IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 14, 2021, accepted January 5, 2022, date of publication February 7, 2022, date of current version February 17, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3149605*

# Aggregable Confidential Transactions for Efficient Quantum-Safe Cryptocurrencies

**JAYAMINE ALUPOTHA, XAVIER BOYEN, AND MATTHEW MCKAGUE**
School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
Corresponding author: Jayamine Alupotha (alupotha@qut.edu.au)

**ABSTRACT** Confidential Transactions (CT) hide coin amounts even from verifiers without the help of trusted third parties. Aggregable CTs are a scalable category of CTs with ''spent coin record trimming''. For example, if Alice sends coins to Bob, who had sent similar coins to Charles, the aggregated transaction shows only that Alice sent coins to Charles by deleting Bob's coin records. Since the number of spent coin records grows linearly with the number of transactions, faster than the number of accounts, cash systems based on aggregable CTs are highly scalable. However, existing quantum-safe aggregable CT protocols have large unspent coin records, and existing efficient aggregable CTs are vulnerable to quantum attacks. We introduce two aggregable CT protocols, based on new efficient homomorphic zero-knowledge proofs, from either the plain or Module Short Integer Solution (SIS and MSIS) problems, both believed to be secure against quantum adversaries. We further implement the MSIS-based aggregable CT protocol as a C library. Our experiments on $10^4$ transactions show that aggregation reduces the cash system's size by 40%–54% when the output/input rate is in the range 1/1–2/1. For example, a cash system of 1.73 GB can be reduced to 0.98 GB when the output/input rate is 1.5, which has been the historical real-world average rate.

**INDEX TERMS** Confidential transactions, transaction cut-through, scalable cryptocurrencies, PQ ZK.

## I. INTRODUCTION

Zero-Knowledge (ZK) proofs are exhaustively used in multi-party distributed systems to preserve privacy while maintaining public verification. More theoretically, ZK proofs can verify statement(s) regarding concealed information without revealing it. For example, a ZK range proof of a commitment[1] shows that the committed value is in a specific range without revealing the value. Another example is ZK summation proofs, which verify the summation of committed values without revealing the committed values individually. Privacy-preserving cryptocurrencies heavily use these ZK proofs due to this verifiable yet concealed information.

Early cryptocurrencies reveal coin amounts since the coins are linked to pseudonymous identities. However, many studies [1]–[7] show that pseudonymous identities are linkable to real-world identities, creating a demand for Confidential Transaction (CT) protocols as the popularity of decentralized cash systems is growing. Typical CTs present coins as commitments. Since coin amounts are hidden in commitments, CTs contain ZK range proofs and ZK summation proofs:

- ZK range proofs, to ensure that committed coin amounts are not negative, nor creating overflows;
- ZK summation proofs, to ensure that the sum of sent amounts balances the sum of received amounts—the ''*zero-coin generation*'' property.

Decentralized multi-party systems like blockchains can be categorized into (1) *stateful systems* and (2) *stateless systems*.[2] Stateful cash systems [8]–[12] and stateful data systems [13] must preserve everything, including spent coin bundles and stale data records. In these systems the entirety of each transaction is hashed and taken as an input for the cash system's verification. Therefore, removing records from stateful systems can lead to theft and inconsistent consensuses.[3] On the other hand, *stateless* systems allow secure optional deletion for spent coin records and old data records with

The associate editor coordinating the review of this manuscript and approving it for publication was Lo'ai A. Tawalbeh.

[1]Each commitment is associated with a value and a secret masking key. Masking keys are also known as blinding factors since they add randomness required to hide the coin amount. Commitments allow hiding a value or a string and disclosing the hidden value/string when required. Commitments' binding property ensures that the disclosed value/string is the originally committed value/string. In other words, finding another value/string for the same commitment is computationally infeasible.

[2]Better terms for ''stateless'' would be ''*history-free*'' or ''*zero-history*''.

[3]The only way to remove a stateful system's data is to rely on trusted third parties like in Simple Payment Verification (SPV) [14].


This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

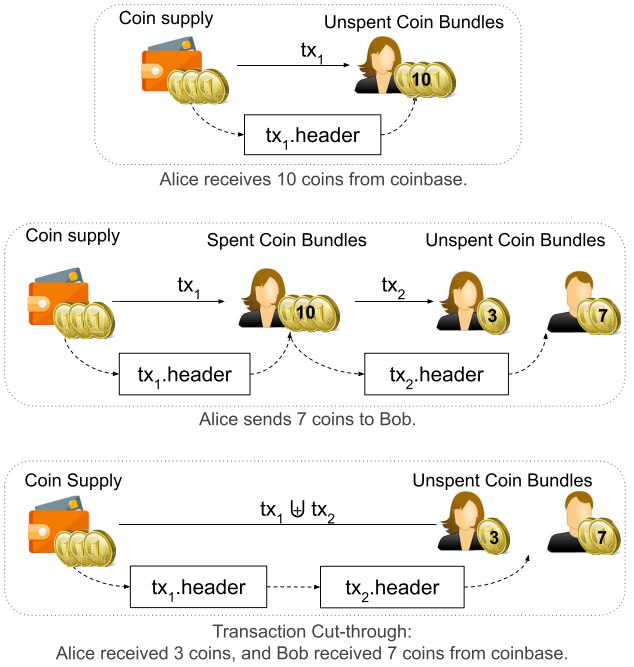



**FIGURE 1.** Transaction aggregation.

the help of homomorphic cryptographic protocols like homomorphic commitments and homomorphic digital signatures. For example, stateless cash systems like Mimblewimble [15]–[17] and stateless data systems like Origami datachains [18] aggregate transactions by safely deleting spent coin bundles and stale data records.

There are two CT protocol classes: (1) Ring CT and (2) aggregate CT protocols. *Stateful* cash systems like [10] are built on Ring CTs [19]–[21] where transactions obfuscate true senders and receivers by mixing them with irrelevant participants, in an attempt to enhance privacy. *Stateless* cash systems depend on aggregable CTs [22] to safely trim spent coin records, also boosting privacy. We illustrate an example in Figure 1. Statistics [23] show that approximately 87% of coin records are spent. As a result, stateless cash systems based on aggregable CTs should be both private and scalable.

Additively homomorphic commitments can be added together without changing the committed values. Let there be two commitments of $[v_1, r_1]$ and $[v_2, r_2]$ with values $(v_1, v_2)$ and secret masks $(r_1, r_2)$. Then there is a *public* function which outputs a commitment of $[v_1 + v_2, r_1 + r_2]$ by adding those commitment without accessing their values or masks. The existing efficient CTs like Maxwell's CTs [22] use Pedersen commitments [24] which are inherently homomorphic with no bounds on the number of commitments that can be added. Due to the homomorphic properties of Pedersen commitments, ZK summation proofs can be implemented directly. However, being based on the Discrete-Log Problem (DLP), these protocols are vulnerable to quantum attacks.

*Conjectured quantum-safe*[4] substitutes for Pedersen commitments could be *element-wise* homomorphic[5] commitments based on the Short Integer Solution (SIS) problem, where the coin amount and the masking key are encoded as a "short" (i.e., low-norm) vector whose norm $\gamma \ll q$ against some fixed modulus $q$. For example, in [25], short-norm vectors are $[v, r_1, r_2..]$ for some coin amount $v$ and masking key $[r_1, r_2..]$. However, the price of quantum-safety is often a combination of inefficiency and limitations on the number and values of commitments that can be added. For example, [25] works with lattice modulus $q \gg 2^{64} \cdot 10^6$ for coin values in $[0, 2^{64})$ and up to $10^6$ unspent coin bundles.

One tempting solution would be to commit to coin amounts as a binary expansion rather than an integer, i.e., as a short vector $[b_0, b_1, .., b_{63}, r_1, r_2, ..])$ to signify $v = \sum_{i=0}^{63} 2^i b_i$. This would allow the use of a small modulus. However, we lose homomorphic properties since the resulting commitments are not globally but only element-wise homomorphic, incompatible with the direct implementation of ZK summation proofs. This is our approach, and we show how to make this work efficiently.

### A. OUR CONTRIBUTION

We introduce the first *Lattice-based Aggregable Confidential Transactions* with binary commitments which are based on the SIS problem and the MSIS problem. Our CTs have the following properties.

- **Confidentiality:** Our CTs hide the coin amounts of all coin bundles except the coins from the coinbase.[6]
- **Scalability:** We can aggregate any number of transactions by removing spent coin records. Hence the stateless cash systems built on our CTs are highly scalable.
- **Efficiency:** Our CTs always store coin amounts in their binary format. As a result, a smaller lattice modulus $q$ can be used than when the direct values are stored, e.g., our MSIS-based CT modulus is $2^{50} - 2^{14} + 1$ even though coin values are in $[0, 2^{64})$.

Moreover, we implement a C library for our MSIS-CT protocol to experimentally determine the size reductions for different input and output rates. The results are stated in Table 1 for [input:output] proportions; [2:2], [2:3], and [2:4], i.e., for $[x, y]$, the numbers for inputs and outputs are randomly chosen from $[1, x]$ and $[1, y]$, respectively. Note that we choose these proportions due to the average real-world [input:output] proportion being about 1.5 in Bitcoin [23].

#### 1) ROADMAP

We explain the basic idea of our protocols in Section II and Section III without deep diving into Lattice-based Cryptography. Then we discuss related work in Section IV. Preliminaries such as commitments, digital signatures, and their security

[4] *Quantum-Safe* a.k.a. *Post-Quantum* (PQ) is in the conjectured sense.

[5] Homomorphism means that adding two commitments of $[v, r_1, r_2, ..]$ and $[v', r_1', r_2', ..]$ yields a commitment of $[v + v', r_1 + r_1', r_2 + r_2', ..]$.

[6] The *coinbase* records the number of possible total coins minus the number of coins currently in circulation.

**TABLE 1. Space savings of transaction aggregation with different [input:output] rates.**

| [input:output] | CTs | Size | Aggregated Size | Saving |
|---|---|---|---|---|
| [2 : 2] | 10000 | 1.15 GB | 0.54 GB | 53.5% |
| [2 : 3] | 10000 | 1.73 GB | 0.98 GB | 43.6% |
| [2 : 4] | 10000 | 2.39 GB | 1.40 GB | 41.3% |

properties are discussed in Section V. After that, we explain the expected security properties of aggregable transactions in Section VI and Section VII. Our SIS-based aggregable CT protocol and its security proofs are given in Section VIII and Section IX. Then we explain the MSIS-based aggregable CT protocol and its security proofs in Section X and Section XI. The MSIS-based CT implementation and its experimental results are discussed in Section XIII. Finally, we summarize the paper in Section XIV.

## II. STATIC CARRIES

As we discussed before, lattice-based commitments are *element-wise* homomorphic, yet we want to commit the binary forms of coin values to reduce the lattice modulus. Making the task more complex, we want to remove spent coin records and still check the number of circulating coins in the system. In this section, we explain how to use the binary form of the coins while completing these tasks.

Our solution for ZK summation proof is to use the binary form of coin values with **static** carries. Let $[v_1, \ldots, v_u] \subseteq [0, 2^L)$ for $\mathtt{u} \in \mathbb{Z}^+$ be coin different coin values where $v_i = \sum_{j=0}^{L-1} b_{i,j}2^j$. We want to compute their summation *out*. If the binary form of *out* is $[o_0, .., o_L]$, we can create carries $\boldsymbol{c} = [c_0 = 0, c_1, .., c_{L-1}, c_L = 0]$ such that

$$\left[0, \left(\sum_{i=1}^{\mathtt{u}} b_{i,0} + c_0\right) \div 2, .., \left(\sum_{i=1}^{\mathtt{u}} b_{i,L-2} + c_{L-2}\right) \div 2, 0\right]$$

when $\div$ is the integer division, e.g., $3 \div 2 = 1$. Note that each $c_* \in [0, \lfloor \mathtt{u}/2 \rfloor]$ and $c_0 = c_L = 0$. The carries are the amounts that need to be carried from one column to the next when performing addition, as shown below for base 2.

| $c_L = 0$ | $c_{L-1}$ | $c_{L-2}$ | .. | $c_1$ | 0 |
|---|---|---|---|---|---|
| | $b_{1,(L-1)}$ | $b_{1,(L-2)}$ | .. | $b_{1,1}$ | $b_{1,0}$ |
| | .. | .. | .. | .. | .. |
| | $b_{\mathtt{u},(L-1)}$ | $b_{\mathtt{u},(L-2)}$ | .. | $b_{\mathtt{u},1}$ | $b_{\mathtt{u},0}$ |
| | $o_{L-1}$ | $o_{L-2}$ | .. | $o_1$ | $o_0$ |

We will explain our approach using a simple example. Here, we restrict the maximum number of inputs and outputs to 2 per transactions, i.e., transactions with $1 \Rightarrow 2$, $2 \Rightarrow 1$, $2 \Rightarrow 2$ (inputs $\Rightarrow$ outputs) are possible. Any other complex transaction can be accomplished using a combination of the above transactions. Hence, this restriction does not impact the cash system's practicality.

As shown in Figure 1, the aggregated cash system only has unspent coin bundles and transaction headers since the aggregation removes all spent coin bundles. Therefore, a stateless cash system has two tables, `Ucoins` for unspent coin bundles and `Headers` for transaction headers. At this stage, we only explain *the correctness of element-wise additions*, not the security, which will be added in the next section.

### A. STEP 1: COINBASE ACCOUNT

The initial cash system only has the coinbase in `Ucoins` where the coinbase has the maximum number of coins, $S$. We store the maximum coin amount in binary format. The coinbase account can transfer coins to other accounts as rewards, and the balance after the transaction will be stored in binary format. Note that these coins do not belong to any participant until they are awarded. Our supplying technique is different from typical cash systems, where they start with zero supply and keep adding supply coins to the system. However, we cannot use their method since we have to fix the circulating coins in the system.[7]

### B. STEP 2: TRANSACTIONS' ZERO-COIN GENERATION

When the $t^{th}$ transaction happens with input coins $[v_i = \sum_{j=0}^{L-1} 2^j b_{i,j}]_{i=1}^{|\mathtt{in}|}$ and output coins $[v'_i = \sum_{j=0}^{L-1} 2^j b'_{i,j}]_{i=1}^{|\mathtt{out}|}$, we compute input values' carries $\boldsymbol{c}_0^{(t)}$ and output values' carries $\boldsymbol{c}_1^{(t)}$ and store them in the transaction header table. If the binary forms and carries correspond to a valid transaction, they will satisfy the following equation for all $j \in [0, L)$:

$$\boxed{\sum_{i=1}^{|\mathtt{out}|} b'_{i,j} - \sum_{i=1}^{|\mathtt{in}|} b_{i,j} + c_{1,j}^{(t)} - 2c_{1,j+1}^{(t)} - c_{0,j}^{(t)} + 2c_{0,j+1}^{(t)} \stackrel{?}{=} 0} \tag{1}$$

Here, "$|\cdot|$" states the number of elements in an array, column, or rows of a table. Note that each $c_{*,*}^{(t)} \in [0, 1]$ — since there are at most two inputs and outputs — and $c_{*,0}^{(t)} = c_{*,L}^{(t)} = 0$.

### C. STEP 3: CASH SYSTEMS' ZERO-COIN GENERATION

The **novelty** of our protocols is that we propose Equation (2) where we can verify the aggregated cash system's circulating coin amount without the spent coin bundles using. A valid state (`Ucoins`, `Headers`) will satisfy the following for all $j \in [0, L)$:

$$\boxed{\sum_{i=1}^{|\mathtt{Ucoins}|} b_{i,j} + \sum_{t=1}^{|\mathtt{Headers}|} \left(c_{1,j}^{(t)} - 2c_{1,j+1}^{(t)} - c_{0,j}^{(t)} + 2c_{0,j+1}^{(t)}\right) \stackrel{?}{=} S_j} \tag{2}$$

This follows by summing (1) over all transactions, noting that spent coins will be cancelled out.

These static carries have multiple advantages.

- Spent coin records can be removed from the system since Equation (2) does not need spent coin records.

[7]The technical reason for fixing the initial coin supply ($S$) is that this way carries are static and previously computed carries will not need to be changed with new transactions. If the total coin of the cash system changes over time, we get volatile carries where previous carries are no longer valid. In that case, all the participants in the system have to join for all transactions even though they are not senders nor the receivers of that particular transaction, which makes the cash system impractical. MatRiCT [21] uses these volatile carries and cannot delete spent coin records freely.

- Only the parties involved in the transaction need to compute the carries since Equation (1) only takes the transaction's inputs, outputs, and carries.
- We easily enforce the carries to be in [0, 1], which will be useful for the range proof computation of the carries.
- *No* input (or output) carries need to be calculated if there is only one input (or output) since those carries are always zeros. This reduces transaction size and verification time drastically.

We explain this by taking a simple example. Assume the maximum coin amount is $2^4 - 1$ coins and $L = 4$. Note that we do not have to store $c_{0,4}^{(t)}$ and $c_{1,4}^{(t)}$ since they are zeros; however, we store them in these introductory sections to give a clear explanation.

Initially, only the coinbase has coins.

| Ucoins |
|---|
| Coinbase : [1, 1, 1, 1] |

Then the coinbase transfers 10 coins to Alice.

| Ucoins |
|---|
| Coinbase : [0, 1, 0, 1] |
| Alice : [1, 0, 1, 0] |

| Headers | | | | |
|---|---|---|---|---|
| tx | in | out | $c_0$ | $c_1$ |
| 1 | 1 | 2 | [0, 0, 0, 0, 0] | [0, 0, 0, 0, 0] |

After that Alice sends 7 coins to Bob.

| Ucoins |
|---|
| Coinbase : [0, 1, 0, 1] |
| Alice : [0, 0, 1, 1] |
| Bob : [0, 1, 1, 1] |

| Headers | | | | |
|---|---|---|---|---|
| tx | in | out | $c_0$ | $c_1$ |
| 1 | 1 | 2 | [0, 0, 0, 0, 0] | [0, 0, 0, 0, 0] |
| 2 | 1 | 2 | [0, 0, 0, 0, 0] | [0, 1, 1, 1, 0] |

From the example, we can see that spent coin bundles like Alice's 10 coins can be removed from the cash system without affecting the verifiability.

## III. AGGREGABLE CONFIDENTIAL TRANSACTIONS

We now explain how our aggregable confidential transactions work by expanding the example used in Section II.

### A. CONFIDENTIAL COIN BUNDLES

CTs store inputs and outputs as confidential coin bundles. Let $\texttt{coin}(v, k)$ be a confidential coin bundle with $v$ coins and masking key $k$.[8] Each $\texttt{coin}(v, k)$ is a tuple of a commitment and the commitment's range proof given by:

$$\texttt{coin}(v, k) := (\texttt{commit}(\mathsf{bin}(v), k), \texttt{range\_proof}(\mathsf{bin}(v), k))$$

where $\mathsf{bin}(v)$ is a vector which represents $v$ in base-2.

[8]In lattice-based cryptography, masking keys are vectors. For simplicity of the explanation, we restrict it to a single element.

A commitment makes sure that the committed value and the masking key are hiding and binding. Here, binding means no one, including the creator of the commitment, can find another value and a masking key for the same commitment. $\texttt{range\_proof}(\mathsf{bin}(v))$ ensures that $\mathsf{bin}(v)$ is a valid binary vector for a number in the range $[0, 2^L)$.

These commitments are element-wise homomorphic, e.g., $\texttt{commit}([v_{1,0}.., v_{1,L}, k_1]) + \texttt{commit}([v_{2,0}.., v_{2,L}, k_2]) = \texttt{commit}([v_{1,0} + v_{2,0}.., v_{1,L} + v_{2,L}, k_1 + k_2])$.

### B. CONFIDENTIAL CARRY PROOFS

We need the knowledge of carries to check the summation since coin bundles commit binary form of the coin amount. Therefore, CTs also contain carry proofs. Let $\texttt{carry}(\boldsymbol{c}, k)$ be a carry proof for some carry vector $\boldsymbol{c} = [c_0, .., c_L]$, given by:

$$\texttt{carry}(\boldsymbol{c}, k) := \left\{ \begin{array}{c} \texttt{carry\_commit}([c_j - 2c_{j+1}]_{j=0}^{L-1}, k), \\ \texttt{carry\_range\_proof}(\boldsymbol{c}, k) \end{array} \right\}$$

Similar to coin bundles, carry proofs have range proofs to check whether $\boldsymbol{c}$ is a binary vector or not. Additionally, carry range proofs ensure that $c_0$ and $c_L$ are zero. Note that we commit $[c_j - 2c_{j+1}]_{j=0}^{L-1}$, not $\boldsymbol{c}$, in carry proofs because our target is to perform arithmetic operations on commitments similar to Equation (1) and Equation (2).

### C. DIGITAL SIGNATURES FROM ZERO VECTOR COMMITMENTS

We depend on digital signatures to show the knowledge of secret keys without revealing them. Let $(k, \texttt{pk}(k))$ be a secret key-public key pair, where the public key is a commitment of a zero vector such that $\texttt{pk}(k) = \texttt{commit}([0, .., 0], k)$. We denote the signature of $k$ as $\texttt{sig}(k)$ where the message is empty. In our protocol, we make sure that a valid signature can be **only** created[9] if the committed value of the public key is a zero vector.

