt{rot}(-2^l, t_{\mathtt{out}}) - 2\vec{\boldsymbol{z}}_{1,j+1,l} \mathtt{rot}(-2^l, t'_{\mathtt{out}})) \in \mathcal{R}$
$\hat{\vec{z}}_0 = \sum_{l=0}^{L_{\mathtt{in}}} \sum_{i=0}^{L} \vec{\boldsymbol{z}}_{0,i} (\vec{\boldsymbol{z}}_{0,i} - \vec{x}_2 \mathtt{rot}(1, iL_{\mathtt{in}} + l)) \in \mathcal{R}$
$\hat{\vec{z}}_1 = \sum_{l=0}^{L_{\mathtt{out}}} \sum_{i=0}^{L} \vec{\boldsymbol{z}}_{1,i} (\vec{\boldsymbol{z}}_{1,i} - \vec{x}_2 \mathtt{rot}(1, iL_{\mathtt{out}} + l)) \in \mathcal{R}$
$\vec{\boldsymbol{s}} = [\hat{\vec{z}}'_1 - \hat{\vec{z}}'_0, \hat{\vec{z}}_1, \hat{\vec{z}}_0, \vec{\boldsymbol{r}}] \in \mathcal{R}^m$
if $\|\vec{\boldsymbol{s}}\| > \gamma$: go to Step 15
▷ Compute hints for $\vec{t}_2$
$\vec{\boldsymbol{t}}'_2 = \mathtt{highbits}_{\gamma'}(\vec{\boldsymbol{H}}\vec{\boldsymbol{s}} - \vec{x}_2(\vec{x}_1 \mathtt{up}_{p_1}(\vec{\boldsymbol{u}}) + \mathtt{up}_{p_2}(\vec{\boldsymbol{t}}_1))) \in \mathcal{R}_q^n$
$\vec{\boldsymbol{h}} = \mathtt{hints}(\chi, \vec{\boldsymbol{t}}'_2, \vec{\boldsymbol{t}}_2)$
if $\vec{\boldsymbol{h}} = \perp$: go to Step 15
return $\mathtt{carry} = (\vec{\boldsymbol{z}}_0, \vec{\boldsymbol{z}}_1, \vec{\boldsymbol{r}}, \vec{\boldsymbol{u}}, \vec{\boldsymbol{t}}_1, \vec{\boldsymbol{h}}, \vec{x}_2)$

$\mathtt{CARRY.ver}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, \mathtt{carry})$:
▷ $(*)$ denotes variables that can be empty
$(\vec{\boldsymbol{z}}_0^{(*)}, \vec{\boldsymbol{z}}_1^{(*)}, \vec{\boldsymbol{r}}^{(*)}, \vec{\boldsymbol{u}}^{(*)}, \vec{\boldsymbol{t}}_1^{(*)}, \vec{\boldsymbol{h}}^{(*)}, \vec{x}_2^{(*)}) := \mathtt{carry}$
if $|\mathtt{in}| > 2^{L'} \wedge |\mathtt{out}| > 2^{L'}$:
    return $\perp$
$L_{\mathtt{in}} = \lceil \log_2(\lfloor (2|\mathtt{in}| - 1) \rfloor) \rceil$
$L_{\mathtt{out}} = \lceil \log_2(\lfloor (2|\mathtt{out}| - 1) \rfloor) \rceil$
$t_{\mathtt{in}} = N - j \cdot (L_{\mathtt{in}} - 1) - l$
$t'_{\mathtt{in}} = N - (j+1) \cdot (L_{\mathtt{in}} - 1) - l - 1$
$t_{\mathtt{out}} = N - j \cdot (L_{\mathtt{out}} - 1) - l$
$t'_{\mathtt{out}} = N - (j+1) \cdot (L_{\mathtt{out}} - 1) - l - 1$
if $\|\vec{\boldsymbol{z}}_0\| > \alpha \vee \|\vec{\boldsymbol{z}}_1\| > \alpha$:
    return 0
$\hat{\vec{z}}'_0 = \vec{x}_1 \sum_{l=0}^{L_{\mathtt{in}}} \sum_{j=0}^{L} (\vec{\boldsymbol{z}}_{0,j,l} \mathtt{rot}(-2^l, t_{\mathtt{in}}) - 2\vec{\boldsymbol{z}}_{0,j+1,l} \mathtt{rot}(-2^l, t'_{\mathtt{in}})) \in \mathcal{R}$
$\hat{\vec{z}}'_1 = \vec{x}_1 \sum_{l=0}^{L_{\mathtt{out}}} \sum_{j=0}^{L} (\vec{\boldsymbol{z}}_{1,j,l} \mathtt{rot}(-2^l, t_{\mathtt{out}}) - 2\vec{\boldsymbol{z}}_{1,j+1,l} \mathtt{rot}(-2^l, t'_{\mathtt{out}})) \in \mathcal{R}$
$\hat{\vec{z}}_0 = \sum_{l=0}^{L_{\mathtt{in}}} \sum_{i=0}^{L} \vec{\boldsymbol{z}}_{0,i} (\vec{\boldsymbol{z}}_{0,i} - \vec{x}_2 \mathtt{rot}(1, iL_{\mathtt{in}} + l)) \in \mathcal{R}$
$\hat{\vec{z}}_1 = \sum_{l=0}^{L_{\mathtt{out}}} \sum_{i=0}^{L} \vec{\boldsymbol{z}}_{1,i} (\vec{\boldsymbol{z}}_{1,i} - \vec{x}_2 \mathtt{rot}(1, iL_{\mathtt{out}} + l)) \in \mathcal{R}$
$\vec{\boldsymbol{s}} = [\hat{\vec{z}}'_1 - \hat{\vec{z}}'_0, \hat{\vec{z}}_1, \hat{\vec{z}}_0, \vec{\boldsymbol{r}}] \in \mathcal{R}^m$
if $\|\vec{\boldsymbol{s}}\| > \gamma$: return 0
▷ Recompute $\vec{t}_2$ from hints
$\vec{\boldsymbol{t}}'_2 = \mathtt{highbits}_{\gamma'}(\vec{\boldsymbol{H}}\vec{\boldsymbol{s}} - \vec{x}_2(\vec{x}_1 \mathtt{up}_{p_1}(\vec{\boldsymbol{u}}) + \mathtt{up}_{p_2}(\vec{\boldsymbol{t}}_1))) \in \mathcal{R}_q^n$
$\vec{\boldsymbol{t}}_2 = \mathtt{use\_hints}(\chi, \vec{\boldsymbol{t}}'_2, \vec{\boldsymbol{h}})$
if $\vec{\boldsymbol{t}}_2 = \perp$: return 0
return $\vec{x}_2 \stackrel{?}{=} \mathtt{hash}(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}}_1, \vec{\boldsymbol{t}}_2)$

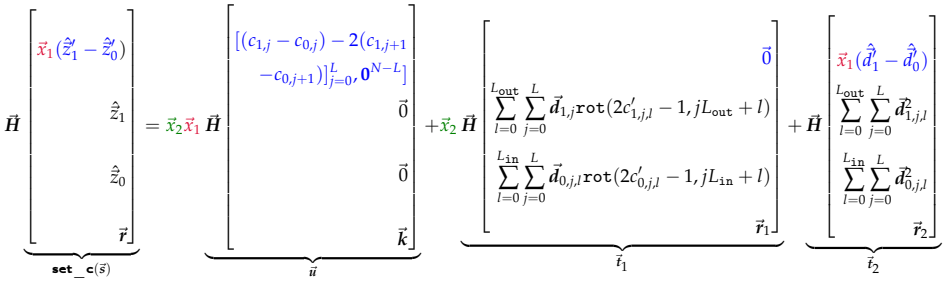

Figure 4. Graphical overview of carry proofs (Algorithm 4).

New carry proofs provide the following properties; *completeness*, *knowledge soundness*, and *zero-knowledge argument*. Zero-knowledge and knowledge soundness are defined for a statement $\mathcal{L}$,

"all input and output carries are in $[0, 2^{L_{\mathtt{in}}})$ and $[0, 2^{L_{\mathtt{out}}})$"

according to Definitions 2 and 3.

Definition 9 (Completeness)**.** `CARRY` *is complete if honestly generated proofs are always valid such that the following is greater than or equal to* $1 - \epsilon(\lambda)$.

$$Pr\left[\begin{array}{c} \mathtt{CARRY.ver}(pp_{(\lambda,L)}, \\ |\mathtt{in}|, |\mathtt{out}|, \mathtt{c}) \end{array} \middle| \begin{array}{c} \boldsymbol{v}, \boldsymbol{v}' \in [0, 2^L - 1] \; s.t. \; |\boldsymbol{v}|, |\boldsymbol{v}'| | \leq 2^{L'} \\ (k, \mathtt{c}) = \mathtt{CARRY.gen}(pp_{(\lambda,L)}, \boldsymbol{v}, \boldsymbol{v}') \end{array}\right]$$

Theorem 3. `CARRY` *is complete, zero-knowledge, and witness extractable if Approx-SIS of* $(n, m, q, \gamma, \gamma', N)$ *is hard (see the security proof in Appendix B).*

3.3. Aggregable Confidential Transactions with Activity Proofs

In this section, we describe LACT+ transactions, `TX`, with new confidential coins, carry proofs, and activity proofs.

A transaction converts a set of existing coin records (inputs) into new coins (outputs). First, users agree on the inputs and outputs. Then they create a carry proof to show the summation. Finally, they create the transaction by computing a signature to prove the ownership. Let $[\boldsymbol{b}]_{i=0}^{*}$, $[\boldsymbol{b}']_{i=0}^{*}$, $\boldsymbol{c}_0$, and $\boldsymbol{c}_1$ be the binary inputs, binary outputs, input carries, and output carries accordingly. If the input coins are equal to output coins, they satisfy Equation (1) and a zero-coefficient vector is output. Therefore, the summed commitment pk is,

$$\mathtt{com}\left(\begin{array}{ll} \sum_{i=0}^{*} \boldsymbol{b}' - \sum_{i=0}^{*} \boldsymbol{b} & \sum_{i=0}^{*} k_i' \\ +(c_j'^{(t)} - 2c_{j+1}'^{(t)}, 0, 0, & \\ -c_j^{(t)} + 2c_{j+1}^{(t)}) & \underbrace{-\sum_{i=0}^{*} k_i + k}_{r} \end{array}\right) = \mathtt{com}(0, 0, 0, r) = pk$$

when k, $[k]_{i=0}^{*}$, and $[k']_{i=0}^{*}$ are the keys of the carry commitment, input coin commitments, and output coin commitments. Therefore, `TX` asks the users to use (r, pk) as the secret–public key pair and create a signature. First, the users picks a random r' and computes $y = \mathtt{com}(0, 0, 0, r')$. Then, users' computers receive the challenge $x = \mathtt{hash}(u, y)$ by hashing pk and y, and compute $\sigma = r' - xr$. The signature is (σ, x). If the input coins are equals to output coins, verifiers can recompute $y' := \mathtt{com}(0, 0, 0, \sigma) - x \times pk$ and check $x \stackrel{?}{=} \mathtt{hash}(pk, y')$. When the challenge is large enough, users cannot cheat and create valid signatures if they illegally generate coins or steal coins.

LACT+ transactions contain activity proofs to enable complete aggregation. Activity proofs are immutable, or it is computationally impossible to find two different input and

output sets for the same activity proof. Hence, preserving activity proofs is sufficient to provide immutability to the whole blockchain, including the removed spent coins.

The transaction generation is a multi-party protocol where each sender or receiver keeps their secret keys without sharing and computes a secure partial signature. At the end, those partial signatures are aggregated into a single signature used as the transaction's signature. Therefore, the senders do not learn the secret keys of other inputs and outputs, and cannot take back the sent coins. We use "green color" to denote the secret computations of each individual.

Before aggregating a transaction into the chain, verifiers check whether the inputs are in the chain or not. If they are in the chain, verifiers check the summation, ownership, and activity of the transaction. When the transaction has valid proofs, the verifiers remove the inputs from the chain, add outputs into the unspent coin set, and preserve the header. The complete protocol of TX is stated in Algorithm 5 and an example of transaction aggregation is stated in Figure 2.

Algorithm 5 LACT+ Transactions and Chains

▷ $|\mathtt{in}|, |\mathtt{out}|$ $(\leq 2^{L'})$ are the number of in/outputs.
▷ owners do not share secret keys of inputs or outputs.
$\mathtt{TX.gen}(pp_{(\lambda,L)}, \boldsymbol{k}, \mathtt{carry}, [\boldsymbol{k}_i, v_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [\boldsymbol{k}'_i, v'_i, \mathtt{coin}'_i]_{i=1}^{|\mathtt{out}|})$:

if $\sum v \neq \sum v'$: return $\perp$
▷ Commitments cannot be repeated
if $\bigcup \mathtt{coin}.\vec{\boldsymbol{u}} \neq \mathtt{coin}.\vec{\boldsymbol{u}} \vee \bigcup \mathtt{coin}'.\vec{\boldsymbol{u}} \neq \mathtt{coin}'.\vec{\boldsymbol{u}}$
  $\vee \mathtt{coin} \bigcap \mathtt{coin}' \neq \phi$: return 0
$\Delta \neq \mathtt{activity}([\mathtt{coin}'_i.\vec{\boldsymbol{u}}]_{i=1}^{|\mathtt{out}|}, [\mathtt{coin}_i.\vec{\boldsymbol{u}}]_{i=1}^{|\mathtt{in}|}) \in \mathbb{H}$
Each sender $i \in [1, |\mathtt{in}|]$ secretly computes:
  ▷ $\vec{\boldsymbol{r}}_{3,i} \xleftarrow{\$} [-\tau_3, \tau_3]^{(m-3)\times N}$
  ▷ $\vec{\boldsymbol{y}}_i = \vec{\boldsymbol{H}}[\vec{0}, \vec{0}, \vec{0}, \vec{\boldsymbol{r}}_{3,i}] \in \mathcal{R}_q^n$ and share $\vec{\boldsymbol{y}}_i$
Each receiver $i \in [1, |\mathtt{out}|]$ secretly computes:
  ▷ $\vec{\boldsymbol{r}}'_{3,i} \xleftarrow{\$} [-\tau_3, \tau_3]^{(m-3)\times N}$
  ▷ $\vec{\boldsymbol{y}}'_i = \vec{\boldsymbol{H}}[\vec{0}, \vec{0}, \vec{0}, \vec{\boldsymbol{r}}'_{3,i}] \in \mathcal{R}_q^n$ and share $\vec{\boldsymbol{y}}'_i$
▷ All receivers compute public key $\vec{\boldsymbol{pk}} \in \mathcal{R}_q^n$.
$\vec{\boldsymbol{pk}} = \mathtt{highbits}_{p_1}(\mathtt{carry}.\vec{\boldsymbol{u}} + \sum_{i=1}^{|\mathtt{out}|} \mathtt{coin}'_i.\vec{\boldsymbol{u}} - \sum_{i=1}^{|\mathtt{in}|} \mathtt{coin}_i.\vec{\boldsymbol{u}})$