Let us explain the digital signature process at higher level. The signature scheme we explain here is a commonly used digital signature with Fiat-Shamir challenges [26]–[32] (see the complete protocol in Figure 3). The only difference is that we force all secret keys to have beginning zeros. First, the signer randomly chooses a signature mask key $r$. Then he/she creates the masking commitment $\texttt{Y} = \texttt{commit}([0, 0, 0, 0], r)$. The Fiat-Shamir challenge $x$ is created by hashing the public key, the masking commitment, and the message, i.e. $x = \mathsf{hash}(\texttt{pk}(k), \texttt{Y}, \varepsilon)$. Here, $\varepsilon$ is the empty message. Then the signature is,

$$\texttt{sig}(k) = (\sigma, x) = (r + xk, \mathsf{hash}(\texttt{pk}(k), \texttt{Y}, \varepsilon)).$$

To verify the signature, first the verifiers check whether the norm of $\sigma$ is less or equal to the hard problem's norm or not, e.g., $\|\sigma\| \leq \gamma$ for the SIS problem. Then the verifiers recreate

$$\texttt{Y}' = \texttt{commit}([0, 0, 0, 0], \sigma) - x\texttt{pk}(k)$$

[9]More accurately, it is computationally infeasible to create a valid signature for a public key with a non-zero vector due to a known hard problem like SIS or MSIS problems.

After Transaction 1:

| Ucoins | Headers | | | | | | |
|---|---|---|---|---|---|---|---|
| | tx | in | out | $c_0$ | $c_1$ | Public Key | Signature |
| Coinbase : $[0,1,0,1]$ | | | | | | | |
| coin$([1,0,1,0],k_A)$ | 1 | 1 | 2 | – | $[0,0,0,0,0]$ | pk$(k_A)$ | sig$(k_A)$ |

After Transaction 2: remove coin$([1,0,1,0],k_A)$

| Ucoins | Headers | | | | | | |
|---|---|---|---|---|---|---|---|
| | tx | in | out | $c_0$ | $c_1$ | Public Key | Signature |
| Coinbase : $[0,1,0,1]$ | | | | | | | |
| coin$([0,0,1,1],k'_A)$ | 1 | 1 | 2 | – | $[0,0,0,0,0]$ | pk$(k_A)$ | sig$(k_A)$ |
| coin$([0,1,1,1],k_B)$ | 2 | 1 | 2 | – | carry$([0,1,1,1,0],k_{c_1})$ | pk$(k'_A+k_B+k_{c_1}-k_A)$ | sig$(k'_A+k_B+k_{c_1}-k_A)$ |

**FIGURE 2.** Example of an aggregated cash system.

and check whether the hash challenge is the same or not by checking,

$$x \stackrel{?}{=} \mathsf{hash}(\mathtt{pk}(k), \mathtt{Y}', \varepsilon).$$

Therefore, a valid signature $\mathtt{sig}(k)$ can be only created if $\mathtt{pk}(k)$ is a commitment to a zero vector.

#### 1) TRANSACTION 1: ALICE RECEIVES 10 COINS FROM THE COINBASE

Recall that the coinbase sent 10 coins to Alice in the first transaction. Even though the received coin amount is public, Alice wants to make sure that her coins cannot be stolen. Therefore, she stores coins as a confidential coin bundle with a secret key. Let Alice's confidential coin bundles be $\mathtt{coin}(10, k_A)$ where $k_A$ is only known to Alice. To secure the confidential coin bundle, Alice sends a transaction $\mathtt{tx}_1$ implying the knowledge of $k_A$ via a digital signature $\mathtt{sig}(k_A)$.

$$\mathtt{tx}_1 = \begin{pmatrix} \mathtt{in} = \{[1,1,1,1]\} \\ \mathtt{out} = \{[0,1,0,1],\ \mathtt{coin}([1,0,1,0],k_A)\} \\ \mathtt{proof} = (\mathtt{pk}(k_A), \mathtt{sig}(k_A)) \end{pmatrix}$$

Here, public key $\mathtt{pk}(k_A)$ is the difference of output commitment summation and input commitment summation according to Equation (1)., i.e.,

$$\begin{aligned} [b'_{1,j}]_{j=0}^{L-1''}\mathtt{commit}([0,\ 1,\ 0,\ 1],\ 0) \\ [b'_{2,j}]_{j=0}^{L-1}\ :\mathtt{commit}([1,\ 0,\ 1,\ 0],k_A) \\ [b_{1,j}]_{j=0}^{L-1}\ :-\mathtt{commit}([1,\ 1,\ 1,\ 1],\ 0) \\ [c^{(1)}_{1,j}-2c^{(1)}_{1,j+1}]_{j=0}^{L-1}\ :\mathtt{carry\_commit}[0,\ 0,\ 0,\ 0],\ 0) \\ [c^{(1)}_{0,j}-2c^{(1)}_{0,j+1}]_{j=0}^{L-1}\ :-\mathtt{carry\_commit}[0,\ 0,\ 0,\ 0],\ 0) \\ \hline \mathtt{pk}(k_A) = \mathtt{commit}([0,\ 0,\ 0,\ 0],k_A) \end{aligned}$$

Note that if a coin value is public, a commitment with a zero key is computed when computing the public key, allowing the verifiers to recreate the same commitment easily. Also, if a carry vector is all zeros, we do not need to compute its commitment; however, we include it to show the equivalence of the public key computation and Equation (1).

Upon receiving the transaction, the verifiers can recreate the public key $\mathtt{pk}(k_A)$ since all the details required are in the transaction. After that verifiers check whether the digital signature $\mathtt{sig}(k_A)$ is correctly generated for the public key or not. This signature shows

1) the knowledge of the aggregate secret key $(k_A)$ since the public key was computed taking the commitments and
2) the transaction's zero-coin generation since the signature can be only created if its committed value of the public key is a zero vector.

Note that we do not need to include carry proofs in $\mathtt{tx}_1$ since coin values are known to the public.

Then the cash system aggregates $\mathtt{tx}_1$ to itself. The state of the aggregated cash system is shown in in Figure 2. *After* the aggregation, we consider Alice has *received* coins. Now, Alice's coins are locked with the secret masking key $k_A$.

#### 2) TRANSACTION 2: ALICE SENDS 7 COINS TO BOB

Alice sends 7 coins to Bob from her 10 coins. We denote the second transaction $\mathtt{tx}_2$ below. Unlike the first transaction, they have to include carry proofs for outputs[10] to the second transaction since carries might reveal the hidden coin amounts. With the second transaction, Alice updates the secret mask of the balance to $k'_A$, Bob receives coins under the secret mask $k_B$, and carries are hidden with $k_{c_1}$.

$$\mathtt{tx}_2 = \begin{pmatrix} \mathtt{in} = \{\mathtt{coin}([1,0,1,0],k_A)\} \\ \mathtt{out} = \begin{Bmatrix} \mathtt{coin}([0,0,1,1],k'_A) \\ \mathtt{coin}([0,1,1,1],k_B) \end{Bmatrix} \\ \mathtt{proof}: \begin{pmatrix} \mathtt{carry}([0,1,1,1,0],k_{c_0}) \\ \mathtt{pk}(k'_A+k_B+k_{c_1}-k_A) \\ \mathtt{sig}(k'_A+k_B+k_{c_1}-k_A) \end{pmatrix} \end{pmatrix}$$

After receiving $\mathtt{tx}_2$, the verifiers check these; (1) the input coin bundle is in Ucoins, (2) the range proofs are correct for all inputs, outputs, and carries, and (3) the public key

[10]They do not have to include input carry proofs since there is only one input coin bundle.

$\mathtt{pk}(k'_A + k_B + k_{c_1} - k_A)$ is equal to $\mathtt{commit}([0,0,0,0], k'_A + k_B + k_{c_1} - k_A)$ such that

$$\begin{aligned}
[b'_{1,j}]_{j=0}^{L-1} &: \mathtt{commit}([0,\ 0,\ 1,\ 1], k'_A)\\
[b'_{2,j}]_{j=0}^{L-1} &: \mathtt{commit}([0,\ 1,\ 1,\ 1], k_B)\\
[b_{1,j}]_{j=0}^{L-1} &: -\mathtt{commit}([1,\ 0,\ 1,\ 0], k_A)\\
[c^{(2)}_{1,j} - 2c^{(2)}_{1,j+1}]_{j=0}^{L-1} &: \mathtt{carry_commit}([1,-1,-1,-2], k_{c_1})\\
[c^{(2)}_{0,j} - 2c^{(2)}_{0,j+1}]_{j=0}^{L-1} &: -\mathtt{carry_commit}([0,\ 0,\ 0,\ 0],\ 0)\\
\hline
&\mathtt{commit}([0,0,0,0], k'_A + k_B + k_{c_1} - k_A)
\end{aligned}$$

At last the digital signature $\mathtt{sig}(k'_A + k_B + k_{c_1} - k_A)$ is verified. This signature prevents stealing the coins due to the need for $k_A$. Also, verifiers use the signature to check that no coins were illegally generated during the transaction according to Equation (1) since a valid signature can be created only if the committed value of the public key is a zero vector.

Once the transaction is *added* to the cash system, we consider that Alice has *sent* coin, and Bob has *received* coins.

### *a: TRANSACTION CUT-THROUGH*

If the cash system is stateful, it must preserve all coin bundles, including the spent coin bundles like $\mathtt{coin}(10, k_A)$. However, stateless cash systems allow optional deletion called "cut-though" or "aggregation" to remove spent coin bundles. Even after the aggregation, we can **fully verify** the cash system, i.e., the unspent coin bundles are not stolen, and their hidden coin amounts are equal to the supply coins. For example, we can check the cash system of Figure 2 s.t.,

$$\begin{pmatrix} \mathtt{commit}([1,1,1,1],0)\\ +\mathtt{pk}(k_A)\\ +\mathtt{pk}(k'_A + k_B + k_{c_1} - k_A) \end{pmatrix} = \begin{pmatrix} \mathtt{carry\_commit}([1,\ -1,\ -1,\ -2], k_{c_1})\\ +[0,\ \ 1,\ \ 0,\ \ 1]\\ +\mathtt{coin}([0,\ \ 0,\ \ 1,\ \ 1], k'_A)\\ +\mathtt{coin}([0,\ \ 1,\ \ 1,\ \ 1], k_B) \end{pmatrix} \quad (3)$$

following Equation (2). Here, $\mathtt{commit}([1,1,1,1],0)$ is the supply coin commitment with no keys and $[0,1,0,1]$ is the coinbase.

The cash system can remove spent coin records without affecting the verification since Equation (3) is independent from spent coin bundles, e.g., $\mathtt{coin}(10, k_A)$.

## IV. RELATED WORK

We compare our contributions with other confidential transactions in Table 2. Typical transaction protocols like [8]–[12] preserve everything, including spent coin records since deletion breaks the public verification. For example, consensus mechanisms like Proof of Work and Proof of Stake, check[11] all spent and unspent coin records to prevent altering and stealing (which is known as immutability). The first aggregable transaction protocol [22] introduced a novel idea to allow public verification even after the transaction cut-through, which was later adapted for a cash system called Mimblewimble [15]. Unlike lattice-based CTs, these CTs [15], [17], [22], [35] enjoy the efficiency of the DL problem based range proofs [39]–[41] with no bounds. Unfortunately, these efficient transaction protocols are vulnerable to quantum adversaries due to the use of the DL problem.

Quantum-safe Ring Confidential Transactions [21], [36]–[38] are more private than typical Aggregable transactions due to the mixing. For example, the real sender is mixed with a set of other participants. The ownership of the coins is shown via a ring signature where the real sender can create signatures for other participants as well. However, these ring signatures prevent the transaction aggregation techniques. Aggregable Confidential Transactions do not provide mixing capabilities, yet hide coin amounts. Therefore, aggregable CTs are both scalable and private.

MatRiCT [21] is one of the most efficient quantum-safe Ring CT protocols since it also uses binary commitments similar to our protocols.[12] The downside of binary commitment is that ZK summation proof requires carry commitments if there is more than one input/output. Also, this limits the maximum number of inputs and outputs to 2, and other complex transactions should be divided into multiple transactions. However, from statistics [23], real-world transactions have average 1.5 outputs/input rate. Therefore, even dividing complex into multiple transactions is more efficient than having larger modulus like $2^{196}$ in [37] (see more details in [21]).

Our confidential transactions also suffer from this limitation on the number of inputs and outputs. However, similar to [21], having binary commitments is more efficient than having large modulus like [25]. Moreover, our protocols are aggregable due to the static carries, unlike MatRiCT with volatile carries, which do not remove additional transactions' inputs. For example, assume a transaction of $2 \Rightarrow 3$. Then we have to create two transactions of $2 \Rightarrow 2$ and $1 \Rightarrow 2$. Since [21] is stateful, they have to add 4 additional coin bundles (totally 6 coin bundles) while cash systems, based on our protocols, delete 2 coin bundles and only add 3 coin bundles. Therefore, our confidential transactions are more efficient and scalable than other quantum-safe confidential transactions due to aggregation and binary commitments.

## V. PRELIMINARIES

We denote the ring of integers modulo $q$ by $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, which is represented by the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$ for an odd prime $q$. For an even $q$, the range is $(-\frac{q}{2}, \frac{q}{2}]$. $\mathcal{R}$ is a polynomial ring of $\mathbb{Z}[X]/[X^N+1]$ with degree $N = 2^k$ for some integer $k > 0$. $\mathcal{R}_q$ is a fully splitting ring of $\mathbb{Z}_q[X]/[X^N+1]$ over $q$ such that $q \equiv 1 \pmod{2N}$. A polynomial $a_{N-1}X^{N-1} + ..+a_1X+a_0 \in \mathcal{R}_q$ is denoted as $\vec{a} = [a_0, ..a_{N-1}]$ when each

[11] Each transaction is hashed and included in the Markle hash tree of a block. The entire transaction must be given to check the Merkle tree. Therefore, deletion is not an option for those cash systems.

[12] The major differences are that they are not aggregable like our protocols and do not embed the coinbase to the cash system at the beginning.

**TABLE 2. Comparison of related work. Note that the root Hermite factor should be less than 1.0045 to achieve 128-bit security against the known attacks [33]. Here, "unlimited" means unlimited coin bundles, and "C. Size" is the size of a commitment in a coin bundle.**

| Work | Type | Hardness | Quantum-Safe | Unlimited | Root Hermite Factor ($\delta$) | Modulus ($q$) | Dimensions | C. Size |
|---|---|---|---|---|---|---|---|---|
| [10] | Ring | DL | ✗ | ✓ | - | - | - | - |
| [20] | Ring | DL | ✗ | ✓ | - | - | - | - |
| [33] | Ring | DL | ✗ | ✓ | - | - | - | - |
| [22] | Aggregate | DL | ✗ | ✓ | - | - | - | - |
| [15] | Aggregate | DL | ✗ | ✓ | - | - | - | - |
| [17] | Aggregate | DL | ✗ | ✓ | - | - | - | - |
| [34] | Aggregate | DL | ✗ | ✓ | - | - | - | - |
| [35] | Ring | Knapsack | ✓ | - | - | - | - | - |
| [36] | Ring | Ring-SIS | ✓ | - | 1.007 (✗) | $\approx 2^{61}$ | $(256, 256 \times 10)$ | 2 KB |
| [37] | Ring | MSIS | ✓ | - | 1.007 (✗) | $\approx 2^{196}$ | $(1024, 132)$ | 25 KB |
| [21] | Ring | MSIS/MLWE | ✓ | ✓ | 1.0045 (✓) | $\approx 2^{53}$ | $(64 \times 32, 64 \times 65)$ | 13.5 KB |
| [25] | Aggregate | SIS | ✓ | ✗ | - | $\gg 2^{64}$ | - | - |
| Ours | Aggregate | SIS | ✓ | ✓ | - | $\gg 2$ | - | - |
| Ours | Aggregate | MSIS | ✓ | ✓ | 1.004 (✓✓) | $\approx 2^{50}$ | $(256 \times 6, 256 \times 5)$ | 9.6 KB |

coefficient is in $\mathbb{Z}_q$. We will often write $\vec{a}$ for both the vector and the corresponding polynomial.

We use simple bold letters like $\boldsymbol{a}$ for integer vectors and capital bold letters such as $\boldsymbol{A}$ for integer matrices. Similarly, $\vec{\boldsymbol{a}}$ denotes a vector of polynomials, and $\vec{\boldsymbol{A}}$ denotes a matrix of polynomials. The $i$th element of a vector $\boldsymbol{a}$ is $a_i$ or we can denote the entire vector as $\boldsymbol{a} = [a_0, .., a_{n-1}] = [a_i]_{i=0}^{n-1}$ when $\boldsymbol{a}$ has $n$ elements. Also, $\boldsymbol{A} = [\boldsymbol{A}_0, .., \boldsymbol{A}_{m-1}]$ when $\boldsymbol{A}$ has $m$ vectors. We denote matrix multiplication as $\boldsymbol{Aa}$ when $\boldsymbol{A}$'s numbers of columns matches with $\boldsymbol{a}$'s number of elements. $\vec{a}\vec{b}$ denotes the polynomial multiplication of $\vec{a}$ and $\vec{b}$. Also, $\vec{a}\vec{a} = \vec{a}^2$. When $x$ is an integer, $x\boldsymbol{a}$ denotes that every element in $\boldsymbol{a}$ is multiplied by $x$. $\vec{x}\vec{\boldsymbol{a}}$ denotes that every polynomial in $\vec{\boldsymbol{a}}$ is multiplied by $\vec{x}$. We use "+", "−", and "∘" for element-wise or polynomial-wise addition, subtraction, and multiplication respectively, i.e. $\boldsymbol{a} + \boldsymbol{b} = [a_0 + b_0, .., a_{n-1} + b_{n-1}]$, $\boldsymbol{a} \circ \boldsymbol{b} = [a_0 b_0, .., a_{n-1} b_{n-1}]$, $\vec{\boldsymbol{a}} + \vec{\boldsymbol{b}} = [\vec{a}_0 + \vec{b}_0, .., \vec{a}_{n-1} + \vec{b}_{n-1}]$, and $\vec{\boldsymbol{a}} \circ \vec{\boldsymbol{b}} = [\vec{a}_0 \vec{b}_0, .., \vec{a}_{n-1} \vec{b}_{n-1}]$. Sometimes, we use $\boldsymbol{a}^2$ and $\vec{\boldsymbol{a}}^2$ for $\boldsymbol{a} \circ \boldsymbol{a}$ and $\vec{\boldsymbol{a}} \circ \vec{\boldsymbol{a}}$, respectively.

The norms are defined as follows, $\|\boldsymbol{a}\| = \mathsf{max}([|a_i|]_{i=0}^{n-1})$, and $\|\boldsymbol{a}\|_1 = \sum_{i=0}^{n-1} |a_i|$. The norms for a polynomial $\vec{a}$ are defined as follows, $\|\vec{a}\| = \mathsf{max}([|a_i|]_{i=0}^{N-1})$ and $\|\vec{a}\|_1 = \sum_{i=0}^{N-1} |a_i|$. For a vector of polynomials $\vec{\boldsymbol{a}}$, $\|\vec{\boldsymbol{a}}\| = \mathsf{max}([\|\vec{a}_i\|]_{i=0}^{n-1})$. Note that $\mathbf{0}^n \in \mathbb{Z}_q^n$ is a zero vector and $\vec{0} \in \mathcal{R}_q$ is the polynomial with all zero coefficients. $\mathsf{bin}(v)$ outputs $\boldsymbol{b} \in \mathbb{Z}_q$ or $\vec{b} \in \mathcal{R}_q$ such that $v = \sum_{i=0}^{N-1} 2^i b_i$.

We use $a \leftarrow S$ to denote $a$ was sampled from $S$ and $a \xleftarrow{\$} S$ to denote $a$ was uniformly sampled from $S$. $a_0, .., a_{m-1} \leftarrow S^m$ denotes each $a_i$ is sampled from $S$. $\epsilon(\lambda) = 1/o(\lambda^c)$ is a negligible function which vanishes faster than any polynomial of degree $c$, $\forall c \in \mathbb{N}$. We generally use $pp_{(\lambda)}$ to denote public parameters of some protocol with $\lambda$ bits of security.

We use $\mathcal{A}$ to denote an adversarial algorithm. $\mathcal{A}$ is a stateful interactive algorithm, e.g., $\mathcal{A}_{step1}()$ and $\mathcal{A}_{step2}()$ denote the first step and the second step of the protocol, respectively. Since $\mathcal{A}$ is stateful, $\mathcal{A}$ can save information from the first step for the second step.

**Definition 1 (Statistical Distance):** Let $X$ and $Y$ be two random variables with range $U$. The statistical distance between $X$ and $Y$ is defined as, $\Delta(X, Y) = \frac{1}{2} \sum_{u \in U} \left| Pr[X = u] - Pr[Y = u] \right|$. For any $\theta > 0$, we say $X$ and $Y$ are statistically $\theta$-close if $\Delta(X, Y) \leq \theta$.

**Theorem 1 ([42]):** The distribution $(\boldsymbol{H}, \boldsymbol{Hr} \mod q)$ is statistically $\epsilon(\lambda)$-close to the distribution of $(\boldsymbol{H}, \boldsymbol{u})$ where $\boldsymbol{H} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\boldsymbol{r} \xleftarrow{\$} [-\tau, \tau]^m$, and $\boldsymbol{u} \xleftarrow{\$} [-q, q]^n$ when $m \log(2\tau) > n \log q + 2 \log(1/\epsilon(\lambda))$ or $(2\tau)^m / q^n > 2/\epsilon(\lambda)$.

**Definition 2 (Short Integer Solution Problem (SIS)):** The *advantage* of an algorithm $\mathcal{A}$ solving a special instance of $\mathtt{SIS}_{n,m,q,\gamma}$ after one execution is given by

$$\mathsf{Adv}_{pp_\lambda}^{\mathtt{SIS},\mathcal{A}} := Pr\left[\mathsf{Game}_{pp_\lambda}^{\mathtt{SIS},\mathcal{A}}() \mid pp_\lambda = (n, m, q, \gamma)\right].$$

**Game 1:** SIS Challenge

$\mathsf{Game}_{pp_\lambda}^{\mathtt{SIS},\mathcal{A}}()$:

$\boldsymbol{H} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ // Choose a random matrix
$(\boldsymbol{s} \in \mathbb{Z}^m) \leftarrow \mathcal{A}(pp_\lambda, \boldsymbol{H})$ // Get the short vector from $\mathcal{A}$
return $0 < \|\boldsymbol{s}\| \leq \gamma \bigwedge \boldsymbol{Hs} = \mathbf{0}^n \in \mathbb{Z}_q^n$

**Definition 3 (Module Short Integer Solution Problem (MSIS)):** *The advantage of an algorithm $\mathcal{A}$ solving an special instance of $\mathtt{MSIS}_{n,m,q,\gamma,N}$ after one execution is given by,*

$$\mathsf{Adv}_{pp_\lambda}^{\mathtt{MSIS},\mathcal{A}} := Pr\left[\mathsf{Game}_{pp_\lambda}^{\mathtt{MSIS},\mathcal{A}}() \mid pp_\lambda = (n, m, q, \gamma, N)\right].$$

**Game 2:** MSIS Challenge

$\mathsf{Game}_{pp_\lambda}^{\mathtt{MSIS},\mathcal{A}}()$:

$\vec{\boldsymbol{H}} \xleftarrow{\$} \mathcal{R}_q^{n \times m}$ // Choose a random matrix
$(\vec{\boldsymbol{s}} \in \mathcal{R}^m) \leftarrow \mathcal{A}(pp_\lambda, \vec{\boldsymbol{H}})$ // Get the short vector from $\mathcal{A}$
return $0 < \|\vec{\boldsymbol{s}}\| \leq \gamma \bigwedge \vec{\boldsymbol{H}}\vec{\boldsymbol{s}} = \vec{\mathbf{0}}^n \in \mathcal{R}_q^n$

### A. COMMITMENTS OF SIS/MSIS PROBLEMS

The purpose of a commitment is to secretly publish some string and open it later to show it. Therefore, a commitment is hiding and binding, i.e., finding the committed string or

finding a different value for the same commitment is difficult. We define a simple commitment scheme COM below.

**Functionality 1**: Commitments

- $\text{COM}.\textbf{gen}(pp_{(\lambda,d)}, in, r)$: return $f(in; r)$ for some hiding and binding function $f$.
- $\text{COM}.\textbf{open}(pp_{(\lambda,d)}, out, in, r)$ : return $f(in; r) \stackrel{?}{=} out$

We define the security properties of a generic COM below,

**Definition 4:** COM is hiding and binding if

$$\mathsf{Adv}^{\mathsf{BND},\mathcal{A}}_{\text{COM},pp_{(\lambda,d)}} := Pr\left[\mathsf{Game}^{\mathsf{BND},\mathcal{A}}_{\text{COM},pp_{(\lambda,d)}}()\right] \leq \epsilon(\lambda)$$

$$\mathsf{Adv}^{\mathsf{HID},\mathcal{A}}_{\text{COM},pp_{(\lambda,d)}} := 2\left|Pr\left[\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\text{COM},pp_{(\lambda,d)}}()\right] - \frac{1}{2}\right| \leq \epsilon(\lambda)$$

**Game 3:** Binding

$\mathsf{Game}^{\mathsf{BND},\mathcal{A}}_{\text{COM},pp_{(\lambda,d)}}()$:
$(out, in, r, in', r') \leftarrow \mathcal{A}(pp_{(\lambda,d)})$
return $(in, r) \stackrel{?}{\neq} (in', r') \wedge \text{COM}.\textbf{open}(pp_{(\lambda,d)}, out, in, r) \wedge \text{COM}.\textbf{open}(pp_{(\lambda,d)}, out, in', r')$

**Game 4:** Hiding

$\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\text{COM},pp_{(\lambda,d)}}()$:
$(in_0, in_1) \leftarrow \mathcal{A}_{step1}(pp_{(\lambda,d)})$
$i \stackrel{\$}{\leftarrow} [0, 1]$; $r \stackrel{\$}{\leftarrow} \mathcal{V}_r$ // $\mathcal{V}_r$ is the space of masks
$out = \text{COM}.\textbf{gen}(pp_{(\lambda,d)}, in_i, r)$
$j \leftarrow \mathcal{A}_{step2}(pp_{(\lambda,d)}, out)$
return $i \stackrel{?}{=} j$

We state two commitment protocols; $\text{COM}_{\texttt{SIS}}$ and $\text{COM}_{\texttt{MSIS}}$ below.

**Protocol 1:** SIS Commitments

$\text{COM}_{\texttt{SIS}}.\textbf{init}(\lambda, n, m, q > 2^{4\lambda/n}, \gamma, d < m)$:
Set $\tau \leq \gamma$ such that $(m-d)\log(2\tau) > 6\lambda$.
Get $\boldsymbol{H} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$.
return $pp_{(\lambda,d)} = (n, m, q, \gamma, \tau, d, \boldsymbol{H})$
$\text{COM}_{\texttt{SIS}}.\textbf{gen}(pp_{(\lambda,d)}, \boldsymbol{v} \in [-\gamma, \gamma]^d, \boldsymbol{r} \stackrel{\$}{\leftarrow} [-\tau, \tau]^{m-d})$:
return $\boldsymbol{H}[\boldsymbol{v}, \boldsymbol{r}] \in \mathbb{Z}_q^n$
$\text{COM}_{\texttt{SIS}}.\textbf{open}(pp_{(\lambda,d)}, \boldsymbol{u} \in \mathbb{Z}_q^n, \boldsymbol{v} \in \mathbb{Z}^d, \boldsymbol{r} \in \mathbb{Z}^{m-d})$ :
return $\|\boldsymbol{v}\| \leq \gamma \bigwedge \|\boldsymbol{r}\| \leq \tau \bigwedge \boldsymbol{H}[\boldsymbol{v}, \boldsymbol{r}] \stackrel{?}{=} \boldsymbol{u} \in \mathbb{Z}_q^n$

**Theorem 2:** $\text{COM}_{\texttt{SIS}}$ *is computationally hiding and computationally binding if solving* $\texttt{SIS}_{n,m,q,\gamma}$ *is at most* $\epsilon(\lambda)$ *after one execution.*

*Proof:* Breaking the binding property of $\text{COM}_{\texttt{SIS}}$ directly solves the SIS problem. However, the computational hiding property is not directly visible. Our target is to show that there are at least $2^{2\lambda}$ different $\boldsymbol{u}$ for the same $\boldsymbol{v}$ over $\mathbb{Z}_q^n$.

**Protocol 2:** MSIS Commitments

$\text{COM}_{\texttt{MSIS}}.\textbf{init}(\lambda, n, m, q > 2^{\frac{4\lambda}{nN}}, \gamma, N, d)$:
Set $\tau \leq \gamma$ such that $(m-d)N\log(2\tau) > 6\lambda$.
Get $\vec{\boldsymbol{H}} \stackrel{\$}{\leftarrow} \mathcal{R}_q^{n\times m}$.
return $pp_{(\lambda,d)} = (n, m, q, \gamma, N, \tau, d, \vec{\boldsymbol{H}})$
$\text{COM}_{\texttt{MSIS}}.\textbf{gen}(pp_{(\lambda,d)}, \vec{\boldsymbol{v}} \in [-\gamma, \gamma]^{d\times N}, \vec{\boldsymbol{r}} \stackrel{\$}{\leftarrow} [-\tau, \tau]^{(m-d)\times N})$: return $\vec{\boldsymbol{H}}[\vec{\boldsymbol{v}}, \vec{\boldsymbol{r}}] \in \mathcal{R}_q^n$
$\text{COM}_{\texttt{MSIS}}.\textbf{open}(pp_{(\lambda,d)}, \vec{\boldsymbol{u}} \in \mathcal{R}_q^n, \vec{\boldsymbol{v}} \in \mathcal{R}^d, \vec{\boldsymbol{r}} \in \mathcal{R}^{m-d})$ :
returns $\|\vec{\boldsymbol{v}}\| \leq \gamma \bigwedge \|\vec{\boldsymbol{r}}\| \leq \tau \bigwedge \vec{\boldsymbol{H}}[\vec{\boldsymbol{v}}, \vec{\boldsymbol{r}}] \stackrel{?}{=} \vec{\boldsymbol{u}} \in \mathcal{R}_q^n$

From Bertrand's postulate, we know there should be at least one prime $p$ such that $2 \leq 2^{2\lambda/n} < p < 2^{4\lambda/n} < q$. Let there be $\boldsymbol{H} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$, positive integer $\tau'$, and a function $f'$ such that

$$f'(\boldsymbol{v} \in [-\gamma, \gamma]^d; \boldsymbol{r} \in [-\tau', \tau']^{m-d}) : \text{return } \boldsymbol{u} = \boldsymbol{H}[\boldsymbol{v}, \boldsymbol{r}] \in \mathbb{Z}^n_{(p>2^{2\lambda/n})}$$

Recall from Theorem 1, this function $f'$ outputs an uniform distribution over $\mathbb{Z}_p^n$ for any $\boldsymbol{v}$ when $\tau'$ satisfies the following:

$$(m-d)\log(2\tau') - 2\lambda \geq n\log(p). \tag{4}$$

In other words, if $\tau'$ satisfies Equation (4), there are at least $2^{2\lambda}$ different outputs for the same $\boldsymbol{v}$ over $\mathbb{Z}_p^n$ with an overwhelming probability since $p > 2^{2\lambda/n}$.

There are at least $2^{2\lambda}$ different $\boldsymbol{u}$ over $\mathbb{Z}_q^n$ as well since $q > p$, and the map of $\mathbb{Z}_p \mapsto \mathbb{Z}_q$ is injective for prime numbers $p$ and $q$. According to the **init** function of $\text{COM}_{\texttt{SIS}}$, $\tau \geq \tau'$ always satisfies Equation (5) s.t.,

$$(m-d)\log(2\tau) \geq 6\lambda \Rightarrow (m-d)\log(2\tau) - 2\lambda \geq 4\lambda$$
$$(m-d)\log(2\tau) - 2\lambda \geq n\log(p) \tag{5}$$

since $p < 2^{4\lambda/n}$. Therefore, $\mathsf{Adv}^{\mathsf{HID},*}_{\text{COM},pp_{(\lambda,d)}} \leq \epsilon(\lambda)$ after one execution. Hence Theorem 2 is true, i.e. $\text{COM}_{\texttt{SIS}}$ is computationally hiding and computationally binding. □

**Theorem 3:** $\text{COM}_{\texttt{MSIS}}$ *is computationally hiding and binding if solving* $\texttt{MSIS}_{n,m,q,\gamma,N}$ *is at most* $\epsilon(\lambda)$ *after one execution.*

*Proof:* Any valid output of $\mathsf{Game}^{\mathsf{BND},*}_{\text{COM},pp_{(\lambda,d)}}$ is a solution to MSIS problem. Therefore, we move to the computational hiding property. Note that, $q < 2^{\frac{4\lambda}{nN}}$.

Therefore, there should be at least one prime $p$ such that $2 \leq 2^{2\lambda/nN} < p < 2^{4\lambda/nN} < q$. Let $\vec{\boldsymbol{H}} \stackrel{\$}{\leftarrow} \mathcal{R}_q^{n\times m}$ and a function $f'$,

$$f'(\vec{\boldsymbol{v}} \in [-\gamma, \gamma]^{d\times N}; \vec{\boldsymbol{r}} \in [-\tau', \tau']^{(m-d)\times N}): \text{return } \vec{\boldsymbol{u}} = \vec{\boldsymbol{H}}[\vec{\boldsymbol{v}}; \vec{\boldsymbol{r}}] \in \mathcal{R}_p^n$$

that outputs a uniform distribution over $\mathcal{R}_p^n$. Provided that,

$$(m-d)N\log(2\tau') - 2\lambda \geq nN\log(p) \tag{6}$$

according to Theorem 1. Since $p > 2^{2\lambda/nN}$ and $p < q$, there will be at least $2^{2\lambda}$ different polynomials over $\mathcal{R}_q^n$.

According to the initialization function of $\mathsf{COM}_{\mathtt{MSIS}}$, $\tau \geq \tau'$ always satisfies Equation (6) because,

$$(m-d)N\log(2\tau) > 6\lambda \Rightarrow (m-d)N\log(2\tau) - 2\lambda > 4\lambda$$
$$(m-d)N\log(2\tau) - 2\lambda > nN\log(p),$$

when $p < 2^{4\lambda/nN}$. Therefore, $\mathsf{Adv}^{\mathsf{HID},*}_{\mathrm{COM},pp_{(\lambda,d)}}$ is $\epsilon(\lambda)$ after one execution and $\mathrm{COM}_{\mathtt{MSIS}}$ is computationally hiding. □

### B. SIGNATURES WITH FIAT SHAMIR CHALLENGES

Next, we define digital signature schemes based on [26]–[32]. In this paper, we employ many-time signatures that are EUF-CMA (Existential Unforgeable under Chosen Message Attack) due to rejection sampling even though only one signature is created from the same key pair.

**Functionality 2**: Digital Signatures

- SIG.**kgen**($pp_\lambda$): return $(sk, pk)$ // Here, $sk$ is the secret key, and $pk$ is the public key.
- SIG.**sign**($pp_\lambda, sk, m$): return $(\sigma, x)$. // The tuple $(\sigma, x)$ is the the signature
- SIG.**ver**($pp_\lambda, pk, m, \sigma, x$) : return 1 if the signature $(\sigma, x)$ is valid, otherwise 0.

We define the completeness and strong EUF-CMA of SIG below.

**Definition 5:** SIG is complete and strong EUF-CMA if

$$Pr\left[\begin{array}{c}\mathsf{ver}(pp_\lambda,\\ pk,m,\sigma,x)\end{array}\middle|\begin{array}{c}(sk,pk)=\mathsf{kgen}(pp_\lambda)\\ (\sigma,x)=\mathsf{sign}(pp_\lambda,sk,m)\end{array}\right] \geq 1-\epsilon(\lambda)$$
$$\mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG},pp_\lambda} := Pr\left[\mathsf{Game}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG},pp_\lambda}()\right] \leq \epsilon(\lambda)$$

**Game 5:** Unforgeability

$\mathbf{Game}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG},pp_\lambda}()$:
$(sk, pk) = \mathrm{SIG}.\mathbf{kgen}(pp_\lambda)$
$(m', (\sigma', x')) \leftarrow \mathcal{A}^{\mathtt{sign}_{sk}}(pp_\lambda, pk)$
return $(m', (\sigma', x')) \notin \mathbf{Q} \wedge \mathrm{SIG}.\mathbf{ver}(pp_\lambda, pk, m', \sigma', x')$

**Oracle** $\mathtt{sign}_{sk}(m)$:
$(\sigma, x) = \mathrm{SIG}.\mathbf{sign}(pp_\lambda, sk, m)$
$\mathbf{Q} = \mathbf{Q} \cup \{(m, (\sigma, x))\}$, *return* $(\sigma, x)$

We state variations of [31], [32] in Figure 3. In these protocols, to reduce the size, the signature contains a hash challenge similar to [32] instead of the masking commitment ($\boldsymbol{Y}$ or $\vec{\boldsymbol{y}}$). $\vec{\boldsymbol{y}}$ is around 9 KB when the hash challenge is 32 bytes in our MSIS-CT protocol.

**Theorem 4:** $\mathrm{SIG}_{\mathtt{SIS}}$ *is complete and EUF-CMA if* SIS *problem is hard for* $(n, m, q, \gamma)$. *(see details in [32])*

**Theorem 5:** $\mathrm{SIG}_{\mathtt{MSIS}}$ *is complete and EUF-CMA if* MSIS *problem is hard for* $(n, m, q, \gamma, N)$. *(see details in [31])*

**Protocol 3:** SIS Signatures

$\mathrm{SIG}_{\mathtt{SIS}}.\mathsf{init}(n, m, q, \gamma)$ :
Set $\mathtt{ch}, \chi, \tau, \tau'$ such that
  $\mathtt{ch} \geq \log_{2\chi} 2^{2\lambda}$ and $\chi\tau \ll \tau' \leq \gamma$
Get $\boldsymbol{H} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$
return $pp_\lambda = (n, m, q, \gamma, \mathtt{ch}, \chi, \tau, \tau', \boldsymbol{H})$

$\mathrm{SIG}_{\mathtt{SIS}}.\mathsf{kgen}(pp_\lambda)$ : // $\boldsymbol{r}$ is the secret signing key
return $(\boldsymbol{r} \xleftarrow{\$} [-\tau, \tau]^m, \boldsymbol{pk} = \boldsymbol{H}\boldsymbol{r} \in \mathbb{Z}_q^n)$

$\mathrm{SIG}_{\mathtt{SIS}}.\mathsf{sign}(pp_\lambda, \boldsymbol{r}, m)$ :
$\boldsymbol{R} \xleftarrow{\$} [-\tau', \tau']^{\mathtt{ch}\times m}$
$\boldsymbol{Y} = [\boldsymbol{H}\boldsymbol{R}_i]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
$\boldsymbol{x} = \mathsf{hash}(\boldsymbol{H}\boldsymbol{r}, \boldsymbol{Y}, m) \in [\chi, \chi]^{\mathtt{ch}}$
**for** $i \in [0, \mathtt{ch})$ : $\boldsymbol{\sigma}_i = \boldsymbol{R}_i + x_i\boldsymbol{r} \in \mathbb{Z}^m$
  **if** $\|\boldsymbol{\sigma}_i\| > (\tau' - \chi\tau)$: go to Step 10
return $(\boldsymbol{\sigma}, \boldsymbol{x})$

$\mathrm{SIG}_{\mathtt{SIS}}.\mathsf{ver}(pp_\lambda, \boldsymbol{pk}, m, \boldsymbol{\sigma}, \boldsymbol{x})$ :
$\boldsymbol{Y} = [\boldsymbol{H}\boldsymbol{\sigma}_i - x_i\boldsymbol{pk}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
return $\|\boldsymbol{\sigma}\| < \gamma \bigwedge \boldsymbol{x} \stackrel{?}{=} \mathsf{hash}(\boldsymbol{pk}, \boldsymbol{Y}, m)$

**Protocol 4:** MSIS Signatures

$\mathrm{SIG}_{\mathtt{MSIS}}.\mathsf{init}(n, m, q, \gamma, N)$ :
Set $\beta, \tau, \tau'$ such that
  $\beta + \log\binom{N}{\beta} \geq 2\lambda$ and $\beta\tau \ll \tau' \leq \gamma$
Get $\vec{\boldsymbol{H}} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$
$\mathcal{C}^N_{\beta,1} = [x \in \mathcal{R}_q \text{ s.t. } \|x\| = 1, \|x\|_1 = \beta]$
return $pp_\lambda = (n, m, q, \gamma, \mathcal{C}^N_{\beta,1}, \tau, \tau', \vec{\boldsymbol{H}})$

$\mathrm{SIG}_{\mathtt{MSIS}}.\mathsf{kgen}(pp_\lambda)$ : // $\vec{\boldsymbol{r}}$ is the secret signing key
return $(\vec{\boldsymbol{r}} \xleftarrow{\$} [-\tau, \tau]^{m\times N}, \vec{\boldsymbol{pk}} = \vec{\boldsymbol{H}}\vec{\boldsymbol{r}} \in \mathcal{R}_q^n)$

$\mathrm{SIG}_{\mathtt{MSIS}}.\mathsf{sign}(pp_\lambda, \vec{\boldsymbol{r}}, m)$ :
$\vec{\boldsymbol{r}}_0 \xleftarrow{\$} [-\tau', \tau']^{m\times N}$
$\vec{\boldsymbol{y}} = \vec{\boldsymbol{H}}\vec{\boldsymbol{r}}_0 \in \mathcal{R}_q^n$
$\vec{x} = \mathsf{hash}(\vec{\boldsymbol{H}}\vec{\boldsymbol{r}}, \vec{\boldsymbol{y}}, m) \in \mathcal{C}^N_{\beta,1}$
$\vec{\boldsymbol{\sigma}} = \vec{\boldsymbol{r}}_0 + \vec{x}\vec{\boldsymbol{r}} \in \mathcal{R}^m$
**if** $\|\vec{\boldsymbol{\sigma}}\| > (\tau' - \beta\tau)$: go to Step 11
return $(\vec{\boldsymbol{\sigma}}, \vec{x})$

$\mathrm{SIG}_{\mathtt{MSIS}}.\mathsf{ver}(pp_\lambda, \vec{\boldsymbol{pk}}, m, \vec{\boldsymbol{\sigma}}, \vec{x})$ :
$\vec{\boldsymbol{y}} = \vec{\boldsymbol{H}}\vec{\boldsymbol{\sigma}} - \vec{x}\vec{\boldsymbol{pk}} \in \mathcal{R}_q^n$
return $\|\vec{\boldsymbol{\sigma}}\| < \gamma \bigwedge \vec{x} \stackrel{?}{=} \mathsf{hash}(\vec{\boldsymbol{pk}}, \vec{\boldsymbol{y}}, m)$

**FIGURE 3.** Digital signatures.

## VI. PROPERTIES OF CONFIDENTIAL COIN BUNDLES

Confidential coin bundles are special commitments accompanied by range proofs of the committed coin values. As discussed before, commitments contain mask keys, and we

use these masks as the ownership of the coin bundles. For example, sending a coin bundle is equivalent to sending the masking key of the coin bundle to the receiver.