$\vec{\boldsymbol{y}} = \mathtt{highbits}_{\gamma'}(\sum_{i=1}^{|\mathtt{out}|} \vec{\boldsymbol{y}}'_i - \sum_{i=1}^{|\mathtt{in}|} \vec{\boldsymbol{y}}_i) \in \mathcal{R}_q^n$
$\vec{x}_0 = \mathtt{hash}(\vec{\boldsymbol{pk}}, \vec{\boldsymbol{y}}, \Delta) \in \mathcal{C}_\beta^N$ ▷ Signature challenge
Each sender $i \in [1, |\mathtt{in}|]$ secretly computes:
  ▷ $\vec{\boldsymbol{\sigma}}_i = \vec{\boldsymbol{r}}_{3,i} + \vec{x}_0 \vec{\boldsymbol{k}}_i \in \mathcal{R}^{(m-3)}$
  ▷ **if** $\|\vec{\boldsymbol{\sigma}}_i\| > (\tau_3 - \beta\tau)$: go to Step 9
Each receiver $i \in [1, |\mathtt{out}|]$ secretly computes:
  ▷ $\vec{\boldsymbol{\sigma}}'_i = \vec{\boldsymbol{r}}'_{3,i} + \vec{x}_0 \vec{\boldsymbol{k}}'_i \in \mathcal{R}^{(m-3)}$
  ▷ **if** $\|\vec{\boldsymbol{\sigma}}'_i\| > (\tau_3 - \beta\tau)$: go to Step 9
▷ All receivers compute the signature
$\vec{\boldsymbol{\sigma}} = \vec{x}_0(\vec{\boldsymbol{k}} + \sum_{i=1}^{|\mathtt{out}|} \vec{\boldsymbol{\sigma}}'_i - \sum_{i=1}^{|\mathtt{in}|} \vec{\boldsymbol{\sigma}}_i) \in \mathcal{R}^{(m-3)}$
if $\|\vec{\boldsymbol{\sigma}}\| > ((|\mathtt{out}| + |\mathtt{in}|)\tau_3 - (|\mathtt{out}| + |\mathtt{in}| + 1)\beta\tau)$:
  go to Step 9
$\vec{\boldsymbol{y}}' = \mathtt{highbits}_{\gamma'}(\vec{\boldsymbol{H}}\vec{\boldsymbol{\sigma}} - \vec{x}_0 \mathtt{up}_{p_1}(\vec{\boldsymbol{pk}})) \in \mathcal{R}_q^n$
$\vec{\boldsymbol{h}} = \mathtt{hints}(\chi, \vec{\boldsymbol{y}}', \vec{\boldsymbol{y}})$ ▷ Hints for $\vec{\boldsymbol{y}}$
if $\vec{\boldsymbol{h}} = \perp$: go to Step 9
$\mathtt{header} = (\vec{\boldsymbol{pk}}, \vec{\boldsymbol{h}}, \vec{\boldsymbol{\sigma}}, \vec{x}_0, \mathtt{carry}, \Delta)$
return $\mathtt{tx} = (\mathtt{header}, [\mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [\mathtt{coin}'_i]_{i=1}^{|\mathtt{out}|})$

▷ A chain is an aggregated CT with multiple headers
$\mathtt{TX.aggregate}(pp_{(\lambda,L)}, \mathtt{chain}, \mathtt{tx})$:
$(\mathtt{header}, \mathtt{in}, \mathtt{out})) := \mathtt{tx}$
$(\mathtt{headers}, S, \mathtt{ucoins}) := \mathtt{chain}$
if $\neg \mathtt{TX.ver}(pp_{(\lambda,L)}, \mathtt{tx})$: return $\perp$
▷ Spending coins should be in the chain
if $\mathtt{tx.in} \notin \mathtt{ucoins}$: return $\perp$
▷ Remove spending coins from the chain
$\mathtt{ucoins} = (\mathtt{ucoins} \setminus \mathtt{tx.in}) \uplus \mathtt{tx.out}$
▷ Add proofs to the transaction table
$\mathtt{headers} = \mathtt{headers} \uplus (|\mathtt{tx.in}|, |\mathtt{tx.out}|, \mathtt{tx.header})$
▷ Commitments cannot be repeated
if $\bigcup \mathtt{ucoins}.\vec{\boldsymbol{u}} \neq \mathtt{ucoins}.\vec{\boldsymbol{u}}$: return $\perp$
return $\mathtt{chain} = (\mathtt{headers}, S, \mathtt{ucoins})$)

$\mathtt{TX.header\_ver}\ (pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, \mathtt{header})$:
$(\vec{\boldsymbol{pk}}, \vec{\boldsymbol{h}}, \vec{\boldsymbol{\sigma}}, \vec{x}_0, \mathtt{carry}, \Delta) := \mathtt{header}$
▷ Recompute $\vec{\boldsymbol{y}}$ from hints
$\vec{\boldsymbol{y}}' = \mathtt{highbits}_{\gamma'}(\vec{\boldsymbol{H}}\vec{\boldsymbol{\sigma}} - \vec{x}_0 \mathtt{up}_{p_1}(\vec{\boldsymbol{pk}})) \in \mathcal{R}_q^n$
$\vec{\boldsymbol{y}} = \mathtt{use\_hints}(\chi, \vec{\boldsymbol{y}}', \vec{\boldsymbol{h}})$
if $\vec{\boldsymbol{y}} = \perp$: return 0
return $\vec{x}_0 \stackrel{?}{=} \mathtt{hash}(\vec{\boldsymbol{pk}}, \vec{\boldsymbol{y}}, \Delta) \wedge$
  $\mathtt{CARRY.ver}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, \mathtt{carry})$

$\mathtt{TX.ver}(pp_{(\lambda,L)}, \mathtt{tx})$:
$(\mathtt{header}, [\mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [\mathtt{coin}'_i]_{i=1}^{|\mathtt{out}|})) := \mathtt{tx}$
$(\vec{\boldsymbol{pk}}, \vec{\boldsymbol{h}}, \vec{\boldsymbol{\sigma}}, \vec{x}_0, \mathtt{carry}, \Delta) := \mathtt{header}$
▷ Commitments cannot be repeated
if $\bigcup \mathtt{coin}.\vec{\boldsymbol{u}} \neq \mathtt{coin}.\vec{\boldsymbol{u}} \vee \bigcup \mathtt{coin}'.\vec{\boldsymbol{u}} \neq \mathtt{coin}'.\vec{\boldsymbol{u}}$:
  $\vee \mathtt{coin} \bigcap \mathtt{coin}' \neq \phi$: return 0
▷ Check activity
if $\Delta = \mathtt{activity}([\mathtt{coin}'_i.\vec{\boldsymbol{u}}]_{i=1}^{|\mathtt{out}|}, [\mathtt{coin}_i.\vec{\boldsymbol{u}}]_{i=1}^{|\mathtt{in}|}) \in \mathbb{H}$: return 0

if $[\neg \mathtt{COIN.ver}(pp_{(\lambda,L)}, \mathtt{coin}_i)]_{i=1}^{|\mathtt{in}|}$: return 0
if $[\neg \mathtt{COIN.ver}(pp_{(\lambda,L)}, \mathtt{coin}'_i)]_{i=1}^{|\mathtt{out}|}$: return 0
if $\neg \mathtt{TX.header\_ver}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, \mathtt{header})$:
  return 0
return $\mathtt{highbits}_{\gamma'-p1}(\vec{\boldsymbol{pk}}) = \mathtt{highbits}_{\gamma'}(\mathtt{carry}.\vec{\boldsymbol{u}}$
  $+ \sum_{i=1}^{|\mathtt{out}|} \mathtt{coin}'_i.\vec{\boldsymbol{u}} - \sum_{i=1}^{|\mathtt{in}|} \mathtt{coin}_i.\vec{\boldsymbol{u}}) \in \mathcal{R}_q^n$