Let there be a confidential coin bundle scheme COIN.

**Functionality 3**: Confidential Coin Bundles

- COIN.**gen**$(pp_{(\lambda,L)}, v)$: returns $(\texttt{mask}, \texttt{coin})$ if $v$ is in $[0, 2^L - 1]$; otherwise returns $\perp$. Here, mask is a secret mask(s), and $L$ is defined in $pp_{(\lambda,L)}$.
- COIN.**open**$(pp_{(\lambda,L)}, v, \texttt{mask}, \texttt{coin})$: returns 1 if coin is generated for valid $(v \in [0, 2^L), \texttt{mask})$; otherwise, returns 0.
- COIN.**ver**$(pp_{(\lambda,L)}, \texttt{coin})$: returns 1 if the coins bundle is in $[0, 2^L - 1]$ otherwise, returns 0. Note that this public verification *does not* need the secret mask(s).

We define the expected security properties of COIN below. COIN is complete if honestly generated coin bundles can be opened and publicly verified.

**Definition 6 (Completeness):** COIN is complete if,

$$Pr\left[\begin{array}{c|c} \text{COIN.}\mathsf{open}(pp_{(\lambda,L)}, & v \in [0, 2^L - 1] \\ v, \texttt{mask}, \texttt{coin}) & (\texttt{mask}, \texttt{coin}) = \\ \wedge\text{COIN.}\mathsf{ver}(pp_{(\lambda,L)}, & \text{COIN.}\mathsf{gen}(pp_{(\lambda,L)}, v) \\ \texttt{coin}) & \end{array}\right]$$

is greater than or equal to $1 - \epsilon(\lambda)$.

Once the coin bundle is generated, the coin value and mask should be binding, or no one (including the creator) should be able to come up with a different coin value and mask for the same coin bundle.

**Definition 7 (Binding):** COIN is binding if

$$\mathsf{Adv}^{\mathsf{BND},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}} := Pr\left[\mathsf{Game}^{\mathsf{BND},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}()\right] \leq \epsilon(\lambda).$$

**Game 6:** Binding

$\mathsf{Game}^{\mathsf{BND},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}()$:

$(\texttt{coin}, v, \texttt{mask}, v', \texttt{mask}') \leftarrow \mathcal{A}(pp_{(\lambda,L)})$

return $(v, \texttt{mask}) \overset{?}{\neq} (v', \texttt{mask}')$
$\bigwedge$ COIN.**open**$(pp_{(\lambda,L)}, v, \texttt{mask}, \texttt{coin})$
$\bigwedge$ COIN.**open**$(pp_{(\lambda,L)}, v', \texttt{mask}', \texttt{coin})$

As the name implies, coin bundles should be confidential, i.e., no algorithm should be able to distinguish the coin amount of the coin bundle. We capture this hiding property as follows.

**Definition 8 (Hiding):** COIN is hiding if $\mathsf{Adv}^{\mathsf{HID},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}$ is

$$2\left|Pr\left[\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}()\right] - 1/2\right| \leq \epsilon(\lambda).$$

As we discussed before, confidential coin bundles contain range proofs for the committed values. Knowledge soundness ensures that no p.p.t. algorithm can compute a valid coin bundle for a coin amount that is not in $[0, 2^L - 1]$. Let $\mathcal{E}$ be an extractor that extracts the committed value of a coin bundle.

**Game 7:** Hiding

$\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}()$:

$(v_0, v_1) \leftarrow \mathcal{A}_{step1}(pp_{(\lambda,L)})$

$i \xleftarrow{\$} [0, 1]$; $(\texttt{mask}, \texttt{coin}) = \text{COIN.}\mathsf{gen}(pp_{(\lambda,L)}, v_i)$

$j \leftarrow \mathcal{A}_{step2}(pp_{(\lambda,L)}, v_0, v_1, \texttt{coin})$

return $i \overset{?}{=} j \bigwedge (v_0, v_1) \overset{?}{\in} [0, 2^L - 1]$

We define knowledge soundness such that no p.p.t. algorithm can compute a valid coin bundle for an invalid coin value, i.e., the extracted coin value of $\mathcal{E}$ is not in $[0, 2^L - 1]$.

**Definition 9 (Knowledge Soundness):** COIN is knowledge sound if

$$\mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}} = Pr\left[\mathsf{Game}^{\mathsf{KS},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}()\right] \leq \epsilon(\lambda).$$

Note that extractors do not exist in real-world unless the public parameters have been tampered with trapdoor functions. These extractors are commonly used in zero-knowledge proofs or arguments to verify security. In this paper, we construct extractors based on the Generalized Forking Lemma [27] where the adversarial algorithm is rewound just before the challenge computation.

**Game 8:** Knowledge Soundess

$\mathsf{Game}^{\mathsf{KS},\mathcal{A}}_{\text{COIN},pp_{(\lambda,L)}}()$:

$\texttt{coin} \leftarrow \mathcal{A}(pp_{(\lambda,L)})$

return $\mathcal{E}(\texttt{coin}) \overset{?}{\notin} [0, 2^L - 1] \bigwedge \text{COIN.}\mathsf{ver}(pp_{(\lambda,L)}, \texttt{coin})$

## VII. PROPERTIES OF CONFIDENTIAL TRANSACTIONS

Confidential transactions contain input coin bundles, output coin bundles, and proofs to verify the ownership and zero-coin generation. Here, zero-coin generation ensures that input coin summation is equal to output coin summation. The ownership proofs (typically a multi-signature) are created by taking the inputs' and outputs' masking keys. Therefore, coin bundles cannot be stolen without the masking keys. Recall the ownership of a coin bundle is its masking key. Therefore, once a recipient gets a masking key, he/she sends a transaction updating the masking key of the coin bundle. Once the transaction is appended to the cash system, only the recipient knows the coin bundle's new masking key, and no one can steal coins, including the previous owner.

Let CTx be a confidential transaction scheme for transactions with $|\texttt{in}|$ inputs and $|\texttt{out}|$ outputs.

We say CTx is complete if honestly generated confidential transactions can be publicly verified.

**Definition 10 (Completeness):** CTx is complete if,

$$Pr\left[\mathsf{random\_tx}(pp_{(\lambda,L)}, |\texttt{in}|, |\texttt{out}|)\right] \geq 1 - \epsilon(\lambda).$$

Confidential transactions should be hiding, or all input coin amounts, output coin amounts and carries should be hiding. We define the hiding property of CTx below.

**Functionality 4**: Confidential Transactions

- CTx.**gen**$(pp_{(\lambda,L)}, [v_i, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'_i, \mathtt{mask}'_i, \mathtt{coin}'_i]_{i=1}^{|\mathtt{out}|})$: returns $\mathtt{tx} = (\mathtt{in}, \mathtt{out}, \mathtt{proof})$ if coin values ($v_i$ or $v'_i$) are in the accepted range $[0, 2^L - 1]$, $\sum_{i=1}^{|\mathtt{in}|} v_i = \sum_{i=1}^{|\mathtt{out}|} v'_i$, and input and output coin bundles are valid; otherwise returns $\perp$.
- CTx.**ver**$(pp_{(\lambda,L)}, \mathtt{tx})$: returns 1 if the transaction is valid; otherwise, returns 0.

**Functionality 5**: Random Transaction Generation

**random_tx**$(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|)$:
For any $[v_i]_{i=1}^{|\mathtt{in}|} \in [0, 2^L - 1]$ and $[v'_i]_{i=1}^{|\mathtt{out}|} \in [0, 2^L - 1]$ such that $\sum_{i=1}^{|\mathtt{in}|}[v_i] = \sum_{i=1}^{|\mathtt{out}|}[v'_i] \in [0, 2^L - 1]$:
$[(\mathtt{mask}_i, \mathtt{coin}_i) = \mathrm{COIN}.\mathbf{gen}(pp_{(\lambda,L)}, v_i)]_{i=1}^{|\mathtt{in}|}$
$[(\mathtt{mask}'_i, \mathtt{coin}'_i) = \mathrm{COIN}.\mathbf{gen}(pp_{(\lambda,L)}, v'_i)]_{i=1}^{|\mathtt{out}|}$
$\mathtt{tx} = \mathrm{CTx}.\mathbf{gen}(pp_{(\lambda,L)}, [v_i, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'_i, \mathtt{mask}'_i, \mathtt{coin}'_i]_{i=1}^{|\mathtt{out}|})$
// check inputs and outputs
return $\mathtt{tx.in} \stackrel{?}{=} [\mathtt{coin}]_{i=1}^{|\mathtt{in}|} \bigwedge \mathtt{tx.out} \stackrel{?}{=} [\mathtt{coin}']_{i=1}^{|\mathtt{out}|} \bigwedge$
$\mathrm{CTx}.\mathbf{ver}(pp_{(\lambda,L)}, \mathtt{tx})$

**Definition 11 (Hiding):** CTx is complete if,

$$2\left|Pr\left[\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}()\right] - 1/2\right| \leq \epsilon(\lambda).$$

**Game 9:** Hiding

$\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}(|\mathtt{in}|, |\mathtt{out}|)$:
// get coin amounts from $\mathcal{A}$
$([v_{0,i}]_{i=1}^{|\mathtt{in}|}, [v'_{0,i}]_{i=1}^{|\mathtt{out}|}), ([v_{1,i}]_{i=1}^{|\mathtt{in}|}, [v'_{1,i}]_{i=1}^{|\mathtt{out}|}) \leftarrow \mathcal{A}_{step1}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|)$
**if** $\sum_{i=1}^{|\mathtt{in}|} v_{0,i} \neq \sum_{i=1}^{|\mathtt{out}|} v'_{0,i} \notin [0, 2^L)$
$\bigvee \sum_{i=1}^{|\mathtt{in}|} v_{1,i} \neq \sum_{i=1}^{|\mathtt{out}|} v'_{1,i} \notin [0, 2^L)$
$\bigvee$ given coin amounts are not in $[0, 2^L)$: return $\perp$
$b \stackrel{\$}{\leftarrow} \{0, 1\}$
$[(\mathtt{mask}_i, \mathtt{coin}_i) = \mathrm{COIN}.\mathbf{gen}(pp_{(\lambda,L)}, v_{b,i})]_{i=1}^{|\mathtt{in}|}$
$[(\mathtt{mask}'_i, \mathtt{coin}'_i) = \mathrm{COIN}.\mathbf{gen}(pp_{(\lambda,L)}, v'_{b,i})]_{i=1}^{|\mathtt{out}|}$
$\mathtt{tx}_b = \mathrm{CTx}.\mathbf{gen}(pp_{(\lambda,L)}, [v_{b,i}, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'_{b,i}, \mathtt{mask}'_i, \mathtt{coin}'_i]_{i=1}^{|\mathtt{out}|})$
$b' \leftarrow \mathcal{A}_{step3}(\mathtt{tx}_b)$
return $b \stackrel{?}{=} b'$

Theft resistance of confidential transactions ensures that the given coin bundle cannot be spent without its secret mask. Here, we define a strong version of theft-resistance where the adversarial algorithm chooses the coin amount since the coin value is known in some cases, e.g., coinbase reward.

**Definition 12 (Theft-Resistance):** CTx is theft-resistant if

$$\mathsf{Adv}^{\mathsf{TR},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}} := Pr\left[\mathsf{Game}^{\mathsf{TR},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}()\right] \leq \epsilon(\lambda).$$

**Game 10:** Theft Resistance

$\mathsf{Game}^{\mathsf{TR},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}()$:
$v \leftarrow \mathcal{A}_{step1}(pp_{(\lambda,L)})$
**if** $v \notin [0, 2^{L-1}]$: return $\perp$
$(\mathtt{mask}, \mathtt{coin}) = \mathrm{COIN}.\mathbf{gen}(pp_{(\lambda,L)}, v)$
$\mathtt{tx} \leftarrow \mathcal{A}_{step2}(pp_{(\lambda,L)}, v, \mathtt{coin})$
return $\mathrm{CTx}.\mathbf{ver}(pp_{(\lambda,L)}, \mathtt{tx}) \bigwedge \mathtt{coin} \stackrel{?}{\in} \mathtt{tx.in}$

**Protocol 5:** SIS based Transaction Protocol Initiation

$\mathrm{CTx}_{\mathtt{SIS}}.\mathsf{init}(pp_{(\lambda,L)} = n, m, q, \gamma)$:
Choose $\alpha, \chi, \mathtt{ch}, \tau, \tau_1, \tau_2, \tau_3, \gamma$ such that
// for computational hiding
$(m - 3L)\log(2\tau) \geq 6\lambda$,
// soundness of signature/range-proof challenges
$\mathtt{ch} \geq \log_{2\chi} 2^{2\lambda}$
// to have valid samples after the rejection sampling
$0 < \chi \ll \alpha,\ 0 < \chi^2\tau + \chi\tau_1 \ll \tau_2, 0 < 5\chi\tau \ll \tau_3$
// for security of range proofs
$0 < \alpha - \chi < \alpha^2 \leq \gamma, 0 < \tau_2 \leq \gamma$
// for security of one-time signature
$0 < \tau_3 \leq \gamma$.
$\boldsymbol{H} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$ s.t. $\mathtt{SIS}_{n,m,p,\gamma}$ is hard.
return $pp_{(\lambda,L)} : (n, m, q, \alpha, \chi, \mathtt{ch}, \tau, \tau_1, \tau_2, \tau_3, \gamma, \boldsymbol{H})$

**FIGURE 4.** Public parameters for SIS problem-based confidential transactions.

Zero-coin generation of transactions means that no coins can be generated during a transaction. Here, the extractor $\mathcal{E}$ outputs the hidden coin amount of the coin bundle similar to the extractor of Definition 9.

**Definition 13 (Zero-Coin Generation):** CTx ensures zero-coin generation property if

$$\mathsf{Adv}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}} := Pr\left[\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}()\right] \leq \epsilon(\lambda).$$

**Game 11:** Zero-coin Generation

$\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}()$:
$\mathtt{tx} \leftarrow \mathcal{A}(pp_{(\lambda,L)})$
return $\mathrm{CTx}.\mathbf{ver}(pp_{(\lambda,L)}, \mathtt{tx}) \bigwedge$
$\sum_{i=1}^{|\mathtt{tx.in}|} \mathcal{E}(\mathtt{tx.in}_i) - \sum_{i=1}^{|\mathtt{tx.out}|} \mathcal{E}(\mathtt{tx.out}_i) \stackrel{?}{\neq} 0$

## VIII. AGGREGABLE CONFIDENTIAL TRANSACTIONS WITH GENERIC LATTICES

This section explains the confidential transaction protocol based on SIS problem when coin values are in $[0, 2^L)$. Concrete Protocols related to this section are stated in Figure 4, 5, 8 and 9.

As discussed before, coin bundles include range proofs to ensure that the committed value is in the valid range. If we

**Protocol 6:** SIS Confidential Coin Bundles

$\underline{\text{COIN}_{\text{SIS}}.\mathsf{gen}(pp_{(\lambda,L)}, v)}$:
$\boldsymbol{b} = [b_0, b_2, .., b_{L-1}] = \mathsf{bin}(v)$ s.t. $\sum_{i=0}^{L-1} 2^i b_i = v$
$\boldsymbol{r} \xleftarrow{\$} [-\tau, \tau]^{m-3L}$
$\boldsymbol{s} = [\boldsymbol{b}, \mathbf{0}^{2L}, \boldsymbol{r}] \in \mathbb{Z}^m$
$\boldsymbol{u} = \boldsymbol{H}\boldsymbol{s} \in \mathbb{Z}_q^n$ // Commit the binary form
**for** $i \in [0, \mathtt{ch} - 1]$:
  // Lazy Sampling
  $[a_{i,j} \xleftarrow{\$} [-(\alpha - (1 - b_j)\chi), (\alpha - (1 - b_j)\chi)]]_{j=0}^{L-1}$
  $\hat{\boldsymbol{b}_i} = \boldsymbol{a}_i \circ (2\boldsymbol{b} - \mathbf{1}^L) \in \mathbb{Z}^L$
  $\boldsymbol{r}_{1,i} \xleftarrow{\$} [-\tau_1, \tau_1]^{m-3L}$
  $\boldsymbol{s}_{1,i} = [\mathbf{0}^L, \hat{\boldsymbol{b}_i}, \mathbf{0}^L, \boldsymbol{r}_{1,i}] \in \mathbb{Z}^m$
  $\boldsymbol{T}_{1,i} = \boldsymbol{H}\boldsymbol{s}_{1,i} \in \mathbb{Z}_q^n$
$\boldsymbol{x}_1 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1) \in [-\chi, \chi]^{\mathtt{ch}}$ // Challenge 1
**for** $i \in [0, \mathtt{ch} - 1]$: $\boldsymbol{r}_{2,i} \xleftarrow{\$} [-\tau_2, \tau_2]^{m-3L}$
  $\boldsymbol{s}_{2,i} = [x_{1,i}\boldsymbol{a}_i, \boldsymbol{a}_i \circ \boldsymbol{a}_i, \mathbf{0}^L, \boldsymbol{r}_{2,i}] \in \mathbb{Z}^m$
  $\boldsymbol{T}_{2,i} = \boldsymbol{H}\boldsymbol{s}_{2,i} \in \mathbb{Z}_q^n$
$\boldsymbol{x}_2 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2) \in [-\chi, \chi]^{\mathtt{ch}}$ // Challenge 2
$\boldsymbol{Z} = [\boldsymbol{b}x_{2,i} + \boldsymbol{a}_i]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times L}$
$\boldsymbol{R} = [x_{2,i}(x_{1,i}\boldsymbol{r} + \boldsymbol{r}_{1,i}) + \boldsymbol{r}_{2,i}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times(m-3L)}$
**if** $\|\boldsymbol{Z}\| > (\alpha - \chi) \vee \|\boldsymbol{R}\| > (\tau_2 - \chi^2\tau - \chi\tau_1)$:
  go to Step 3
return $\mathtt{mask} = (\boldsymbol{r}), \mathtt{coin} = (\boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_2)$

$\underline{\text{COIN}_{\text{SIS}}.\mathsf{open}(pp_{(\lambda,L)}, v, \mathtt{mask}, \mathtt{coin})}$ :
$(\boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_2) := \mathtt{coin}$; $\boldsymbol{s} = [\mathsf{bin}(v), \mathbf{0}^{2L}, \mathtt{mask}]$
return $\|\boldsymbol{s}\| \overset{?}{\leq} \tau \bigwedge \boldsymbol{u} \overset{?}{=} \boldsymbol{H}\boldsymbol{s} \in \mathbb{Z}_q^n$

$\underline{\text{COIN}_{\text{SIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{coin})}$ :
$(\boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_2) := \mathtt{coin}$
**if** $\|\boldsymbol{Z}\| > \alpha \vee \|\boldsymbol{R}\| > \tau_2$: return 0
$\boldsymbol{x}_1 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1) \in [-\chi, \chi]^{\mathtt{ch}}$
// Recreate $\boldsymbol{T}_2$
**for** $i \in [0, \mathtt{ch} - 1]$ :
  $\boldsymbol{s}_i = [x_{1,i}\boldsymbol{Z}_i, \boldsymbol{Z}_i \circ (\boldsymbol{Z}_i - x_{2,i}), \mathbf{0}^L, \boldsymbol{R}_i] \in \mathbb{Z}^m$
  $\boldsymbol{T}_{2,i} = \boldsymbol{H}\boldsymbol{s}_i - x_{2,i}(x_{1,i}\boldsymbol{u} + \boldsymbol{T}_{1,i}) \in \mathbb{Z}_q^n$
return $\boldsymbol{x}_2 \overset{?}{=} \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2) \in [-\chi, \chi]^{\mathtt{ch}}$

These coin bundles commit $3L$ elements including extra $\mathbf{0}^L$ to be compatible with aggregated carry proofs.

**FIGURE 5.** SIS problem-based confidential coin bundles.

have a method to prove some bit is either 0 or 1 then we can extend that method for all $L$ bits since we commit the binary form of the coin values. First, we explain how to show a bit is 0 or 1 without revealing the bit.

Assume a prover $\mathcal{P}$ wants to hide $b \in [0, 1]$ but wants to prove to a verifier $\mathcal{V}$ that $b$ is definitely 0 or 1.

1) First, $\mathcal{P}$ chooses a random $a$.
2) $\mathcal{P}$ computes these commitments; $\boldsymbol{u} = \mathtt{commit}([b, 0], r)$ and $\boldsymbol{t}_1 = \mathtt{commit}([0, a(2b-1)], r_1)$ with some masks $r, r_1$. Then $\mathcal{P}$ sends $\boldsymbol{u}$ and $\boldsymbol{t}_1$ to $\mathcal{V}$.
3) $\mathcal{V}$ chooses a random challenge $x_1$ and sends to $\mathcal{P}$.
4) $\boldsymbol{t}_2 = \mathtt{commit}([x_1 a, a^2], r_2)$ with some mask $r_2$. Then $\mathcal{P}$ sends the commitment $\boldsymbol{t}_2$ to $\mathcal{V}$
5) $\mathcal{V}$ chooses a random challenge $x_2$ and sends to $\mathcal{P}$.
6) $\mathcal{P}$ sends $z = bx_2 + a$, $R = x_2(x_1 r + r_1) + r_2$ to $\mathcal{V}$.
7) $\mathcal{V}$ accepts if

$$\mathtt{commit}([x_1 z, z(z - x_2)], R) = x_2(x_1\boldsymbol{u} + \boldsymbol{t}_1) + \boldsymbol{t}_2.$$

If there is $z = bx + a$ for some random $a$ and challenge $x$, then

$$z(z - x) = \underbrace{x^2 b(b-1)}_{0} + xa(2b - 1) + a^2 \tag{7}$$

when $b$ is 0 or 1. Therefore, $\mathcal{P}$ can prove that $b$ is indeed 0 or 1 without revealing $b$ directly.

Before proceeding, we define the range for $a$ to be $[-\alpha, \alpha]$ and the range of the challenge $x_1$ and $x_2$ to be $[-\chi, \chi]$ when $0 < \chi < \alpha$. Now, we can define the safe range for rejection sampling, i.e., $z$ should be $[-(\alpha - \chi), \alpha - \chi]$; otherwise, $z$ leaks information about $b$. If $z$ is out of this range, we will reject the sample and start over. We use lazy sampling for $a$ to reduce the number of rejections. For example,

- $a$ is chosen from $[-(\alpha - \chi), \alpha - \chi]$ when $b$ is 0, and
- $a$ is chosen from $[-\alpha, \alpha]$ when $b$ is 1.

Also, we make sure that $\|z(z - x)\| \leq \gamma$ (recall $\gamma$ is from SIS problem) by $0 < (\alpha - \chi) < \alpha(\alpha - \chi) < \alpha^2 \leq \gamma$. A similar approach should be applied to the masks as well. When $r \in [-\tau, \tau]$, $r_1 \in [-\tau_1, \tau_1]$ and $r_2 \in [-\tau_2, -\tau_2]$, we can define the safe range of $R$ a $[-(\tau_2 - \chi^2\tau - \chi\tau_1), (\tau_2 - \chi^2\tau - \chi\tau_1)]$ when $0 < \chi^2\tau + \chi\tau_1 < \tau$. We state these conditions in Figure 4.

We can see that the soundness error is $2^{-(2\chi)}$ when $x \in [-\chi, \chi]$. Therefore, we repeat the interactions between $\mathcal{P}$ and $\mathcal{V}$ multiple times ($\mathtt{ch} \approx \lceil \log_{2\chi} 2^{2\lambda} \rceil$ times) to reduce the soundness error to $2^{-\lambda}$.

In confidential coin bundles, we commit the binary form $[b_j]_{j=0}^{L-1} \in [0, 1]^L$ of the coin amount instead of a bit $b$, e.g., $\boldsymbol{u} = \mathtt{commit}(\left[[b_j]_{j=0}^{L-1}, \mathbf{0}^{2L}\right], \boldsymbol{r} \in \mathbb{Z}^{m-3L})$ (see Figure 6).

Aggregate CTs should hide all binary values and carries while ensuring other security properties like theft resistance and zero-coin generation. When we examine the structure of carries, we can see that proving the range of carries is similar to proving the coins' range. However, what we want to commit is not input carries ($\boldsymbol{c}_0$) nor output carries ($\boldsymbol{c}_1$) but $[c_{0,j} - 2c_{0,j+1}]_{j=0}^{L-1}$ and $[c_{1,j} - 2c_{1,j+1}]_{j=0}^{L-1}$ to check Equation (1) and Equation (2). Therefore, we commit $[c_{1,j} - 2c_{1,j+1} - c_{0,j} + 2c_{0,j+1}]_{j=0}^{L-1}$ as shown in Figure 7. As we discussed in Section III, a transaction should contain a carry proof for inputs and a carry proof for outputs. In our protocol, transactions have combined carry proofs where the input and output carry proof are combined into a single proof.

We create `proof` (or the header) of the transactions by combining the carry proofs and a multi-signature. A signature

$$\boldsymbol{H}\underbrace{\begin{bmatrix} x_{1,i}\boldsymbol{Z}_i \\ \boldsymbol{Z}_i \circ (\boldsymbol{Z}_i - x_{2,i}) \\ \boldsymbol{0}^L \\ \boldsymbol{R}_i \end{bmatrix}}_{\text{set\_b}(x_{2,i}, x_{1,i}, \boldsymbol{Z}_i, \boldsymbol{R}_i)} = x_{2,i}x_{1,i}\,\boldsymbol{H}\underbrace{\begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0}^L \\ \boldsymbol{0}^L \\ \texttt{mask} = \boldsymbol{r} \end{bmatrix}}_{\boldsymbol{u}} + x_{2,i}\,\boldsymbol{H}\underbrace{\begin{bmatrix} \boldsymbol{0}^L \\ \boldsymbol{a}_i \circ (2\boldsymbol{b} - \boldsymbol{1}^L) \\ \boldsymbol{0}^L \\ \boldsymbol{r}_{1,i} \end{bmatrix}}_{\boldsymbol{T}_{1,i}} + \boldsymbol{H}\underbrace{\begin{bmatrix} x_{1,i}\boldsymbol{a}_i \\ \boldsymbol{a}_i \circ \boldsymbol{a}_i \\ \boldsymbol{0}^L \\ \boldsymbol{r}_{2,i} \end{bmatrix}}_{\boldsymbol{T}_{2,i}}$$

**FIGURE 6.** Coin structure of coin bundle $(Z, R, u, T_1, x_2 = \mathsf{hash}(u, T_1, T_2))$ with its $T_2$ and $x_1 = \mathsf{hash}(u, T_1)$.

$$\boldsymbol{H}\underbrace{\begin{bmatrix} x_{1,i}[(\boldsymbol{Z}_{1,i,j} - \boldsymbol{Z}_{0,i,j} \\ -2(\boldsymbol{Z}_{1,i,j+1} - \\ \boldsymbol{Z}_{0,i,j+1}))]_{j=0}^{L-1} \\ \boldsymbol{Z}_{1,i} \circ (\boldsymbol{Z}_{1,i} - x_{2,i}) \\ \boldsymbol{Z}_{0,i} \circ (\boldsymbol{Z}_{0,i} - x_{2,i}) \\ \boldsymbol{R}_i \end{bmatrix}}_{\text{set\_c}(x_{2,i}, x_{1,i}, \boldsymbol{Z}_{0,i}, \boldsymbol{Z}_{1,i}, \boldsymbol{R}_i)} = x_{2,i}x_{1,i}\,\boldsymbol{H}\underbrace{\begin{bmatrix} [c_{1,j} - c_{0,j} \\ -2(c_{1,j+1} - c_{0,j+1})]_{j=0}^{L-1} \\ \boldsymbol{0}^L \\ \boldsymbol{0}^L \\ \boldsymbol{r} \end{bmatrix}}_{\boldsymbol{u}} + x_{2,i}\,\boldsymbol{H}\underbrace{\begin{bmatrix} \boldsymbol{0}^L \\ \boldsymbol{d}_{1,i} \circ (2\boldsymbol{c}_1 - \boldsymbol{1}^L) \\ \boldsymbol{d}_{0,i} \circ (2\boldsymbol{c}_0 - \boldsymbol{1}^L) \\ \boldsymbol{R}_{1,i} \end{bmatrix}}_{\boldsymbol{T}_{1,i}} + \boldsymbol{H}\underbrace{\begin{bmatrix} x_{1,i}[(d_{1,i,j} - d_{0,i,j} \\ -2(d_{1,i,j+1} \\ -d_{0,i,j+1}))]_{j=0}^{L-1} \\ \boldsymbol{d}_{1,i} \circ \boldsymbol{d}_{1,i} \\ \boldsymbol{d}_{0,i} \circ \boldsymbol{d}_{0,i} \\ \boldsymbol{R}_{2,i} \end{bmatrix}}_{\boldsymbol{T}_{2,i}}$$

$$\boldsymbol{H}\begin{bmatrix} \boldsymbol{0}^L \\ \boldsymbol{0}^{2L} \\ \boldsymbol{\sigma}_i \end{bmatrix} = \boldsymbol{H}\underbrace{\begin{bmatrix} \boldsymbol{0}^L \\ \boldsymbol{0}^{2L} \\ \sum_{j=1}^{|\text{out}|} \boldsymbol{R}'_{3,j,i} - \sum_{j=1}^{|\text{in}|} \boldsymbol{R}_{3,j,i} \end{bmatrix}}_{\boldsymbol{Y}_i} + x_{0,i}\,\boldsymbol{H}\underbrace{\begin{bmatrix} \left[\sum_{j=1}^{|\text{out}|} b'_{j,l} - \sum_{j=1}^{|\text{in}|} b_{j,l} + c_{1,l} - c_{0,l} - 2(c_{1,l+1} - c_{0,l+1})\right]_{l=0}^{L-1} \\ \boldsymbol{0}^{2L} \\ \sum_{j=1}^{|\text{out}|} \texttt{mask}'_j - \sum_{j=1}^{|\text{in}|} \texttt{mask}_j + \texttt{proof}.\boldsymbol{r} \end{bmatrix}}_{\boldsymbol{pk} = \sum_{j=1}^{|\text{out}|} \texttt{coin}'_j.\boldsymbol{u} - \sum_{j=1}^{|\text{in}|} \texttt{coin}_j.\boldsymbol{u} + \texttt{proof}.\boldsymbol{u}}$$

**FIGURE 7.** Carry proof structure of $(Z_0, Z_1, R, u, T_1, x_0 = \mathsf{hash}(pk, Y), x_2 = \mathsf{hash}(u, T_1, T_2, \sigma), \sigma, pk)$ with $(T_2, Y, x_1 = \mathsf{hash}(u, T_1, \sigma))$.

$\boldsymbol{\sigma}$ is created for the aggregate masking key of the inputs', outputs', and carries' masks (see Step 44). Assume Alice wants to send coins to two receivers, Bob and Charles, and they have a secure communication channel where none of them can impersonate others. During the transaction generation, Alice should be able to hide the new mask from others, and the same goes for Bob and Charles as well. On the other hand, they have to show their knowledge of the masks to compute the multi-signature.

Therefore, each sender and receiver computes partial signatures separately with a signature mask in $[-\tau_3, \tau_3]$. In the end, they aggregate partial signatures to compute the multi-signature. We perform rejection sampling in Step 40 and Step 44 for partial signatures to make sure that partial signatures do not leak information. Therefore, the safe range for a partial signature is $[-(\tau_3 - \chi\tau), \tau_3 - \chi\tau]$ (recall that coin masks are in $[-\tau, \tau]$). In that way, they can hide the mask of the coin bundle from each other. Aggregated signatures are also subjected to rejection sampling because the knowledge of aggregate keys may leak the proof's masking key.

The protocol provides the following security measures.

- The sender, Alice, can stop the transaction by not sharing her signature.
- Bob (or Charles) cannot update the created transaction to steal Charles' (or Bob's) coins due to the challenge $\boldsymbol{x}_1$, i.e., changing any $\boldsymbol{u}$ of a coin bundles changes $\boldsymbol{x}_1$ of that coin bundle.
- No sender or receiver can see or derive others' secret masks due to the rejection sampling on partial signatures.

We define the concrete non-interactive confidential coin bundle scheme COIN in Figure 5 and our aggregate CT protocol CTx in Figure 8 and Figure 9. Here, "non-interactive" means that the prover(s) gets challenges from a hash function instead of the verifier, e.g., $\boldsymbol{x}_1 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1)$ and $\boldsymbol{x}_2 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2)$ from Figure 6. Since we have a precise method to recompute challenges, we do not include $\boldsymbol{T}_2$ in the coin bundle. Instead, we include $\boldsymbol{x}_2$ and recompute $\boldsymbol{T}_2$ using $\boldsymbol{x}_2$ (see Step 33). At the end, we check whether the hash challenge $\boldsymbol{x}_2$ is equal to $\mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2)$ or not. This method *saves* space since $\boldsymbol{T}_2$ is a few kilobytes while $\boldsymbol{x}_2$ is just 32 bytes.

We use a similar method for the carry proofs and the signatures as well. For example, the recomputable hash challenge of a signature is computed as $\boldsymbol{x}_0 = \mathsf{hash}(\boldsymbol{pk}, \boldsymbol{Y})$. To save space, we include $\boldsymbol{x}_0$, not $\boldsymbol{Y}$. During the signature verification, we recompute $\boldsymbol{Y}$ and check whether $\boldsymbol{x}_0$ is equal to $\mathsf{hash}(\boldsymbol{pk}, \boldsymbol{Y})$ or not (see Step 92).

We illustrate the correctness in Figure 6 and Figure 7.

## IX. SECURITY PROOFS FOR SIS PROBLEM BASED TRANSACTIONS

This section shows the security of our confidential coin bundles and confidential transactions.

**Protocol 7:** Proofs of SIS Confidential Transactions

This protocol shows how to compute and verify transactions when both input carries and output carries should be computed. When there is only one input or only one output, we do not compute their range proofs and commit zero vectors instead.

// $|\mathtt{in}|, |\mathtt{out}|$ ($\leq 2$) are the number of inputs and outputs.
// Senders and receivers do not share their $\mathtt{mask}$ or $\mathtt{mask}'$ with other senders or receivers in any step.
$\mathrm{CTx}_{\mathtt{SIS}}.\mathsf{proof}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, \underline{[v^{(i)}, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'^{(i)}]_{i=1}^{|\mathtt{out}|}}$ ):
$[\boldsymbol{b}_i := \mathsf{bin}(v^{(i)})]_{i=1}^{|\mathtt{in}|}, [\boldsymbol{b}'_i := \mathsf{bin}(v'^{(i)})]_{i=1}^{|\mathtt{out}|}$
// Carries when $\div$ is the integer division, e.g., $3 \div 2 = 1$
$\boldsymbol{c}_0 = \left[0, [c_{0,j+1} = \left(\sum_{i=1}^{|\mathtt{in}|} b_{i,j} + c_{0,j}\right) \div 2]_{j=0}^{L-2}, 0\right]$
$\boldsymbol{c}_1 = \left[0, [c_{1,j+1} = \left(\sum_{i=1}^{|\mathtt{out}|} b'_{i,j} + c_{1,j}\right) \div 2]_{j=0}^{L-2}, 0\right]$
// All senders and receivers compute the followings
**for** $i \in [0, \mathtt{ch}-1]$ : // Lazy Sampling
  $[[d_{0,i,l} \xleftarrow{\$} [-(\alpha-(1-c_{0,l})\chi), (\alpha-(1-c_{0,l})\chi)]]_{l=1}^{L-1}$
  Set $d_{0,i,0} = d_{0,i,L} = 0$
  $[d_{1,i,l} \xleftarrow{\$} [-(\alpha-(1-c_{1,l})\chi), (\alpha-(1-c_{1,l})\chi)]]_{l=1}^{L-1}$
  Set $d_{1,i,0} = d_{1,i,L} = 0$
  $\hat{\boldsymbol{c}}_{0,i} = [d_{0,i,j} \circ (2c_{0,j}-1)]_{j=0}^{L-1} \in \mathbb{Z}^L$
  $\hat{\boldsymbol{c}}_{1,i} = [d_{1,i,j} \circ (2c_{1,j}-1)]_{j=0}^{L-1} \in \mathbb{Z}^L$
  $\hat{\boldsymbol{d}}_{0,i} = [d_{0,i,j} \circ d_{0,i,j}]_{j=0}^{L-1} \in \mathbb{Z}^L$
  $\hat{\boldsymbol{d}}_{1,i} = [d_{1,i,j} \circ d_{1,i,j}]_{j=0}^{L-1} \in \mathbb{Z}^L$
$\hat{\boldsymbol{c}}' = [(c_{1,j} - c_{0,j}) - 2(c_{1,j+1} - c_{0,j+1})]_{j=0}^{L-1} \in \mathbb{Z}^L$
$\hat{\boldsymbol{d}}' = [[(d_{1,i,j} - d_{0,i,j}) - 2(d_{1,i,j+1} - d_{0,i,j+1})]_{j=0}^{L-1}]_{i=0}^{\mathtt{ch}-1}$
// Masking keys for the carries
$\boldsymbol{r} \xleftarrow{\$} [-\tau, \tau]^{m-3L}$
$\boldsymbol{R}_1 \xleftarrow{\$} [-\tau_1, \tau_1]^{\mathtt{ch}\times(m-3L)}$
$\boldsymbol{R}_2 \xleftarrow{\$} [-\tau_2, \tau_2]^{\mathtt{ch}\times(m-3L)}$
$\boldsymbol{u} = \boldsymbol{H}[\hat{\boldsymbol{c}}', \boldsymbol{0}^{2L}, \boldsymbol{r}] \in \mathbb{Z}_q^n$
Each sender $j \in [1, |\mathtt{in}|]$ secretly computes:
  ▷ $\boldsymbol{R}_{3,j} \xleftarrow{\$} [-\tau_3, \tau_3]^{\mathtt{ch}\times(m-3L)}$
  ▷ share $\boldsymbol{Y}_j = [\boldsymbol{H}[\boldsymbol{0}^{3L}, \boldsymbol{R}_{3,j,i}]]_{i=0}^{\mathtt{ch}-1}$ with others
Each receiver $j \in [1, |\mathtt{out}|]$ secretly computes:
  ▷ $(\mathtt{mask}'_j, \mathtt{coin}'_j) := \mathrm{COIN}.\mathsf{gen}(pp_{(\lambda,L)}, v'_j)$
  ▷ $\boldsymbol{R}'_{3,j} \xleftarrow{\$} [-\tau_3, \tau_3]^{\mathtt{ch}\times(m-3L)}$
  ▷ share $\mathtt{coin}'_j$ and $\boldsymbol{Y}'_j = [\boldsymbol{H}[\boldsymbol{0}^{3L}, \boldsymbol{R}'_{3,j,i}]]_{i=0}^{\mathtt{ch}-1}$
// Public key of the transaction
$\boldsymbol{pk} = \boldsymbol{u} + \sum_{j=1}^{|\mathtt{out}|} \mathtt{coin}'_j.\boldsymbol{u} - \sum_{j=1}^{|\mathtt{in}|} \mathtt{coin}_j.\boldsymbol{u} \in \mathbb{Z}_q^n$
$\boldsymbol{Y} = [\sum_{j=1}^{|\mathtt{out}|} \boldsymbol{Y}'_{j,i} - \sum_{j=1}^{|\mathtt{in}|} \boldsymbol{Y}_{j,i}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
$\boldsymbol{x}_0 = \mathsf{hash}(\boldsymbol{pk}, \boldsymbol{Y}) \in [-\chi, \chi]^{\mathtt{ch}}$ // Signature challenge
Each sender $j \in [1, |\mathtt{in}|]$ secretly computes:
  ▷ $\boldsymbol{\sigma}_j = [\boldsymbol{R}_{3,j,i} + x_{0,i}\mathtt{mask}_j]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times(m-3L)}$
  ▷ **if** $\|\boldsymbol{\sigma}_j\| > \tau_3 - \chi\tau$:
      go to Step 9
Each receiver $j \in [1, |\mathtt{out}|]$ secretly computes:
  ▷ $\boldsymbol{\sigma}'_j = [\boldsymbol{R}'_{3,j,i} + x_{0,i}\mathtt{mask}'_j]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times(m-3L)}$
  ▷ **if** $\|\boldsymbol{\sigma}'_j\| > \tau_3 - \chi\tau$:
      go to Step 9
// The signature is $(\boldsymbol{\sigma} \in \mathbb{Z}^{\mathtt{ch}\times(m-3L)}, \boldsymbol{x}_0)$
$\boldsymbol{\sigma} = [x_{0,i}\boldsymbol{r} + \sum_{j=1}^{|\mathtt{out}|} \boldsymbol{\sigma}'_{j,i} - \sum_{j=1}^{|\mathtt{in}|} \boldsymbol{\sigma}_{j,i}]_{i=0}^{\mathtt{ch}-1}$
**if** $\|\boldsymbol{\sigma}\| > (|\mathtt{in}| + |\mathtt{out}|)\tau_3 - (|\mathtt{in}| + |\mathtt{out}| + 1)\chi\tau$:
  go to Step 9
$\boldsymbol{T}_1 = [\boldsymbol{H}[\boldsymbol{0}^L, \hat{\boldsymbol{c}}_{1,i}, \hat{\boldsymbol{c}}_{0,i}, \boldsymbol{R}_{1,i}]]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
$\boldsymbol{x}_1 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{\sigma}) \in [-\chi, \chi]^{\mathtt{ch}}$ // Challenge 1
$\boldsymbol{T}_2 = [\boldsymbol{H}[x_{1,i}\hat{\boldsymbol{d}}'_i, \hat{\boldsymbol{d}}_{1,i}, \hat{\boldsymbol{d}}_{0,i}, \boldsymbol{R}_{2,i}]]_{i=0}^{\mathtt{ch}} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
$\boldsymbol{x}_2 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2, \boldsymbol{\sigma}) \in [-\chi, \chi]^{\mathtt{ch}}$ // Challenge 2
$\boldsymbol{Z}_0 = [[c_{0,j}x_{2,i} + d_{0,i,j}]_{j=1}^{L-1}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times(L-1)}$
**if** $\|\boldsymbol{Z}_0\| > (\alpha - \chi)$:
  go to Step 9
$\boldsymbol{Z}_1 = [[c_{1,j}x_{2,i} + d_{1,i,j}]_{j=1}^{L-1}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times(L-1)}$
**if** $\|\boldsymbol{Z}_1\| > (\alpha - \chi)$:
  go to Step 9
$\boldsymbol{R} = [x_{2,i}(x_{1,i}\boldsymbol{r} + \boldsymbol{R}_{1,i}) + \boldsymbol{R}_{2,i}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times(m-3L)}$
**if** $\|\boldsymbol{R}\| > (\tau_2 - \chi^2\tau - \chi\tau_1)$:
  go to Step 9
return $\mathtt{proof} = (\boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{pk}, \boldsymbol{\sigma})$,
    $\mathtt{out} = [\mathtt{coin}']_{i=1}^{|\mathtt{out}|}$
// **if** $|\mathtt{in}| = 1 \wedge |\mathtt{out}| = 2$ :
  $\mathtt{proof} = (\boldsymbol{Z}_1, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{pk}, \boldsymbol{\sigma})$
// **if** $|\mathtt{in}| = 2 \wedge |\mathtt{out}| = 1$ :
  $\mathtt{proof} = (\boldsymbol{Z}_0, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{pk}, \boldsymbol{\sigma})$
// **else**: $\mathtt{proof} = (\boldsymbol{x}_0, \boldsymbol{pk}, \boldsymbol{\sigma})$