$\mathtt{TX.chain\_ver}(pp_{(\lambda,L)}, \mathtt{chain})$:
$(\mathtt{headers}, S, \mathtt{ucoins}) := \mathtt{chain}$
$\vec{\boldsymbol{pk}} = \vec{\boldsymbol{u}} = \vec{\boldsymbol{0}}^n \in \mathcal{R}_q^n$; $\vec{\boldsymbol{s}} = [\mathtt{bin}(S), \vec{\boldsymbol{0}}^{(m-1)}] \in \mathcal{R}_q^m$
▷ Commitments cannot be repeated
if $\bigcup \mathtt{ucoins}.\vec{\boldsymbol{u}} \neq \mathtt{ucoins}.\vec{\boldsymbol{u}}$: return 0
for all $(\mathtt{coin}_i) \in \mathtt{ucoins}$:
  if $\neg \mathtt{COIN.ver}(pp_{(\lambda,L)}, \mathtt{coin}_i)$: return 0
  $\vec{\boldsymbol{u}} = \vec{\boldsymbol{u}} + \mathtt{coin}_i.\vec{\boldsymbol{u}} \in \mathcal{R}_q^n$
for all $(|\mathtt{in}|, |\mathtt{out}|, \mathtt{h}_i) \in \mathtt{headers}$:
  if $\neg \mathtt{TX.header\_ver}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, \mathtt{h})$: d return 0
  $\vec{\boldsymbol{pk}} = \vec{\boldsymbol{pk}} + \mathtt{h.carry}_i.\vec{\boldsymbol{pk}} \in \mathcal{R}_q^n$
  $\vec{\boldsymbol{u}} = \vec{\boldsymbol{u}} + \mathtt{h.carry}_i.\vec{\boldsymbol{u}} \in \mathcal{R}_q^n$
return $\mathtt{highbits}_{\gamma'-p1}(\vec{\boldsymbol{pk}}) \stackrel{?}{=} \mathtt{highbits}_{\gamma'-p1}(\vec{\boldsymbol{u}} - \vec{\boldsymbol{H}}\vec{\boldsymbol{s}})$
$\wedge \prod_{\Delta \in \mathtt{headers}} \Delta \stackrel{?}{=} \prod_{\mathtt{c} \in \mathtt{ucoins}} \mathtt{hash}_0(\mathtt{c}.\vec{\boldsymbol{u}}) \in \mathbb{H}$

TX is expected to have the following security properties: *completeness*, *zero-knowledge*, *knowledge soundness*, *theft resistance*, and *immutability*. We reuse zero-knowledge and knowledge soundness from Definitions 2 and 3 when the statements of $\mathcal{L}$ are:

"all coins are in $[0, 2^L)$" and
"total unspent coins are equal to total supplied coins".

Definition 10 (Completeness)**.** TX *is complete if the honestly generated transactions are always valid, and aggregating a valid transaction into a valid chain always produces a correct chain with the unspent coins of that transaction. Let* $\mathcal{G}$ *be an arbitrary chain generator.*

$$Pr[\texttt{arbitrary\_tx}() \wedge \texttt{arbitrary\_chain}()] \geq 1 - \epsilon(\lambda).$$

$\underline{\texttt{arbitrary\_tx}() :}$
For any $[v_i]_{i=1}^{|\texttt{in}|} \in [0, 2^L - 1]$ and $[v'_i]_{i=1}^{|\texttt{out}|} \in [0, 2^L - 1]$
  such that $\sum_{i=1}^{|\texttt{in}|}[v_i] = \sum_{i=1}^{|\texttt{out}|}[v'_i] \in [0, 2^L - 1]$:
  $[(k_i, \texttt{coin}_i) = \texttt{COIN.gen}(pp_{(\lambda,L)}, v_i)]_{i=1}^{|\texttt{in}|}$
  $[(k'_i, \texttt{coin}'_i) = \texttt{COIN.gen}(pp_{(\lambda,L)}, v'_i)]_{i=1}^{|\texttt{out}|}$
$\texttt{tx} = \texttt{TX.gen}(pp_{(\lambda,L)}, [k_i, \texttt{coin}_i]_{i=1}^{|\texttt{in}|}, [k'_i, \texttt{coin}'_i]_{i=1}^{|\texttt{out}|})$
return $\texttt{tx.in} \stackrel{?}{=} [\texttt{coin}]_{i=1}^{|\texttt{in}|} \wedge \texttt{tx.out} \stackrel{?}{=} [\texttt{coin}']_{i=1}^{|\texttt{out}|} \wedge$
  $\texttt{TX.ver}(pp_{(\lambda,L)}, \texttt{tx})$
$\underline{\texttt{arbitrary\_chain}() :}$
Get any chain such that
  $\texttt{TX.chain\_ver}(pp_{(\lambda,L)}.\texttt{chain}) = 1$ and $\texttt{T} = |\texttt{headers}|$
for $i \in [0, \texttt{t}]$:
  $\texttt{tx}_i \xleftarrow{\$} \mathcal{G}(pp_{(\lambda,L)})$ s.t. $\texttt{CTx.ver}(pp_{(\lambda,L)}, \texttt{tx}_i) = 1$
  $\texttt{chain} = \texttt{TX.aggregate}(pp_{(\lambda,L)}, \texttt{chain}, \texttt{tx})$
  if $\texttt{chain} = \perp$: continue
  if $\texttt{headers}_{\texttt{T}+i} \neq \texttt{tx.carry}$
    $\vee \texttt{tx}_\texttt{i}\texttt{.out} \not\subset \texttt{chain}'\texttt{.ucoins}$: return 0
return $\texttt{TX.chain\_ver}(pp_{(\lambda,L)}, \texttt{chain}) \wedge |\texttt{headers}| \stackrel{?}{=} \texttt{T} + \texttt{t}$

Even after aggregation, TX ensures that unspent coins cannot be spent without their secret keys. We consider strong theft resistance where the adversary has control over everything except an honest user's secret key. Here, the adversarial algorithm generates a chain (an aggregated CT). In the first step, the algorithm sends some coins to the honest user, where the user locks those coins with a new key. Then, the algorithm tries to spend those coins without obtaining the new secret key. In other words, the network is set to have only one honest user, simulating a real-world decentralized network where everyone can be malicious except the honest user.