$\mathrm{CTx}_{\mathtt{SIS}}.\mathsf{proof\_ver}(pp_{(\lambda,L)}, \mathtt{proof})$ :
// $(*)$ denotes optional variables
$(\boldsymbol{Z}_0^{(*)}, \boldsymbol{Z}_1^{(*)}, \boldsymbol{R}^{(*)}, \boldsymbol{u}^{(*)}, \boldsymbol{T}_1^{(*)}, \boldsymbol{x}_0, \boldsymbol{x}_2^{(*)}, \boldsymbol{pk}, \boldsymbol{\sigma})$ : $\mathtt{proof}$
**if** $\|\boldsymbol{\sigma}\| > \tau_3$: return 0
**if** $|\mathtt{in}| \stackrel{?}{=} 2 \vee |\mathtt{out}| \stackrel{?}{=} 2$:
  **if** $\|\boldsymbol{R}\| > \tau_2$: return 0
  $\boldsymbol{x}_1 = \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{\sigma}) \in [-\chi, \chi]^{\mathtt{ch}}$
  Set $\hat{\boldsymbol{Z}}_0 = \boldsymbol{0}^{\mathtt{ch}\times L}$, $\hat{\boldsymbol{Z}}_1 = \boldsymbol{0}^{\mathtt{ch}\times L}$ and $\hat{\boldsymbol{Z}}' = \boldsymbol{0}^{\mathtt{ch}\times L}$
  **if** $|\mathtt{in}| \stackrel{?}{=} 2$:
    **if** $\|\boldsymbol{Z}_0\| > \alpha$: return 0
    Set $\boldsymbol{Z}_{0,i,0}$ and $\boldsymbol{Z}_{0,i,L}$ to 0
    $\hat{\boldsymbol{Z}}_0 = [[\boldsymbol{Z}_{0,i,j} \circ (\boldsymbol{Z}_{0,i,j} - x_{2,i})]_{j=0}^{L-1}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times L}$
    $\hat{\boldsymbol{Z}}' = [[\boldsymbol{Z}_{0,i,j} - 2\boldsymbol{Z}_{0,i,j+1}]_{j=0}^{L-1}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times L}$
  **if** $|\mathtt{out}| \stackrel{?}{=} 2$:
    **if** $\|\boldsymbol{Z}_1\| > \alpha$: return 0
    Set $\boldsymbol{Z}_{1,i,0}$ and $\boldsymbol{Z}_{1,i,L}$ to 0
    $\hat{\boldsymbol{Z}}_1 = [[\boldsymbol{Z}_{1,i} \circ (\boldsymbol{Z}_{1,i,j} - x_{2,i,j})]_{j=0}^{L-1}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times L}$
    $\hat{\boldsymbol{Z}}' = [[\boldsymbol{Z}_{1,i,j} - 2\boldsymbol{Z}_{1,i,j+1}]_{j=0}^{L-1} - \hat{\boldsymbol{Z}}'_i]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times L}$
  // Recompute $\boldsymbol{T}_2$
  $\boldsymbol{S} = [[x_{1,i}\hat{\boldsymbol{Z}}'_i,\ \hat{\boldsymbol{Z}}_{1,i},\ \hat{\boldsymbol{Z}}_{0,i},\ \boldsymbol{R}_i]]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}^{\mathtt{ch}\times m}$
  $\boldsymbol{T}_2 = [\boldsymbol{H}\boldsymbol{S}_i - x_{2,i}(x_{1,i}\boldsymbol{u} + \boldsymbol{T}_{1,i})]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
  **if** $\boldsymbol{x}_2 \neq \mathsf{hash}(\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2, \boldsymbol{\sigma})$: return 0
// Recompute $\boldsymbol{Y}$
$\boldsymbol{Y} = [\boldsymbol{H}[\boldsymbol{0}^{3L}, \boldsymbol{\sigma}_i] - x_{0,i}\boldsymbol{pk}]_{i=0}^{\mathtt{ch}-1} \in \mathbb{Z}_q^{\mathtt{ch}\times n}$
return $\boldsymbol{x}_0 \stackrel{?}{=} \mathsf{hash}(\boldsymbol{pk}, \boldsymbol{Y})$

**FIGURE 8.** Proofs (headers) of aggregate confidential transactions.

**Protocol 8:** SIS Confidential Transactions

1: // $|\mathtt{in}|, |\mathtt{out}|$ ($\leq 2$) are the number of inputs and outputs.
2: // Senders and receivers do not share their $\mathtt{mask}$ or $\mathtt{mask}'$ with other senders or receivers in any step.
3: $\mathrm{CTx}_{\mathtt{SIS}}.\mathsf{gen}(pp_{(\lambda,L)}, [v_i, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'_i]_{i=1}^{|\mathtt{out}|})$:
4: $(\mathtt{proof}, \mathtt{out}) = \mathrm{CTx}.\mathsf{proof}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|,$
5: $\qquad [v^{(i)}, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'^{(i)}]_{i=1}^{|\mathtt{out}|})$
6: return $\mathtt{tx} = (\mathtt{in} = [\mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, \mathtt{out}, \mathtt{proof})$

7: $\mathrm{CTx}_{\mathtt{SIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{tx})$ :
8: $(\boldsymbol{Z}_0, \boldsymbol{Z}_1, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{pk}, \boldsymbol{\sigma}) := \mathtt{tx}.\mathtt{proof}$
9: // Verify input and output coin bundles
10: **if** $[\mathtt{coin} \in \mathtt{tx}.\mathtt{in} : \neg \mathrm{COIN}_{\mathtt{SIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{coin})] \bigvee$
11: $\quad [\mathtt{coin}' \in \mathtt{tx}.\mathtt{out} : \neg \mathrm{COIN}_{\mathtt{SIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{coin}')]$ :
12: $\quad$ return 0
13: // Verify the proof of the transaction
14: **if** $\neg \mathrm{CTx}_{\mathtt{SIS}}.\mathsf{proof\_ver}(pp_{(\lambda,L)}, \mathtt{tx}.\mathtt{proof})$ : return 0
15: return $\boldsymbol{pk} \stackrel{?}{=} \sum_{\mathtt{c}' \in \mathtt{tx}.\mathtt{out}} \mathtt{c}'.\boldsymbol{u} - \sum_{\mathtt{c} \in \mathtt{tx}.\mathtt{in}} \mathtt{c}.\boldsymbol{u} + \boldsymbol{u} \in \mathbb{Z}_q^n$

**FIGURE 9.** SIS aggregate confidential transactions.

### A. SECURITY PROOFS FOR CONFIDENTIAL COIN BUNDLES

**Theorem 6:** $\mathrm{COIN}_{\mathtt{SIS}}$ *is complete, binding, hiding, and knowledge sound if* $\mathrm{COM}_{\mathtt{SIS}}$ *is binding and hiding, and* $\mathtt{SIS}_{n,m,q,\gamma}$ *is hard.*

We give the proof in a series of lemmas below. We should be able to verify and open the honestly generated confidential coin bundles as defined in Definition 6. Here, "honestly generated" includes (1) the coin value is in $[0, 2^L)$, (2) the commitment contains the correct binary form of the coin value, and (3) the random masks are chosen from the defined range. We directly see the opening of a coin bundle is correct. We show the correctness of verification function in Figure 6. Therefore, we conclude $\mathrm{COIN}_{\mathtt{SIS}}$ is complete.

**Lemma 1:** $\mathrm{COIN}_{\mathtt{SIS}}$ *is binding if* $\mathrm{COM}_{\mathtt{SIS}}$ *is binding.*

*Proof:* Assume $\mathrm{COIN}_{\mathtt{SIS}}$ is not binding as stated in Definition 7, then there is a p.p.t. algorithm $\mathcal{A}$ which finds two different openings for the same confidential coin bundle. Then $\mathcal{A}$ can be used to break the binding property of $\mathrm{COM}_{\mathtt{SIS}}$. Therefore, we conclude that $\mathrm{COIN}_{\mathtt{SIS}}$ is binding if $\mathrm{COM}_{\mathtt{SIS}}$ is hard, or Lemma 1 is valid. □

**Lemma 2:** $\mathrm{COIN}_{\mathtt{SIS}}$ *is computationally hiding if* $\mathrm{COM}_{\mathtt{SIS}}$ *is computationally hiding.*

*Proof:* This proof has two parts; (1) $\boldsymbol{Z}$ statistically hides the coin amount, and (2) $\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2$ are computationally hiding.

First, we prove that there is exactly one possible $[\boldsymbol{a}_i]_{i=0}^{\mathtt{ch}-1} \in [-\alpha, \alpha]^{\mathtt{ch} \times L}$ for any combination of $(\boldsymbol{b}, \boldsymbol{Z})$ and any $\boldsymbol{x}_2$. Since $b_j \in [0, 1]$, $x_{2,i} \in [-\chi, \chi]$, and $\boldsymbol{Z}_{i,j} \in [-\alpha + \chi, \alpha - \chi]$; for $j \in [0, L)$,

- When $b_j = 0$: $\boldsymbol{a}_{i,j} = \boldsymbol{Z}_{i,j} \in [-\alpha, \alpha]$
- When $b_j = 1$: $\boldsymbol{a}_{i,j} = \boldsymbol{Z}_{i,j} - x_{2,i} \in [-\alpha, \alpha]$

Secondly, we show that $\boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2$ computationally hide their committed values. Let there be a p.p.t. algorithm that wins $\mathsf{Game}^{\mathsf{HID},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}}$. Then $\mathcal{A}$ can be used to break the hiding property of $\mathrm{COM}_{\mathtt{SIS}}$ when $d = 3L$ since $\tau < \tau_1 < \tau_2$ and $(m - 3L)\log(2\tau) \geq 6\lambda$.

Therefore, $\mathrm{COIN}_{\mathtt{SIS}}$ is computationally hiding if $\mathrm{COM}_{\mathtt{SIS}}$ is computationally hiding. □

**Lemma 3:** $\mathrm{COIN}_{\mathtt{SIS}}$ *is computationally knowledge sound if* $\mathrm{COM}_{\mathtt{SIS}}$ *is binding,* $\mathtt{SIS}_{n,m,q,\gamma}$ *is hard, and* $\mathtt{ch}\log(2\chi) \geq 2\lambda$.

*Proof:* Let there be $\mathcal{A}$ that wins $\mathsf{Game}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}}$ when the extractor $\mathcal{E}(\mathtt{coin})$ outputs $\boldsymbol{b} \notin \{0, 1\}^L$. Then $\boldsymbol{b} \circ (\boldsymbol{b} - 1)$ is not equal to $\boldsymbol{0}^L$. In other words, when $\mathrm{COM}_{\mathtt{SIS}}$ is binding, $\mathcal{A}$ finds some $(\boldsymbol{Z}, \boldsymbol{R}, \boldsymbol{u}, \boldsymbol{T}_1, \boldsymbol{T}_2)$ s.t. for each $i \in [0, \mathtt{ch})$:

$$\boldsymbol{H} \begin{bmatrix} x_{1,i}\boldsymbol{Z}_i \\ \boldsymbol{Z}_i \circ (\boldsymbol{Z}_i - x_{2,i}) \\ \boldsymbol{0}^L, \boldsymbol{R}_i \end{bmatrix} = x_{2,i}^2 \underbrace{\boldsymbol{H} \underbrace{\begin{bmatrix} \boldsymbol{0}^L \\ \boldsymbol{b} \circ (\boldsymbol{b} - 1) \\ \boldsymbol{0}^{m-2L} \end{bmatrix}}_{\boldsymbol{s} \neq \boldsymbol{0}^m}}_{=\boldsymbol{0}^n} + \begin{pmatrix} x_{2,i}x_{1,i}\boldsymbol{u} \\ +x_{2,i}\boldsymbol{T}_{1,i} \\ +\boldsymbol{T}_{2,i} \end{pmatrix}$$

when $\boldsymbol{x}_1, \boldsymbol{x}_2 \in [-\chi, \chi]^{\mathtt{ch}}$ are given by $\mathsf{hash}$ (see Step 17 of $\mathrm{COIN}_{\mathtt{SIS}}.\mathsf{gen}()$). We highlight that $\boldsymbol{s} \neq \boldsymbol{0}^m$ is a solution to $\mathtt{SIS}_{n,m,q,\gamma}$ of $\boldsymbol{H}$. Therefore, we reduce $\mathcal{A}$'s winning probability after *one execution* as follows,

$$\mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}} \leq \mathsf{Adv}^{\mathtt{SIS},\mathcal{A}}_{pp_\lambda} + \frac{1}{(2\chi)^{\mathtt{ch}}} + \mathsf{Adv}^{\mathsf{BND},\mathcal{A}}_{\mathrm{COM},pp_{(\lambda,3L)}}.$$

Hence, we claim Lemma 3 is correct or $\mathrm{COIN}_{\mathtt{SIS}}$ is computationally knowledge sound. □

### B. SECURITY PROOFS FOR CONFIDENTIAL TRANSACTIONS

**Theorem 7:** $\mathrm{CTx}_{\mathtt{SIS}}$ *is complete, theft-resistant, hiding, and provides zero coin generation property if* $\mathrm{COIN}_{\mathtt{SIS}}$ *is complete, binding, hiding, and knowledge sound,* $\mathrm{SIG}_{\mathtt{SIS}}$ *is complete and EUF-CMA, and* $\mathtt{SIS}_{n,m,q,\gamma}$ *is hard.*

We prove each security property of $\mathrm{CTx}_{\mathtt{SIS}}$ individually.

We start with the completeness of $\mathrm{CTx}_{\mathtt{SIS}}$. The completeness of $\mathrm{CTx}_{\mathtt{SIS}}$ states that the verification function accepts correctly generated transactions. Here, we focus on two verification functions; (1) proof-verification, which checks the range of the carries, and (2) transaction verification that checks whether total coin summation is zero or not. We visualize the correctness of $\mathrm{CTx}_{\mathtt{SIS}}$ in Figure 7.

**Lemma 4:** $\mathrm{CTx}_{\mathtt{SIS}}$ *of* $(n, m, q, \gamma)$ *is resistant to theft if* $\mathrm{SIG}_{\mathtt{SIS}}$ *of* $(n, m - 3L, q, \gamma)$ *is EUF-CMA.*

*Proof:* If $\mathcal{A}$ steals coins without knowing the secret mask then we can use $\mathcal{A}$ in a simulation to break EUF-CMA of $\mathrm{SIG}_{\mathtt{SIS}}$. Therefore, we claim that $\mathrm{CTx}_{\mathtt{SIS}}$ is theft resistant when $\mathrm{SIG}_{\mathtt{SIS}}$ is EUF-CMA. □

**Lemma 5:** $\mathrm{CTx}_{\mathtt{SIS}}$ *is computationally hiding when* $\mathrm{COM}_{\mathtt{SIS}}$ *is computationally hiding.*

*Proof:* We claim that carry proofs are hiding similar to the confidential coin bundles since (1) the smallest random mask $\tau$ satisfies the condition; $(m-3L)\log(2\tau) \geq 6\lambda$, and (2) any pair of $(\boldsymbol{Z}_0, \boldsymbol{Z}_1)$ statistically hides the input carries and output carries due to the rejection sampling. □

**Lemma 6:** *Breaking* $\mathrm{CTx}_{\mathtt{SIS}}$*'s zero coin generation property is computationally infeasible if* $\mathrm{COIN}_{\mathtt{SIS}}$ *is knowledge sound,* $\mathrm{SIG}_{\mathtt{SIS}}$ *of* $(n, m-3L, q, \gamma)$ *is EUF-CMA,* $\mathtt{SIS}_{pp_\lambda}$ *is hard for* $pp_\lambda = (n, m, q, \gamma)$*, and* $\mathtt{ch}\log(2\chi) \geq 2\lambda$.

*Proof:* Let there be a p.p.t. algorithm $\mathcal{A}$ that breaks the zero coin generation property when $\mathrm{COIN}_{\mathtt{SIS}}$ is knowledge sound, and $\mathrm{SIG}_{\mathtt{SIS}}$ is EUF-CMA. Assume the extractor $\mathcal{E}$ of $\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}$ outputs inputs $[v_i]_{i=1}^{|\mathtt{in}|}$ and outputs $[v'_i]_{i=1}^{|\mathtt{out}|}$ such that, $\sum_{i=1}^{|\mathtt{in}|} v_i \neq \sum_{i=1}^{|\mathtt{out}|} v'_i$.

Here, we assume $\mathrm{COIN}_{\mathtt{SIS}}$ is knowledge sound. We can replace coin values with their binary forms $([\boldsymbol{b}_i]_{i=1}^{|\mathtt{in}|}, [\boldsymbol{b}'_i]_{i=1}^{|\mathtt{out}|})$ and their **real** carries $(\boldsymbol{c}_0, \boldsymbol{c}_1)$,

$$\left[\sum_{i=1}^{\mathtt{in}} b_{i,j}+c_{0,j}-2c_{0,(j+1)} \neq \sum_{i=1}^{\mathtt{out}} b'_{i,j}+c_{1,j}-2c_{1,(j+1)}\right]_{j=0}^{L-1} \tag{8}$$

Note that there are two verification steps; checking (1) the zero coin summation and (2) signature verification. If $\mathcal{A}$ creates a valid transaction then one of the following scenarios can happen.

1) $\mathcal{A}$ commits some other $(\boldsymbol{c}'_0, \boldsymbol{c}'_1) \neq (\boldsymbol{c}_0, \boldsymbol{c}_1)$ s.t. for all $j \in [0, L)$:

$$\sum_{i=1}^{\mathtt{in}} b_{i,j} + c'_{0,j} - 2c'_{0,(j+1)} = \sum_{i=1}^{\mathtt{out}} b'_{i,j} + c'_{1,j} - 2c'_{1,(j+1)}.$$

These $(\boldsymbol{c}'_0, \boldsymbol{c}'_1)$ cannot be a pair of binary vectors since $(\boldsymbol{c}_0, \boldsymbol{c}_1)$ are the only one binary vector pair that satisfies Equation (8). Therefore, $\boldsymbol{c}'_0(\boldsymbol{c}'_0-1) \neq \boldsymbol{0}^m$ and/or $\boldsymbol{c}'_1(\boldsymbol{c}'_1-1) \neq \boldsymbol{0}^m$. Note that,

$$\underbrace{x_{2,i}^2 \boldsymbol{H} \underbrace{[\boldsymbol{0}^L, (\boldsymbol{c}_1(\boldsymbol{c}_1-1), \boldsymbol{c}_0(\boldsymbol{c}_0-1)), \boldsymbol{0}^{m-3L}]}_{\boldsymbol{s}\neq\boldsymbol{0}^m} = \boldsymbol{0}^n}_{\boldsymbol{h}}$$

due to the carry proof verification (Step 89). Therefore, $\boldsymbol{x}_2 = \boldsymbol{0}^{\mathtt{ch}}$ or $\boldsymbol{h} = \boldsymbol{0}^n$. Note that $\boldsymbol{s}$ is a solution to the $\mathtt{SIS}$ problem if $\boldsymbol{h} = \boldsymbol{0}^n$ since $\boldsymbol{s} \leq \gamma$.

2) $\mathcal{A}$ finds

$$\underbrace{\boldsymbol{H}\begin{bmatrix} \left[\sum_{i=1}^{\mathtt{out}} b'_{i,j} - \sum_{i=1}^{\mathtt{in}} b_{i,j} + c_{1,j} - c_{0,j} \right. \\ \left. -2(c_{1,(j+1)} - 2c_{0,(j+1)})\right]_{j=0}^{L-1}, \boldsymbol{0}^{m-L} \end{bmatrix}}_{=\boldsymbol{s}\neq\boldsymbol{0}^m} = \boldsymbol{0}^n$$

due to the signature verification (Step 92). Here, $\boldsymbol{s} \neq \boldsymbol{0}^m$ is a solution to the $\mathtt{SIS}$ problem.

3) $\mathcal{A}$ breaks the knowledge soundness of $\mathrm{COIN}_{\mathtt{SIS}}$.

**Protocol 9:** MSIS based Transaction Protocol Initiation

1: $\mathsf{init}(pp_{(\lambda,L)} = n, m, q, N, \gamma)$:
2: Choose $\mathcal{C}^N_{\beta,1}, \alpha, \tau, \tau_1, \tau_2, \tau_3$ such that
3: // for fully split polynomial $(X^N+1)$ over prime $q$
4: $q \equiv 1 \pmod{2N}$
5: // for negligible soundness error
6: $\log_2\binom{N}{\beta} + \beta \geq 2\lambda$,
7: // for rejection sampling in range proofs
8: $0 < 2 \ll \alpha,\ 0 < \beta^2\tau + \beta\tau_1 \ll \tau_2 \leq \gamma$,
9: // for rejection sampling in signatures
10: $0 < 5\beta\tau \ll \tau_3 \leq \gamma$,
11: // for computational hiding of coin amounts
12: $(m-3)N\log_2(2\tau) \geq 6\lambda$,
13: // skipped norm check, e.g., Step 48 in Figure 14
14: $L\alpha \ll \gamma$
15: $\vec{\boldsymbol{H}} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$ s.t. $\mathtt{MSIS}_{n,m,q,N,\gamma}$ is hard.
16: // challenge space
17: $\mathcal{C}^N_{\beta,1} = [x \in \mathcal{R}_q \text{ s.t. } \|x\| = 1, \|x\|_1 = \beta]$
18: return $pp_{(\lambda,L)} = (n, m, q, N, \mathcal{C}^N_{\beta,1}, \alpha, \tau_1, \tau_2, \tau_3, \gamma, \vec{\boldsymbol{H}})$

**FIGURE 10.** Public parameters for $\mathtt{MSIS}$ problem based confidential transactions.

Hence, we reduce $\mathcal{A}$'s advantage to,

$$\mathsf{Adv}^{\mathcal{A},\mathtt{carry}}_{\mathrm{CTx},pp_{(\lambda,L)}} \leq \begin{bmatrix} \mathsf{Adv}^{\mathtt{SIS},\mathcal{A}}_{pp_\lambda} + \frac{1}{(2\chi)^{\mathtt{ch}}} \\ +\mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG},pp_\lambda} \end{bmatrix}.$$

Considering above scenarios, we claim the advantage is $\epsilon(\lambda)$ when $\mathtt{SIS}_{pp_\lambda}$ is hard, and $\mathtt{ch}\log(2\chi+1) \geq 2\lambda$. Therefore, we claim that $\mathrm{CTx}_{\mathtt{SIS}}$ holds zero coin generation property. □

## X. AGGREGABLE CONFIDENTIAL TRANSACTIONS WITH IDEAL LATTICES

As we saw in $\mathrm{CTx}_{\mathtt{SIS}}$, we have to repeat the process multiple times to reduce the soundness error. Therefore, the overall size of the coin bundles and carry proofs rapidly increases with the number of challenges.

As a solution, we introduce a new aggregate confidential transaction protocol based on the $\mathtt{MSIS}$ problem. This protocol uses a single challenge polynomial chosen from a large space and does not need to repeat the same process multiple times. We choose challenge polynomials from $\mathcal{C}^N_{\beta,1}$ with exactly $\beta$ number of $\pm1$ and $N-\beta$ number of zeros. Therefore, there are $\binom{N}{\beta}\cdot 2^\beta$ possible challenges. We can achieve a negligible soundness error by setting $\binom{N}{\beta}\cdot 2^\beta \geq 2^{2\lambda}$ as shown in Step 17 of Figure 10. Other related protocols are explained in Figure 11, 13, and 15.

This protocol also commits the binary form of the coin value in confidential coin bundles. Since coin bundles require range proofs, first, we explain a method to show a bit is 0 or 1 without revealing it. Later we expand this method for $L$ number of bits. Assume a prover $\mathcal{P}$ wants to hide the

**Protocol 10:** MSIS Confidential Coin Bundles

1: $\mathsf{rot}(pp_{(\lambda,L)}, val, i \in [0,L])$: $\vec{b} = \vec{0}$; $b_i = val$; return $b$

2: $\mathsf{COIN.gen}(pp_{(\lambda,L)}, v)$:
3: $\vec{b} = \mathsf{bin}(v)$ such that $\sum_{i=0}^{L-1} 2^i b_i = v$
4: $\vec{r}_0 \xleftarrow{\$} [-\tau,\tau]^{(m-3)\times N}$ // Secret mask key
5: $\vec{s} = [\vec{b}, \vec{0}, \vec{0}, \vec{r}_0] \in \mathcal{R}_q^m$
6: $\vec{u} = \vec{H}\vec{s} \in \mathcal{R}_q^n$ // Commit the coin value and its mask
7: // Lazy Sampling
8: $[\vec{a}_i \xleftarrow{\$} [-(\alpha-(1-b_i)), (\alpha-(1-b_i))]^N]_{i=0}^{L-1}$
9: // Range-proofs' masks
10: $\vec{r}_1 \xleftarrow{\$} [-\tau_1,\tau_1]^{(m-3)\times N}$
11: $\vec{r}_2 \xleftarrow{\$} [-\tau_2,\tau_2]^{(m-3)\times N}$
12: $\vec{s}_1 = [\vec{0}, \sum_{i=0}^{L-1} \vec{a}_i \mathsf{rot}(2b_i-1,i), \vec{0}, \vec{r}_1] \in \mathcal{R}^m$
13: $\vec{t}_1 = \vec{H}\vec{s}_1 \in \mathcal{R}_q^n$
14: $\vec{x}_1 = \mathsf{hash}(\vec{u}, \vec{t}_1) \in \mathcal{C}_{\beta,1}^N$ // Challenge 1
15: $\vec{s}_2 = [\vec{x}_1 \sum_{i=0}^{L-1} \vec{a}_i, \sum_{i=0}^{L-1} \vec{a}_i\vec{a}_i, \vec{0}, \vec{r}_2] \in \mathcal{R}^m$
16: **if** $\|\vec{s}_2\| > \gamma$: go to Step 4
17: $\vec{t}_2 = \vec{H}\vec{s}_2 \in \mathcal{R}_q^n$
18: $\vec{x}_2 = \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{t}_2) \in \mathcal{C}_{\beta,1}^N$ // Challenge 2
19: $[\vec{z}_i = \vec{x}_2 \mathsf{rot}(b_i,i) + \vec{a}_i]_{i=0}^{L-1} \in \mathcal{R}^L$
20: $\vec{r} = \vec{x}_2(\vec{x}_1\vec{r}_0 + \vec{r}_1) + \vec{r}_2 \in \mathcal{R}^{m-3}$
21: **if** $\|\vec{z}\| > (\alpha-1) \vee \|\vec{r}\| > (\tau_2 - \beta^2\tau - \beta\tau_1)$:
22: go to Step 4
23: **if** $\|\sum_{i=0}^{L-1}(\vec{z}_i - \vec{x}_2\mathsf{rot}(1,i))\| > \gamma$: go to Step 4
24: return $\mathtt{mask} = (\vec{r}_0)$, $\mathtt{coin} = (\vec{z}, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_2)$

25: $\mathsf{COIN.open}(pp_{(\lambda,L)}, v, \mathtt{mask}, \mathtt{coin})$ :
26: $(\vec{z}, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_2) = \mathtt{coin}$
27: $\vec{s} = [\mathsf{bin}(v), \vec{0}, \vec{0}, \mathtt{mask}] \in \mathcal{R}^m$
28: return $\|\vec{s}\| \overset{?}{\leq} \alpha \bigwedge \vec{u} \overset{?}{=} \vec{H}\vec{s} \in \mathcal{R}_q^n$

29: $\mathsf{COIN.ver}(pp_{(\lambda,L)}, \mathtt{coin})$ : $(\vec{z}, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_2) = \mathtt{coin}$
30: $\vec{x}_1 = \mathsf{hash}(\vec{u}, \vec{t}_1)$
31: $\vec{s} = [\vec{x}_1 \sum_{i=0}^{L-1} \vec{z}_i,\ \sum_{i=0}^{L-1} \vec{z}_i(\vec{z}_i - \vec{x}_2\mathsf{rot}(1,i)),\ \vec{0},\ \vec{r}] \in \mathcal{R}^m$
32: $\vec{t}_2 = \vec{H}\vec{s} - \vec{x}_2(\vec{x}_1\vec{u} + \vec{t}_1) \in \mathcal{R}_q^n$ // Recreate $\vec{t}_2$
33: return $\vec{x}_2 \overset{?}{=} \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{t}_2) \bigwedge \|\vec{s}\| \leq \gamma$

We commit an additional zero polynomial to make the protocol compatible with the carry proofs.

**FIGURE 11.** Confidential coin bundles based on MSIS problem.

$i$th bit $b_i$ but wants to prove to a verifier $\mathcal{V}$ that $b_i$ is definitely 0 or 1. Here, we commit the $i$th bit as a polynomial of $\vec{b} = [\mathbf{0}^{i-1}, b_i, \mathbf{0}^{N-i}]$ which we denote as $\mathsf{rot}(b_i, i)$ through out the paper (see Figure 11).

The interactive range proof protocol works as follows.

1) First, $\mathcal{P}$ chooses a random small polynomial $\vec{a}$.
2) $\mathcal{P}$ computes these commitments;

$$\vec{u} = \mathtt{commit}([\vec{b}, \vec{0}],\ \vec{r}) \text{ and}$$
$$\vec{t}_1 = \mathtt{commit}([\vec{0}, \vec{a}\mathsf{rot}(2b_i-1,i)],\ \vec{r}_1)$$

with some masks $\vec{r}, \vec{r}_1$. Then $\mathcal{P}$ sends $(\vec{u}, \vec{t}_1)$ to $\mathcal{V}$.

3) $\mathcal{V}$ chooses a random challenge $\vec{x}_1$ and sends to $\mathcal{P}$.
4) $\mathcal{P}$ computes $\vec{t}_2 = \mathtt{commit}([\vec{x}_1\vec{a}, \vec{a}\vec{a}],\ \vec{r})$ with some mask $\vec{r}_2$. Then $\mathcal{P}$ sends the commitment $\vec{t}_2$ to $\mathcal{V}$.
5) $\mathcal{V}$ chooses a random challenge $\vec{x}_2$ and sends to $\mathcal{P}$.
6) $\mathcal{P}$ sends $\vec{z} = \vec{b}\vec{x}_2 + \vec{a}$, $\vec{R} = \vec{x}_2(\vec{x}_1\vec{r} + \vec{r}_1) + \vec{r}_2$ to $\mathcal{V}$.
7) $\mathcal{V}$ accepts if

$$\mathtt{commit}([\vec{x}_1\vec{z},\ \vec{z}(\vec{z} - \vec{x}_2\mathsf{rot}(1,i))],\ \vec{R}) = \vec{x}_2(\vec{x}_1\vec{u} + \vec{t}_1) + \vec{t}_2.$$

If there is $\vec{z} = \vec{b}\vec{x} + \vec{a}$ for some $\vec{a}$ and challenge $\vec{x}$, then

$$\vec{z}(\vec{z} - \vec{x}\mathsf{rot}(1,i)) = \vec{x}^2 \underbrace{\vec{b}(\vec{b} - \mathsf{rot}(1,i))}_{0} + \vec{x}\vec{a}\mathsf{rot}(2b-1,i) + \vec{a}\vec{a}$$

when $b$ is 0 or 1. Therefore, $\mathcal{P}$ can prove that $b_i$ is indeed 0 or 1 without revealing $b_i$ directly.

We commit the binary form $\vec{b} = \mathsf{bin}(v)$ of the coin amount $v$ when $v = \sum_{i=0}^{L-1} 2^i b_i$. First we choose random $\vec{a}_i$ and create $\vec{z}_i = \vec{x}_2\mathsf{rot}(b_i,i) + \vec{a}_i$ for each $b_i$. Note that the secure range for $\vec{z}$ is $[-(\alpha-1), \alpha-1]$ to prevent information leakage. Therefore, we perform rejection sampling for $\vec{z}$ and start over if $\vec{z}$ is out of the range. To increase the acceptance rate, we use "lazy sampling" where,

- $\vec{a}_i$ is chosen from $[-(\alpha-1), \alpha-1]^N$ if $b_i$ is 0, and
- $\vec{a}_i$ is chosen from $[-\alpha, \alpha]^N$ if $b_i$ is 1.

MSIS confidential transactions contain carry proofs since confidential coin bundles commit binary forms of the coin values. These carry proofs commit $[c_{1,j} - c_{0,j} - 2(c_{1,j+1} - c_{0,j+1})]_{j=0}^{L}$ when the input carry vector and output carry vector are $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, respectively. Therefore, we can check the summation of inputs and outputs following Equation (1). Not only that, carry proofs need range proofs to make sure that carries are binary vectors. For that, we create a single range proof for both input carries and output carries, which we call "an aggregated range proof". We illustrate the structure of carry proofs in Figure 13.

The theft resistance of a transaction comes from a multi-signature. As we explained in Section III, we use the masks of input commitments, output commitments, input carry commitment, and output commitment (if available) to compute the secret key of the whole transaction. Since senders and receivers want to hide their secret keys from each other, they create partial signatures with rejection sampling. Then they aggregate all partial signatures to create the multi-signature. In that way, they can hide their masking keys and yet create a valid multi-signature.

We state the complete non-interactive confidential coin bundle protocol in Figure 11 and non-interactive confidential transaction protocol in Figure 14 and 15. Also, we illustrate the correctness of the protocol in Figure 12 and Figure 13. Since our protocols are non-interactive, or the challenges are computed from a hash function, we can recreate them. Using this property, we do not include $\vec{t}_2$ vectors of both coin bundles and carry proofs. Instead, we include $\vec{x}_2$ and recompute $\vec{t}_2$. As a result, we can save significant amount of space, e.g., a vector of $\vec{t}_2$ is 9.6 KB while $\vec{x}_2$ is 32 bytes in our MSIS-CTx implementation. The same technique is used for signatures as well where we include $\vec{x}_0$, not $\vec{y}$.

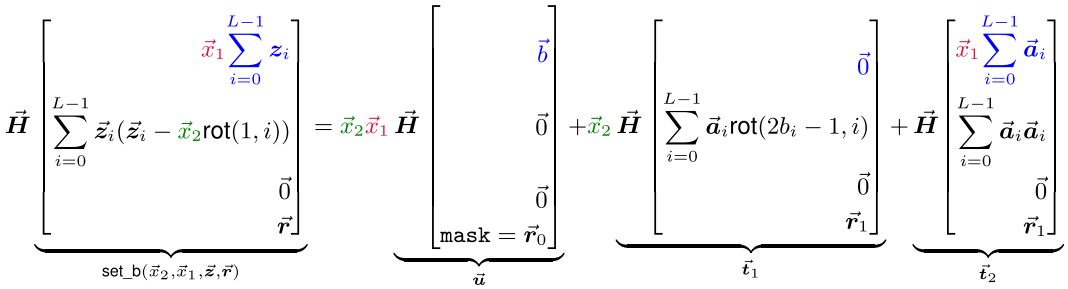
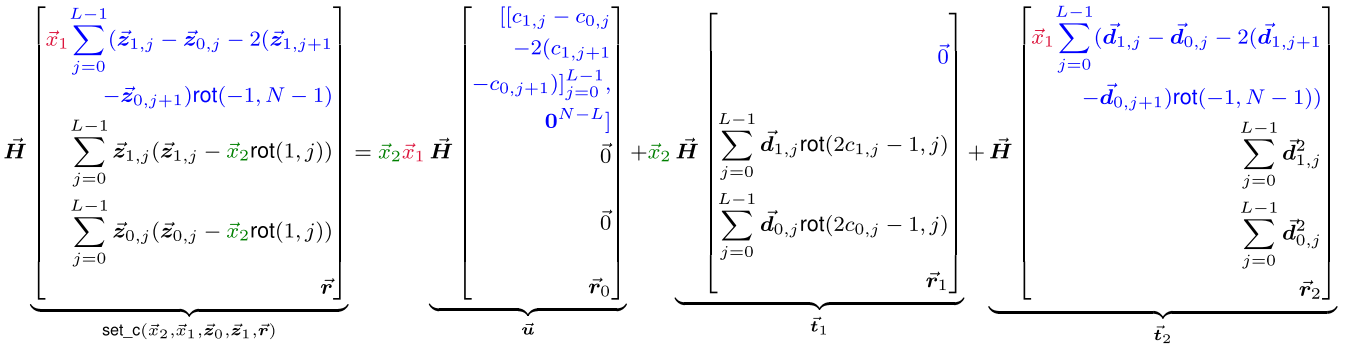


**FIGURE 12.** Coin structure of coin bundle $(\vec{z}, \vec{r}, \vec{u}, \vec{t}_1\vec{u}, \vec{t}_1, , \vec{x}_2 = \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{t}_2))$ with its $\vec{t}_2$ and $\vec{x}_1 = \mathsf{hash}(\vec{u}, \vec{t}_1)$.



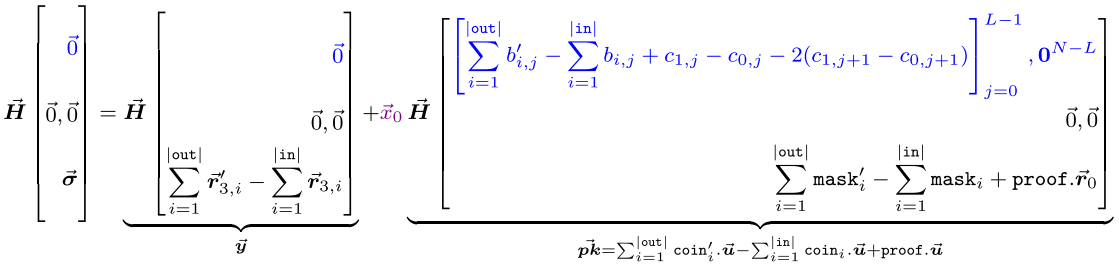


**FIGURE 13.** Carry proof structure of $(\vec{z}_0, \vec{z}_1, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_0 = \mathsf{hash}(\vec{pk}, \vec{y}), \vec{x}_2 = \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{t}_2, \sigma)), \vec{\sigma}, \vec{pk})$ with $(\vec{t}_2, \vec{y}, \vec{x}_1 = \mathsf{hash}(\vec{u}, \vec{t}_1, \sigma))$.

## XI. SECURITY PROOFS FOR MSIS PROBLEM BASED TRANSACTIONS

We start with proving the confidential coin bundles' security. Then, we prove the security of confidential transactions.

### A. SECURITY OF CONFIDENTIAL COIN BUNDLES

**Theorem 8:** $\mathrm{COIN}_{\mathtt{MSIS}}$ *is complete, binding, hiding, and knowledge sound if* $\mathrm{COM}_{\mathtt{MSIS}}$ *is binding and hiding, and* $\mathtt{MSIS}_{n,m,q,\gamma,N}$ *is hard.*

The completeness of coin bundles states that honestly generated coin bundles are always valid. In other words, for coin amounts in $[0, 2^L)$, the generation function should output a coin bundle that will be accepted by the verification function. We conclude that COIN is complete since the verification is equivalent to the generation as shown in Figure 12.

**Lemma 7:** $\mathrm{COIN}_{\mathtt{MSIS}}$ *is binding if* $\mathrm{COM}_{\mathtt{MSIS}}$ *is binding.*

*Proof:* If there is $\mathcal{A}$ that breaks the binding property of $\mathrm{COIN}_{\mathtt{MSIS}}$, we can simulate $\mathcal{A}$ to break $\mathrm{COM}_{\mathtt{MSIS}}$'s binding property. Hence, we conclude that $\mathrm{COIN}_{\mathtt{MSIS}}$ is binding if $\mathrm{COM}_{\mathtt{MSIS}}$ is binding. □

**Lemma 8:** $\mathrm{COIN}_{\mathtt{MSIS}}$ *is computationally hiding if* $\mathrm{COM}_{\mathtt{MSIS}}$ *is computationally hiding.*

*Proof:* We show (1) $\vec{z}$ statistically hides the coin amount, and (2) $(\vec{u}, \vec{t}_1, \vec{t}_2)$ computationally hide their committed values.

Recall that each $\|\vec{z}_i\|$ is in $[-(\alpha - 1), (\alpha - 1)]^N$ and $\|\vec{x}_2\| = 1$. As a result, there exists a valid $\vec{a}_i \in [-\alpha, \alpha]^N$ for any $b \in [0, 1]$ because,

- when $b = 0$: $\vec{a}_i = \vec{z}_i \in [-\alpha, \alpha]^N$
- when $b = 1$: $\vec{a}_i = \vec{z}_i - \vec{x}_2\mathsf{rot}(1, i) \in [-\alpha, \alpha]^N$.

Therefore, we claim $\vec{z}$ statistically hides the coin amount.

Recall that the smallest random mask $\tau$ satisfies the condition; $(m - 3)N \log(2\tau) > 6\lambda$. If $\mathcal{A}$ breaks the hiding property then we can simulate $\mathcal{A}$ to break the hiding property of $\mathrm{COM}_{\mathtt{MSIS}}$ when $d = 3$. Therefore, we claim $\mathrm{COIN}_{\mathtt{MSIS}}$ is computationally hiding if $\mathrm{COM}_{\mathtt{MSIS}}$ is computationally hiding. □

**Lemma 9:** *Breaking knowledge soundness of* $\mathrm{COIN}_{\mathtt{MSIS}}$ *is computationally infeasible if* $\mathrm{COM}_{\mathtt{MSIS}}$ *is binding, and* $\mathtt{MSIS}_{pp_\lambda}$ *is hard for* $pp_\lambda = (n, m, q, \gamma, N)$.