Definition 11 (Theft Resistance)**.** TX *is theft resistant if*

$$\texttt{Adv}^{\texttt{TR},\mathcal{A}}_{\texttt{TX},pp_{(\lambda,L)}} = Pr\left[\texttt{Game}^{\texttt{TR},\mathcal{A}}_{\texttt{TX},pp_{(\lambda,L)}} \stackrel{?}{=} 1\right] \leq \epsilon(\lambda).$$

$\underline{\texttt{Game}^{\texttt{TR},\mathcal{A}}_{\texttt{TX},pp_{(\lambda,L)}} : \texttt{chain} \leftarrow \mathcal{A}_{step1}(pp_{(\lambda,L)})}$
if $\texttt{TX.chain\_ver}(pp_{(\lambda,L)}, \texttt{chain})$: return $\perp$
$\texttt{T} = |\texttt{chain.headers}|$
▷ Receive $\texttt{coin}' \in \texttt{tx}$ from $\mathcal{A}$ s.t. $\mathcal{A}$ does not know the key of $\texttt{coin}'$.
$\texttt{chain} = \texttt{TX.aggregate}(pp_{(\lambda,L)}, \texttt{chain}, \texttt{tx})$
$\texttt{chain}' \leftarrow \mathcal{A}_{step2}(pp_{(\lambda,L)}, \texttt{chain})$
▷ Check whether the coin has been spent or not
return $\texttt{TX.chain\_ver}(pp_{(\lambda,L)}, \texttt{chain}')$
  $\wedge \texttt{coin}' \not\subset \texttt{chain}'\texttt{.ucoins}$
  $\wedge [\texttt{chain.headers}_\texttt{i}]_{i=0}^{\texttt{T}+1} \stackrel{?}{=} [\texttt{chain}'\texttt{.headers}]_{i=0}^{\texttt{T}+1}$

Informally, blockchains' immutability states that generating two different input and output sets for the same consensus data should be computationally infeasible. Because we build TX for a generic consensus mechanism, we capture the immutability such that finding two different input and output sets for the same header(s) is computationally difficult. Therefore, preserving headers is sufficient to provide immutability to the whole blockchain, *including removed spent coin records*.

Definition 12 (Immutability). TX *ensures immutability if*

$$\mathtt{Adv}^{\mathtt{IM},\mathcal{A}}_{\mathtt{TX},pp_{(\lambda,L)}} = Pr\left[\mathtt{Game}^{\mathtt{IM},\mathcal{A}}_{\mathtt{TX},pp_{(\lambda,L)}} \stackrel{?}{=} 1\right] \leq \epsilon(\lambda).$$

$$\begin{aligned}&\mathtt{Game}^{\mathtt{IM},\mathcal{A}}_{\mathtt{TX},pp_{(\lambda,L)}} : (\mathtt{tx},\mathtt{tx}') \leftarrow \mathcal{A}(pp_{(\lambda,L)})\\ &\text{return } \mathtt{TX.ver}(pp_{(\lambda,L)},\mathtt{tx}) \wedge \mathtt{TX.ver}(pp_{(\lambda,L)},\mathtt{tx}')\\ &(\mathtt{tx.in.com},\mathtt{tx.out.com}) \stackrel{?}{\neq} (\mathtt{tx}'\mathtt{.in.com},\mathtt{tx}'\mathtt{.out.com})\\ &\quad \wedge \mathtt{tx.header}.\Delta \stackrel{?}{=} \mathtt{tx}'\mathtt{.header}.\Delta\end{aligned}$$

Theorem 4. TX *is complete, computationally theft-resistant, zero-knowledge, and immutable if Approx-SISs of* $(n, m, q, \gamma, \gamma', N)$ *and* $(n, m-3, q, \gamma, \gamma', N)$ *are hard (see the security proofs in Appendix C).*

4. Implementation

We implement a C library ref. [25] for LACT+ aiming to achieve 128-bit security. Therefore, a root-Hermite factor $\delta = 1.004$ was used for lattices implementations allowing a gap for future attacks ref. [37]. A prime-order multiplicative group $\mathbb{H}$ with at least multiplicative 2^{384} elements was used for membership proofs. The prime order of $\mathbb{H}$ is q_2 such that $(q_2 - 1)/2 > 2^{3\lambda}$ is also prime (q_2 = 3a2c6ad1f4ef4084fbf76e7c6201b32850c57c408a6e0c4a6cda6c290c61e6dadd4e6b7312dd3aa6bd610a917c1d42f03). The concrete parameters used for tests are defined in Table 2. According to the parameters, a coin record is 29.9 KB (a commitment and a range proof), and a public key is 5760 Bytes.

Table 2. Concrete Parameters and Variables.

Parameter	Value	Description
(n, m)	(6, 4)	Approx-SIS hard
N	256	Polynomial Size
q	$2^{44} - 2^{14} + 1$	$q = 1 \pmod{2N}$
$L, 2^{L'}$	64, 15	Ranges
γ	2^{36}	Short-Vector Norm
γ'	2^{36}	Error Norm
β	60	$\log_2 \binom{N}{\beta} + \beta \geq 256$
$\log(p_1), \log(p_2)$	17, 29	$< \log(\gamma'), < \log(\gamma')$
$\alpha(+\lceil \log_2(\lvert L' \rvert/2) \rceil)$	2^9 to 2^{11}	maximum 16 in/outputs
τ, τ_1, τ_2	$2^4 - 1, 2^7 - 1, 2^{28} - 1$	$< \gamma$
τ_3	2^{16}	$< 2^{L'}\gamma$
χ	60	$\lVert \cdot \rVert_1$ Norm of Hints
$\lvert \vec{x} \rvert$	48 bytes	SHAKE256 ref. [38]
$\lvert \Delta \rvert$	49 bytes	Activity Proofs
$\mathtt{highbits}_{p_1}(\cdot)$	5.7 KB	Packed High-bits
$\mathtt{highbits}_{p_2}(\cdot)$	3.1 KB	Packed High-bits

We use two multiplications: number theoretic transform (NTT) for regular polynomials and an easy multiplication when one of the polynomials is a $\mathtt{rot}(*, *)$ (see below). The C library uses 64-bit integers except for the NTT, which uses 128-bit integers. In NTT multiplication, we convert polynomials to NTT space using Cooley–Tukey butterflies ref. [39],

point-wise multiply them using Montgomery point-wise multiplications ref. [40], and then convert them back using inverse-transform from Gentleman–Sande butterflies ref. [41].

Recall that $\texttt{rot}(v_i, i)$ is a polynomial where coefficient i is v_i, and all other coefficients are zero. Therefore, a multiplication of $\vec{a}\texttt{rot}(v_i, i)$ is computed as follows: convert coefficients in $[N - i, N)$ into their negative values, rotate all coefficients by i, and multiply all coefficients by v_i. Because this is faster than NTT multiplication, we use this easy multiplication whenever possible, e.g., most of the short-norm vectors in CARRY.

We chose bits' masking keys from $[-\alpha, \alpha]$ such that some functional polynomials like Step 29 in COIN and Step 53 in CARRY can be larger than γ. However, there is a high probability that they will be in the accepted range due to centrally reduced numbers. Therefore, we perform norm checks during the generation of COIN and CARRY to ensure completeness.

All the benchmarks were configured as follows: The maximum coin amount and the initial coinbase value are $2^{64} - 1$. Here, we use "aggregated size" to denote the database size, which is slightly larger than theoretical sizes due to indexing and identifiers. "Deleted size" means the exact size of deleted coin records as explained in Section 3.1. Therefore, "unaggregated size" is computed as the summation of the aggregated and deleted sizes. The verification times are measured on an i7-1065G7 CPU at 1.30GHz. Furthermore, $[x : y]$ denotes that the number of inputs and outputs are randomly chosen from $[1, x]$ and $[1, y]$.

Efficiency of New Carry Proofs. First, we run benchmarks to get the sizes of new carry proofs and LACT carry proofs. Recall that LACT creates multiple LACT transactions if the numbers of input coins and out coins are larger than two. Therefore, the size impact of carry proofs to the total blockchain is proportional to the number of transactions, whereas in LACT+, this impact is now logarithmic. We simulate carry proof benchmarks to see this size and verification time impact for normal transactions where the input size can be larger than two. Figures 5 and 6 show the accumulated sizes and total verification times for carry proof(s) of LACT and LACT+. From the graphs, we conclude that new LACT+ carry proofs are more efficient than LACT carry proofs. Therefore, the whole system is more efficient than prior post-quantum CT schemes, from unaggregated ones to aggregated ones like LACT, in spite of its other added benefits such as fully trustless verifiable immutability.

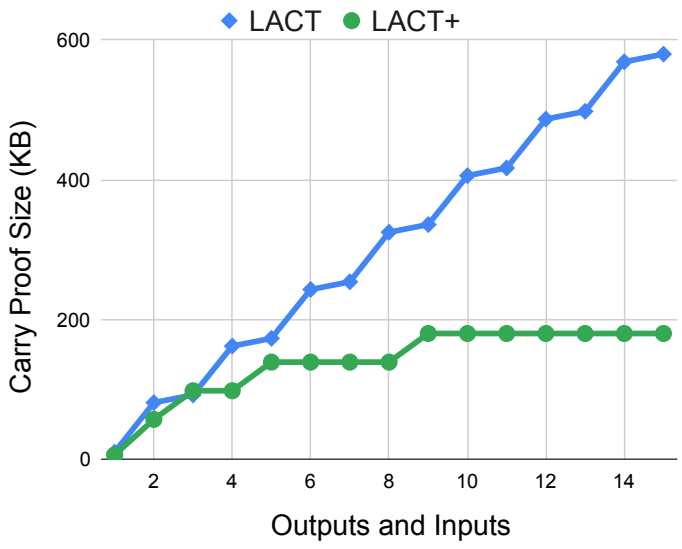

Figure 5. Carry proof sizes.

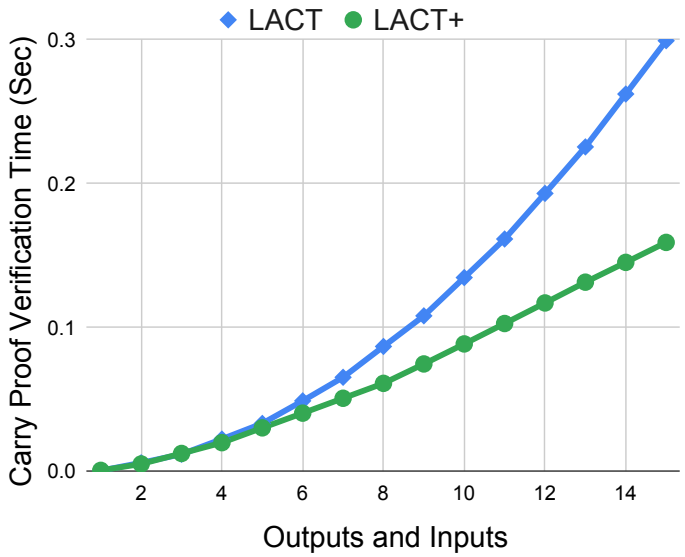

Figure 6. Carry proof verification times.

Impact of Aggregation. The main efficiency mechanism of LACT+ is transaction aggregation. The impact of the transaction aggregation changes according to input/output rates because more unspent outputs lead to larger blockchains. We ran benchmarks for different input/output rates, and we show the results in Figure 7. From the graph, we see that even if the output rate is higher than the input rate, aggregation achieves significant reductions in size. Verification times for these benchmarks are shown in Figure 8.

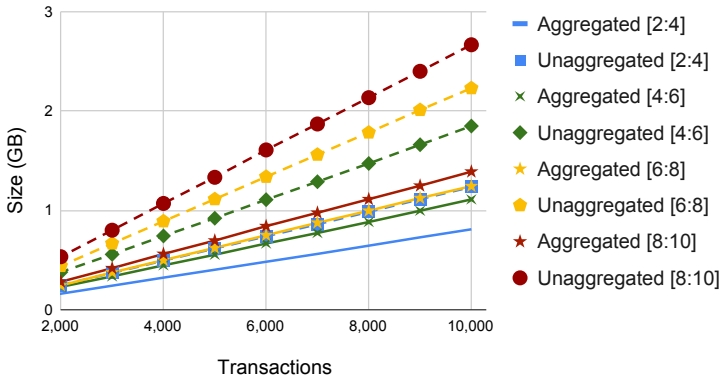

Figure 7. Aggregated sizes and unaggregated sizes vs. transactions.

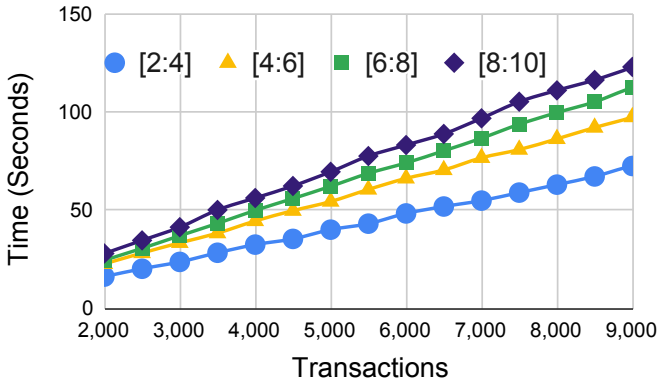

Figure 8. Verification times vs. transactions for different input and output rates.

A Theoretical Comparison with MatRiCT+ Esgin et al. proposed ref. [22], an efficient lattice-based Ring CT protocol that uses the Chinese Remainder Theorem (CRT) packing, e.g., packing all L number of $\vec{z}$ into a single polynomial. Even without CRT packing, LACT+ is more efficient than MatRiCT+ as illustrated in Figure 9.

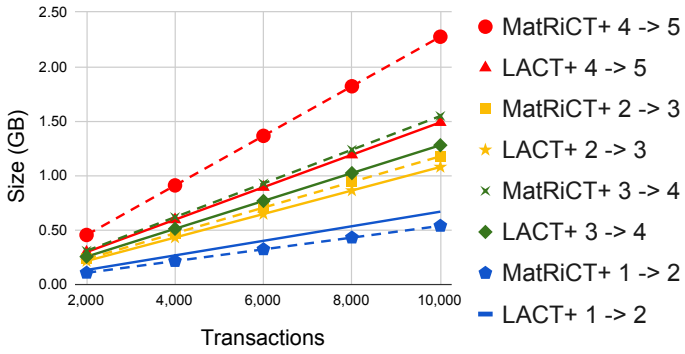

Figure 9. LACT+ and MatRiCT+: For MatRiCT, Coins (`cn`) are 4.48 KB, public keys are 3.4 KB, and $(2 \rightarrow 2)$ proofs are 47 KB at 1/11 anonymity.

5. Conclusions

Aggregable confidential transactions are more private and scalable than typical transactions. However, the scalability of previous aggregable transactions does not provide trustless verification at the consensus level because real-world transactions should be immutable. LACT+ post-quantum aggregable confidential transactions solve this problem by adding aggregable activity proofs. Equally importantly, LACT+ is more efficient and faster than existing post-quantum aggregable transactions due to our use of approximate SIS and logarithmic carry proofs. LACT+ is the first practical post-quantum aggregable confidential transaction protocol that provides consensus-level aggregation and fully trustless verification.

Author Contributions: Conceptualization, J.A.; Writing-original draft, J.A.; Writing—review, editing, X.B. and M.M.; Supervision, X.B. and M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Security Proofs of Confidential Coins

Binding. We claim `COIN` is binding because finding two openings $(v, \vec{k}) \neq (v', \vec{k}')$ for the same coin record means the adversary finds a solution to the Approx-SIS problem such that

$$\vec{s} = [\texttt{binary}(v), \vec{k}, \vec{0}^{m-2}] - [\texttt{binary}(v'), \vec{k}', \vec{0}^{m-2}] \neq \vec{0}^{m}$$
$$\|\vec{s}\| \leq \gamma \text{ and } \texttt{highbits}_{\gamma'}(\vec{H}\vec{s}) = \vec{0}^{n} \in \mathcal{R}_q^n.$$

Knowledge Soundness. Let there be a p.p.t. algorithm that breaks the knowledge soundness such that the extractor $\mathcal{E}$ outputs $\vec{b}$ of number not in $[0, 2^L)$ for a valid proof. According to the verification in Step 44, the algorithm finds $\vec{s}$ for $[\vec{z}_i = \vec{x}_2 \texttt{rot}(b_i, i) + \vec{a}_i]_{i=0}^{L'-1} \in \mathcal{R}^L$ such that b_i is not 0 or 1, but

$$\texttt{highbits}_{\gamma'}(\vec{H}\vec{s}) \overset{?}{\approx} \texttt{highbits}_{\gamma'}(\vec{x}_2(\vec{x}_1\vec{u} + \vec{t}_1) + \vec{t}_2) \in \mathcal{R}_q^n.$$

Because b_i is not 0 or 1, the following is true,

$$\texttt{highbits}_{\gamma'}(\vec{x}_2^2\Big[\underbrace{\vec{H}\,[\vec{0},\sum_{i=0}^{L}(\texttt{rot}(b_i(b_1-1),i)),\vec{0}^{m-2}}_{\vec{h}\neq\vec{0}^m}\Big]) = \vec{0}^n \in \mathcal{R}_q^n.$$

Here, $\vec{h}$ is a solution to the Approx-SIS problem. Therefore, we claim range proofs are knowledge sound.

Zero-Knowledge Argument of Range Proofs. $\vec{z}_i$ statistically hides b_i because $\vec{z}_i = b_i\vec{x}_2 + \vec{a}_i \in [-(\alpha-1),(\alpha-1)]$ when the range of $\vec{a}_i$ is $[-\alpha,\alpha]$. We claim the computational hiding for commitments $(\vec{u},\vec{t}_1,\vec{t}_2)$ of `COIN` due to dropped bits of Approx-SIS and sufficient randomness of $\vec{k},\vec{r}_1,\vec{r}_2$ (see ref. [26]). If there is a p.p.t. algorithm that breaks the zero-knowledge argument by successfully distinguishing the generated proofs over the simulated proofs, then the algorithm can be used to break the hiding property of the commitments; in other words, solving the Approx-SIS problem. Therefore, we claim that the range proofs satisfy the zero-knowledge argument.

Finally, we prove that Theorem 2 is true because `COIN` is complete (Figure 3), binding, knowledge-sound, and zero-knowledge.

Appendix B. Security Proofs of Carry Proofs

Knowledge Soundness If there is an algorithm that breaks the knowledge soundness of `CARRY` then it creates

$$\texttt{highbits}_{\gamma'}\left(\vec{x}_2^2\vec{H}\underbrace{\begin{bmatrix}\vec{0},\sum_{l=0}^{L_{|\texttt{out}|}}\sum_{i=0}^{L}(\texttt{rot}(c'_{1,i}(c'_{1,i,l}-1),iL_{|\texttt{out}|}+l))\\ \sum_{l=0}^{L_{|\texttt{in}|}}\sum_{i=0}^{L}(\texttt{rot}(c'_{0,i,l}(c'_{0,i}-1),iL_{|\texttt{in}|}+l)),\vec{0}^{m-3}\end{bmatrix}}_{\vec{h}\neq\vec{0}^m}\right) = \vec{0}^n \qquad \text{(A1)}$$

for invalid binary vectors of carries; $\boldsymbol{c}'_0$ and $\boldsymbol{c}'_1$ which are not 0 nor 1 (see Step 22).

Here, $\vec{h}$ is a solution to Approx-SIS problem. Therefore, we conclude that `CARRY` is knowledge sound.

Zero-Knowledge Argument. Due to the rejection sampling, $\vec{z}_0$ and $\vec{z}_1$ statistically hide bits of both input carries and output carries. Furthermore, commitments $\vec{u}$, $\vec{t}_1$ and $\vec{t}_2$ hide their short vectors due to the Approx-SIS problem. Therefore, we claim `CARRY` holds the zero-knowledge argument property.

Finally, we conclude Theorem 3 is accurate due to the above proofs. Or `CARRY` is complete (Figure 4), computationally knowledge sound, and holds the zero-knowledge argument property.

Appendix C. Security Proofs of Aggregable CT

Theft Resistance. If a p.p.t. algorithm wins the game of theft resistance, then the algorithm creates a transaction with a valid signature unknowing the secret key. In other words, the algorithm breaks EUF-CMA of `SIG` defined in Section 2.6, knowledge soundness of `COIN`, or knowledge soundness of `CARRY`. Therefore, if `COIN`, `CARRY`, and `SIG` are secure, then `TX` is theft resistant.

Zero-Knowledge and Knowledge Soundness. Suppose a p.p.t. algorithm creates a chain with more or less unspent coins than the supplied coins. In that case, the algorithm either breaks `COIN`'s knowledge soundness, `CARRY`'s knowledge soundness, or forges a signature for a public key with non-zero beginning polynomials. Therefore, we conclude that `TX` is knowledge soundness when `COIN`, `CARRY`, and `SIG` are secure. We directly claim the zero-knowledge of `TX` because `COIN`, `CARRY`, and `SIG` are zero-knowledge.

Immutability. `TX`' immutability states that no p.p.t algorithm can find two different valid transactions with the same membership proof but with different in/out commitment sets.

If an algorithm wins the game of immutability, then the algorithm solves the group collision resistance problem (GCR). Therefore, if GCR is computationally hard, TX is computationally immutable.

Therefore, we claim Theorem 4 is true or TX is complete, computationally theft-resistant, zero-coin generating, and immutable.

References

1. Zhang, H.; Zhang, F.; Wei, B.; Du, Y. Implementing confidential transactions with lattice techniques. *IET Inf. Secur.* **2019**, *14*, 30–38. [CrossRef]
2. Alupotha, J.; Boyen, X.; Mckague, M. Aggregable Confidential Transactions for Efficient Quantum-Safe Cryptocurrencies. *IEEE Access* **2022**, *10*, 17722–17747. [CrossRef]
3. Androulaki, E.; Karame, G.O.; Roeschlin, M.; Scherer, T.; Capkun, S. Evaluating User Privacy in Bitcoin. In Proceedings of the Financial Cryptography and Data Security, Okinawa, Japan, 1–5 April 2013; Sadeghi, A.R., Ed.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 34–51.
4. Spagnuolo, M.; Maggi, F.; Zanero, S. BitIodine: Extracting Intelligence from the Bitcoin Network. In Proceedings of the Financial Cryptography and Data Security, Okinawa, Japan, 1–5 April 2013; Christin, N., Safavi-Naini, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 457–468.
5. Fleder, M.; Kester, M.S.; Pillai, S. Bitcoin transaction graph analysis. *arXiv* **2015**, arXiv:1502.01657.
6. Reid, F.; Harrigan, M. An Analysis of Anonymity in the Bitcoin System. In Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, Boston, MA, USA, 9–11 October 2011; pp. 1318–1326. [CrossRef]
7. Herrera-Joancomartí, J. Research and Challenges on Bitcoin Anonymity. In *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*; Garcia-Alfaro, J., Herrera-Joancomartí, J., Lupu, E., Posegga, J., Aldini, A., Martinelli, F., Suri, N., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 3–16.
8. Khalilov, M.C.K.; Levi, A. A survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2543–2585. [CrossRef]
9. Morris, L. Anonymity Analysis of Cryptocurrencies. Master's Thesis, Rochester Institute of Technology, Rochester, NY, USA, 2015. Available online: https://scholarworks.rit.edu/theses/8616/ (accessed on 10 January 2023).
10. Jedusor, T.E. Mimblewimble. 2016. Available online: https://docs.beam.mw/Mimblewimble.pdf (accessed on 10 January 2023).
11. Poelstra, A. Mimblewimble. 2016. Available online: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf (accessed on 10 January 2023).
12. Poelstra, A.; Back, A.; Friedenbach, M.; Maxwell, G.; Wuille, P. Confidential assets. In Proceedings of the International Conference on Financial Cryptography and Data Security, Nieuwpoort, Curaçao, 26 February–2 March 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 43–63.
13. Fuchsbauer, G.; Orrù, M.; Seurin, Y. Aggregate Cash Systems: A Cryptographic Investigation of Mimblewimble. In Proceedings of the Advances in Cryptology—EUROCRYPT 2019, Darmstadt, Germany, 19–23 May 2019; Ishai, Y., Rijmen, V., Eds.; Springer: Cham, Switzerland, 2019; pp. 657–689.
14. Alupotha, J.; Boyen, X.; Foo, E. Compact Multi-Party Confidential Transactions. In Proceedings of the Cryptology and Network Security, Vienna, Austria, 14–16 December 2020; Krenn, S., Shulman, H., Vaudenay, S., Eds.; Springer: Cham, Switzerland, 2020; pp. 430–452.
15. IBM-Research. IBM's Roadmap for Scaling Quantum Technology. Available online: https://research.ibm.com/blog/ibm-quantum-roadmap (accessed on 21 March 2022).
16. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 10 January 2023).
17. Wood, G. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*; Ethereum project yellow paper; Ethereum: Zug, Switzerland, 2014; Volume 151, pp. 1–32.
18. Noether, S.; Mackenzie, A.; the Monero Research Lab. Ring confidential transactions. *Ledger* **2016**, *1*, 1–18. [CrossRef]
19. Sun, S.F.; Au, M.H.; Liu, J.K.; Yuen, T.H. RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero. In Proceedings of the Computer Security—ESORICS 2017, Oslo, Norway, 11–15 September 2017; Foley, S.N., Gollmann, D., Snekkenes, E., Eds.; Springer: Cham, Switzerland, 2017; pp. 456–474.
20. Esgin, M.F.; Zhao, R.K.; Steinfeld, R.; Liu, J.K.; Liu, D. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 567–584.
21. Alberto Torres, W.; Kuchta, V.; Steinfeld, R.; Sakzad, A.; Liu, J.K.; Cheng, J. Lattice RingCT V2.0 with Multiple Input and Multiple Output Wallets. In Proceedings of the Information Security and Privacy, Prague, Czech Republic, 23–25 February 2019; Jang-Jaccard, J., Guo, F., Eds.; Springer: Cham, Switzerland, 2019; pp. 156–175.
22. Esgin, M.F.; Steinfeld, R.; Zhao, R.K. MatRiCT+: More Efficient Post-Quantum Private Blockchain Payments. *Cryptol. ePrint Arch.* **2021**, *545*, 1–21.

23. Grin tech.org. Minimal Implementation of the MimbleWimble Protocol. Available online: https://github.com/mimblewimble/grin (accessed on 27 January 2021).
24. Scalable Confidential Cryptocurrency—MimbleWimble Implementation. Available online: https://www.beam.mw/ (accessed on 27 January 2021).
25. Alupotha, J. LACT+: Post-Quantum Aggregable Confidential Transactions. 2022. Available online: https://github.com/jaymine/LACTv2 (accessed on 11 January 2023).
26. Chen, Y.; Genise, N.; Mukherjee, P. Approximate trapdoors for lattices and smaller hash-and-sign signatures. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, 8–12 December 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 3–32.
27. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **2009**, *56*, 34. [CrossRef]
28. Ajtai, M. Generating hard instances of lattice problems. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; ACM: New York, NY, USA, 1996; pp. 99–108.
29. Alupotha, J.; Boyen, X. Origami Store: UC-Secure Foldable Datachains for The Quantum Era. *IEEE Access* **2021**, *9*, 81454–81484. [CrossRef]
30. Noether, S.; Noether, S. Monero Is Not That Mysterious. Technical Report. 2014. Available online: https://web.getmonero.org/ru/resources/research-lab/pubs/MRL-0003.pdf (accessed on 11 January 2023).
31. Maxwell, G. Confidential Transactions. 2015. Available online: https://people.xiph.org/greg/confidential_values.txt (accessed on 11 January 2023).
32. Fiat, A.; Shamir, A. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Proceedings of the Advances in Cryptology—CRYPTO' 86, Santa Barbara, CA, USA, 1 January 1987; Odlyzko, A.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1987; pp. 186–194.
33. Pointcheval, D.; Stern, J. Security arguments for digital signatures and blind signatures. *J. Cryptol.* **2000**, *13*, 361–396. [CrossRef]
34. Abdalla, M.; An, J.H.; Bellare, M.; Namprempre, C. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In Proceedings of the Advances in Cryptology—EUROCRYPT 2002, Amsterdam, The Netherlands, 28 April–2 May 2002; Knudsen, L.R., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 418–433.
35. Lyubashevsky, V. Lattice-Based Identification Schemes Secure Under Active Attacks. In Proceedings of the Public Key Cryptography—PKC 2008, Barcelona, Spain, 9–12 March 2008; Cramer, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 162–179.
36. Lyubashevsky, V. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In Proceedings of the Advances in Cryptology—ASIACRYPT 2009, Tokyo, Japan, 6–10 December 2009; Matsui, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 598–616.
37. Albrecht, M.R. LWE Estimator. Available online: https://lwe-estimator.readthedocs.io/en/latest/readme_link.html (accessed on 22 October 2021).
38. Gleen, M.L. Device for and Method of One-Way Cryptographic Hashing. U.S. Patent 6829355, 12 July 2004.
39. Gauss, C.F. Nachlass: Theoria interpolationis methodo nova tractata. *Carl Friedrich Gauss Werke* **1866**, *3*, 265–327.
40. Montgomery, P.L. Modular multiplication without trial division. *Math. Comput.* **1985**, *44*, 519–521. [CrossRef]
41. Gentleman, W.M.; Sande, G. Fast Fourier transforms: For fun and profit. In Proceedings of the Fall Joint Computer Conference, San Francisco, CA, USA, 7–10 November 1966; pp. 563–578.