**Protocol 11:** Proofs of MSIS Confidential Transactions

This protocol shows how to compute and verify transactions when both input carries and output carries should be computed. When there is only one input or only one output, we do not compute their range proofs and commit zero polynomials instead.

// $|\mathtt{in}|, |\mathtt{out}|$ ($\leq 2$) are the number of inputs and outputs.
// Senders and receivers do not share their $\mathtt{mask}$ or $\mathtt{mask}'$ with other senders or receivers in any step.
$\underline{\mathrm{CTX}_{\mathtt{MSIS}}.\mathsf{proof}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|, [v^{(i)}, \mathtt{mask}_i]_{i=1}^{|\mathtt{in}|}, [v'^{(i)}]_{i=1}^{|\mathtt{out}|}):}$
$[b_i = \mathsf{bin}(v^{(i)})]_{i=1}^{|\mathtt{in}|}, [b'_i = \mathsf{bin}(v'^{(i)})]_{i=1}^{|\mathtt{out}|}$
// Carries when $\div$ is the integer division, e.g., $3 \div 2 = 1$
$c_0 = \left[0, [c_{0,j+1} = \left(\sum_{i=1}^{|\mathtt{in}|} b_{i,j} + c_{0,j}\right) \div 2]_{j=0}^{L-2}, 0\right]$
$c_1 = \left[0, [c_{1,j+1} = \left(\sum_{i=1}^{|\mathtt{out}|} b'_{i,j} + c_{1,j}\right) \div 2]_{j=0}^{L-2}, 0\right]$
// All receivers compute random masks for carries
// Lazy Sampling
$[\vec{d}_{0,i} \xleftarrow{\$} [-(\alpha - (1 - c_{0,i})), (\alpha - (1 - c_{0,i}))]^N]_{i=1}^{L-1}$
Set $\vec{d}_{0,0} = \vec{0}$ and $\vec{d}_{0,L} = \vec{0}$
$[\vec{d}_{1,i} \xleftarrow{\$} [-(\alpha - (1 - c_{1,i})), (\alpha - (1 - c_{1,i}))^N]_{i=1}^{L-1}$
Set $\vec{d}_{1,0} = \vec{0}$ and $\vec{d}_{1,L} = \vec{0}$
$\hat{\vec{c}}_0 = \sum_{j=0}^{L-1} \vec{d}_{0,j}\mathsf{rot}(2c_{0,j} - 1, j) \in \mathcal{R}$
$\hat{\vec{c}}_1 = \sum_{j=0}^{L-1} \vec{d}_{1,j}\mathsf{rot}(2c_{1,j} - 1, j) \in \mathcal{R}$
$\hat{\vec{c}}' = [(c_{1,j} - c_{0,j}) - 2(c_{1,j+1} - c_{0,j+1})]_{j=0}^{L-1}, \mathbf{0}^{N-L}] \in \mathcal{R}$
$\hat{\vec{d}}'_0 = \sum_{j=0}^{L-1} (\vec{d}_{0,j} - 2\vec{d}_{0,j+1}\mathsf{rot}(-1, N-1)) \in \mathcal{R}$
$\hat{\vec{d}}'_1 = \sum_{j=0}^{L-1} (\vec{d}_{1,j} - 2\vec{d}_{1,j+1}\mathsf{rot}(-1, N-1)) \in \mathcal{R}$
// Pick random masks
$\vec{r}_0 \xleftarrow{\$} [-\tau, \tau]^{(m-3)\times N}$
$\vec{r}_1 \xleftarrow{\$} [-\tau_1, \tau_1]^{(m-3)\times N}$
$\vec{r}_2 \xleftarrow{\$} [-\tau_2, \tau_2]^{(m-3)\times N}$
$\vec{u} = \vec{H}[\hat{\vec{c}}', \vec{0}, \vec{0}, \vec{r}_0] \in \mathcal{R}_q^n$ // Commit carries
Each sender $i \in [1, |\mathtt{in}|]$ secretly computes:
  ▷ $\vec{r}_{3,i} \xleftarrow{\$} [-\tau_3, \tau_3]^{(m-3)\times N}$
  ▷ $\vec{y}_i = \vec{H}[\vec{0}, \vec{0}, \vec{0}, \vec{r}_{3,i}] \in \mathcal{R}_q^n$ and share $\vec{y}_i$
Each receiver $i \in [1, |\mathtt{out}|]$ secretly computes:
  ▷ $\vec{r}'_{3,i} \xleftarrow{\$} [-\tau_3, \tau_3]^{(m-3)\times N}$
  ▷ $(\mathtt{mask}'_i, \mathtt{coin}'_i) = \mathrm{COIN}.\mathsf{gen}(pp_{(\lambda,L)}, v'_i)$
  ▷ $\vec{y}'_i = \vec{H}[\vec{0}, \vec{0}, \vec{0}, \vec{r}'_{3,i}] \in \mathcal{R}_q^n$ and share $\vec{y}'_i$ and $\mathtt{coin}'_i$
// All receivers compute public key
$\vec{pk} = \vec{u} + \sum_{i=1}^{|\mathtt{out}|} \mathtt{coin}'_i.\vec{u} - \sum_{i=1}^{|\mathtt{in}|} \mathtt{coin}_i.\vec{u} \in \mathcal{R}_q^n$
$\vec{y} = \sum_{i=1}^{|\mathtt{out}|} \vec{y}'_i - \sum_{i=1}^{|\mathtt{in}|} \vec{y}_i \in \mathcal{R}_q^n$
$\vec{x}_0 = \mathsf{hash}(\vec{pk}, \vec{y}) \in \mathcal{C}_{\beta,1}^N$ // Signature challenge
Each sender $i \in [1, |\mathtt{in}|]$ secretly computes:
  ▷ $\vec{\sigma}_i = \vec{r}_{3,i} + \vec{x}_0\mathtt{mask}'_i \in \mathcal{R}^{(m-3)}$
  ▷ **if** $\|\vec{\sigma}_i\| > (\tau_3 - \beta\tau)$:
    go to Step
Each receiver $i \in [1, |\mathtt{out}|]$ secretly computes:
  ▷ $\vec{\sigma}'_i = \vec{r}'_{3,i} + \vec{x}_0\mathtt{mask}'_i \in \mathcal{R}^{(m-3)}$
  ▷ **if** $\|\vec{\sigma}'_i\| > (\tau_3 - \beta\tau)$:
    go to Step 8
// All receivers compute the signature
$\vec{\sigma} = \vec{x}_0(\vec{r}_0 + \sum_{i=1}^{|\mathtt{out}|} \vec{\sigma}'_i - \sum_{i=1}^{|\mathtt{in}|} \vec{\sigma}_i) \in \mathcal{R}^{(m-3)}$
**if** $\|\vec{\sigma}\| > ((|\mathtt{out}| + |\mathtt{in}|)\tau_3 - (|\mathtt{out}| + |\mathtt{in}| + 1)\beta\tau)$:
  go to Step 8
// All receivers compute range proofs for carries
$\vec{t}_{0,1,i} = \vec{H}[\vec{0}, \hat{\vec{c}}_1, \hat{\vec{c}}_0, \vec{r}_1] \in \mathcal{R}_q^n$
$\vec{x}_1 = \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{\sigma}) \in \mathcal{C}_{\beta,1}^N$ // Proof challenge 1
$\vec{s}_2 = [\vec{x}_1(\hat{\vec{d}}'_1 - \hat{\vec{d}}'_0), \sum_{j=0}^{L-1} \vec{d}_{1,j}^2, \sum_{j=0}^{L-1} \vec{d}_{0,j}^2, \vec{r}_2] \in \mathcal{R}^m$
$\vec{t}_{2,i} = \vec{H}\vec{s}_2 \in \mathcal{R}_q^n$
$\vec{x}_2 = \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{t}_2, \vec{\sigma}) \in \mathcal{C}_{\beta,1}^N$ // Proof challenge 2
$\vec{z}_0 = [\vec{x}_2\mathsf{rot}(c_{0,j}, j) + \vec{d}_{0,j}]_{j=1}^{L-1} \in \mathcal{R}^{L-1}$
$\vec{z}_1 = [\vec{x}_2\mathsf{rot}(c_{1,j}, j) + \vec{d}_{1,j}]_{j=1}^{L-1} \in \mathcal{R}^{L-1}$
**if** $\|\vec{z}_0\| > (\alpha - 1) \vee \|\vec{z}_1\| > (\alpha - 1)$:
  go to Step 8
**if** $\|\sum_{j=0}^{L-1} \vec{z}_{1,i}(\vec{z}_{1,i} - \vec{x}_2\mathsf{rot}(1, i))\| > \gamma$
$\vee \|\sum_{i=0}^{L-1} \vec{z}_{0,i}(\vec{z}_{0,i} - \vec{x}_2\mathsf{rot}(1, i))\| > \gamma$:
  go to Step 8
$\vec{r} = \vec{x}_2(\vec{x}_1\vec{r}_0 + \vec{r}_1) + \vec{r}_2 \in \mathcal{R}^{m-3}$
**if** $\|\vec{r}\| > (\tau_2 - \beta^2\tau - \beta\tau_1)$ : go to Step 8
return $\mathtt{proof} = (\vec{z}_0, \vec{z}_1, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_0, \vec{x}_2, \vec{\sigma}, \vec{pk})$,
  $\mathtt{out} = [\mathtt{coin}']_{i=1}^{|\mathtt{out}|}$
// **if** $|\mathtt{in}| = 1 \wedge |\mathtt{out}| = 2$ :
  $\mathtt{proof} = (\vec{z}_1, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_0, \vec{x}_2, \vec{\sigma}, \vec{pk})$
// **if** $|\mathtt{in}| = 2 \wedge |\mathtt{out}| = 1$ :
  $\mathtt{proof} = (\vec{z}_0, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_0, \vec{x}_2, \vec{\sigma}, \vec{pk})$
// **else**: $\mathtt{proof} = (\vec{x}_0, \vec{\sigma}, \vec{pk})$

$\underline{\mathrm{CTX}_{\mathtt{MSIS}}.\mathsf{proof\_ver}(pp_{(\lambda,L)}, \mathtt{proof}) :}$
// $(*)$ denotes optional variables
$(\vec{z}_0^{(*)}, \vec{z}_1^{(*)}, \vec{r}^{(*)}, \vec{u}^{(*)}, \vec{t}_1^{(*)}, \vec{x}_0, \vec{x}_2^{(*)}, \vec{\sigma}, \vec{pk}) := \mathtt{proof}$
**if** $\|\vec{\sigma}\| > \tau_3$: return 0
**if** $|\mathtt{in}| \stackrel{?}{=} 2 \vee |\mathtt{out}| \stackrel{?}{=} 2$:
  **if** $\|\vec{r}\| > \tau_2$:
  Set $\hat{z}' = \vec{0}$, $\hat{z}_0 = \vec{0}$, and $\hat{z}_1 = \vec{0}$
  $\vec{x}_1 = \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{\sigma}) \in \mathcal{C}_{\beta,1}^N$
  **if** $|\mathtt{in}| \stackrel{?}{=} 2$:
    **if** $\|\vec{z}_0\| > \alpha$: return 0
    Set $\vec{z}_{0,0}$ and $\vec{z}_{0,L}$ to $\vec{0}$
    $\hat{z}' = \vec{x}_1 \sum_{i=0}^{L-1} (\vec{z}_{0,j} - 2\vec{z}_{0,j+1}\mathsf{rot}(-1, N-1)) \in \mathcal{R}$
    $\hat{z}_0 = \sum_{i=0}^{L-1} \vec{z}_{0,i}(\vec{z}_{0,i} - \vec{x}_2\mathsf{rot}(1, i)) \in \mathcal{R}$
  **if** $|\mathtt{out}| \stackrel{?}{=} 2$:
    **if** $\|\vec{z}_1\| > \alpha$: return 0
    Set $\vec{z}_{1,0}$ and $\vec{z}_{1,L}$ to $\vec{0}$
    $\hat{z}' = \vec{x}_1 \sum_{i=0}^{L-1} (\vec{z}_{1,j} - 2\vec{z}_{1,j+1}\mathsf{rot}(-1, N-1)) - \hat{z}'$
    $\hat{z}_1 = \sum_{i=0}^{L-1} \vec{z}_{1,i}(\vec{z}_{1,i} - \vec{x}_2\mathsf{rot}(1, i)) \in \mathcal{R}$
  $\vec{s} = [\hat{z}', \hat{z}_1, \hat{z}_0, \vec{r}] \in \mathcal{R}^m$
  **if** $\|\vec{s}\| > \gamma$: return 0
  $\vec{t}_2 = \vec{H}\vec{s} - \vec{x}_2(\vec{x}_1\vec{u} + \vec{t}_1) \in \mathcal{R}_q^n$ // Recompute $\vec{t}_2$
  **if** $\vec{x}_2 \neq \mathsf{hash}(\vec{u}, \vec{t}_1, \vec{t}_2, \vec{\sigma})$: return 0
$\vec{y} = \vec{H}\vec{\sigma} - \vec{x}_0\vec{pk} \in \mathcal{R}_q^n$ // Recompute $\vec{y}$
return $\vec{x}_0 \stackrel{?}{=} \mathsf{hash}(\vec{pk}, \vec{y})$

**FIGURE 14.** Proofs for MSIS problem based aggregate confidential transactions.

**Protocol 12:** MSIS Confidential Transactions

1: // $|\mathtt{in}|, |\mathtt{out}|$ ($\leq 2$) are the number of inputs and outputs.
2: // Senders and receivers do not share their $\mathtt{mask}$ or $\mathtt{mask}'$ with other senders or receivers in any step.
3: $\mathrm{CTx}_{\mathtt{MSIS}}.\mathsf{gen}(pp_{(\lambda,L)}, [v_i, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'_i]_{i=1}^{|\mathtt{out}|})$:
4: $\mathtt{proof}, \mathtt{out} = \mathrm{CTx}.\mathsf{proof}(pp_{(\lambda,L)}, |\mathtt{in}|, |\mathtt{out}|,$
5: $\qquad [v^{(i)}, \mathtt{mask}_i, \mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, [v'^{(i)}]_{i=1}^{|\mathtt{out}|})$
6: return $\mathtt{tx} = (\mathtt{in} = [\mathtt{coin}_i]_{i=1}^{|\mathtt{in}|}, \mathtt{out}, \mathtt{proof})$

7: $\mathrm{CTx}_{\mathtt{MSIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{tx})$ :
8: $(\vec{z}_0, \vec{z}_1, \vec{r}, \vec{u}, \vec{t}_1, \vec{x}_0, \vec{x}_2, \vec{\sigma}, \vec{pk}) := \mathtt{tx.proof}$
9: // Verify input and output coin bundles
10: **if** $[\mathtt{coin} \in \mathtt{tx.in} : \neg\mathrm{COIN}_{\mathtt{MSIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{coin})] \bigvee$
11: $\quad [\mathtt{coin}' \in \mathtt{tx.out} : \neg\mathrm{COIN}_{\mathtt{MSIS}}.\mathsf{ver}(pp_{(\lambda,L)}, \mathtt{coin}')]$ :
12: $\quad$ return 0
13: // Verify the proof of the transaction
14: **if** $\neg\mathrm{CTx}_{\mathtt{MSIS}}.\mathsf{proof\_ver}(pp_{(\lambda,L)}, \mathtt{tx.proof})$ : return 0
15: return $\vec{pk} = \vec{u} + \sum_{\mathtt{coin}' \in \mathtt{tx.out}} \mathtt{coin}'.\vec{u} - \sum_{\mathtt{coin} \in \mathtt{tx.in}} \mathtt{coin}.\vec{u}$

**FIGURE 15.** `MSIS` problem-based aggregate confidential transactions.

*Proof:* Assume algorithm $\mathcal{A}$ wins $\mathsf{Game}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}}$ with a non-negligible probability when $\mathrm{COM}_{\mathtt{MSIS}}$ is binding. Then $\mathcal{A}$ creates a coin bundle for an invalid $\vec{b}$ such that

$$\begin{aligned}\vec{H}[\vec{x}_1 \sum_{i=0}^{L-1} \boldsymbol{z}_i, \sum_{i=0}^{L-1} \vec{z}_i(\vec{z}_i - \vec{x}_2\mathsf{rot}(1,i)), \vec{r}] \\ = \underbrace{\vec{x}_2^2}_{\neq\vec{0}} \underbrace{\vec{H}\,[\vec{0}, \underbrace{\sum_{i=0}^{L-1} \mathsf{rot}(b_i(b_i-1), i)}_{\vec{s}\neq\vec{0}^m}, \vec{0}^{m-2}]}_{\vec{h}=\vec{0}^n} \\ + \vec{x}_2\vec{x}_1\vec{u} + \vec{x}_2\vec{t}_1 + \vec{t}_2.\end{aligned}$$

First, we prove that $\vec{x}_2^2$ cannot be a zero polynomial because the first coefficient of $\vec{x}_2^2$ is always equal to $\beta$. Therefore, $\vec{h}$ should be a zero polynomial. Also, $\vec{s}$ is not equal to $\vec{0}^m$ since $\vec{b}$ is not a valid binary polynomial. Therefore, $\vec{s}$ is a solution to `MSIS` problem. We reduce $\mathcal{A}$'s probability of winning $\mathsf{Game}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}}$ to,

$$\mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN},pp_{(\lambda,L)}} \leq \mathsf{Adv}^{\mathtt{MSIS},\mathcal{A}}_{pp_\lambda} + \mathsf{Adv}^{\mathsf{BND},\mathcal{A}}_{\mathrm{COM},pp_{(\lambda,3)}}$$

after *one execution*. Finally, we conclude Lemma 9 is valid or $\mathrm{COIN}_{\mathtt{MSIS}}$ is knowledge sound. □

### B. SECURITY PROOFS FOR CONFIDENTIAL TRANSACTIONS

**Theorem 9:** $\mathrm{CTx}_{\mathtt{MSIS}}$ *is complete, theft-resistant, hiding, and provides zero coin generation property if* $\mathrm{COIN}_{\mathtt{MSIS}}$ *is complete, binding, hiding, and knowledge sound,* $\mathrm{SIG}_{\mathtt{MSIS}}$ *is complete and EUF-CMA, and* $\mathtt{MSIS}_{n,m,q,\gamma,N}$ *is hard.*

First we show that CTx is complete by proving that the carry proofs' verification and coin summation verification are correct. We illustrate the correctness of CTx in Figure 13.

**Lemma 10:** $\mathrm{CTx}_{\mathtt{MSIS}}$ *of* $(n, m, q, N, \gamma)$ *is resistant to theft if* $\mathrm{SIG}_{\mathtt{MSIS}}$ *of* $(n, m-3, q, N, \gamma)$ *is EUF-CMA.*

*Proof:* Suppose $\mathcal{A}$ spends the coin bundle after creating a valid transaction. Then, we can simulate $\mathcal{A}$ to break EUF-CMA of $\mathrm{SIG}_{\mathtt{MSIS}}$ since $\mathcal{A}$ creates the signature without knowing the secret mask. Therefore, we claim that CTx is theft resistant when $\mathrm{SIG}_{\mathtt{MSIS}}$ is EUF-CMA. □

**Lemma 11:** $\mathrm{CTx}_{\mathtt{MSIS}}$ *is computationally hiding when* $\mathrm{COM}_{\mathtt{MSIS}}$ *is computationally hiding.*

*Proof:* Similar to $\mathrm{COIN}_{\mathtt{MSIS}}$, we claim that carry proofs are computationally hiding since (1) the smallest random mask $\tau$ satisfies the condition of $(m-3)N\log(2\tau) \geq 6\lambda$, and (2) any pair of $(\vec{z}_0, \vec{z}_1)$ statistically hides the input carries and output carries. □

**Lemma 12:** *Breaking* $\mathrm{CTx}_{\mathtt{MSIS}}$*'s zero coin generation property is computationally infeasible if* $\mathrm{COIN}_{\mathtt{MSIS}}$ *is knowledge sound,* $\mathrm{SIG}_{\mathtt{MSIS}}$ *of* $(n, m-3, q, N, \gamma)$ *is EUF-CMA, and* $\mathtt{MSIS}_{pp_\lambda}$ *is hard for* $pp_\lambda = (n, m, q, \gamma, N)$.

*Proof:* Let there be a p.p.t. algorithm $\mathcal{A}$ that breaks the zero coin generation property of $\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx},pp_{(\lambda,L)}}$ when $\mathrm{COIN}_{\mathtt{MSIS}}$ is knowledge sound, and $\mathrm{SIG}_{\mathtt{MSIS}}$ is EUF-CMA. Then $\mathcal{A}$ creates valid carry proofs when total inputs are not equal to total outputs. Assume the extractor $\mathcal{E}$ outputs input coin values $[v_i]_{i=1}^{|\mathtt{in}|}$ and output coin values $[v'_i]_{i=1}^{|\mathtt{out}|}$ such that,

$$\sum_{i=1}^{|\mathtt{in}|} v_i \neq \sum_{i=1}^{|\mathtt{out}|} v'_i.$$

We can replace the extracted coin values with their binary forms and **real** carries; for $j \in [0, L)$,

$$\sum_{i=1}^{\mathtt{in}} b_{i,j} + c_{0,j} - 2c_{0,(j+1)} \neq \sum_{i=1}^{\mathtt{out}} b'_{i,j} + c_{1,j} - 2c_{1,(j+1)}. \tag{9}$$

We can reduce $\mathcal{A}$'s advantage to the following three scenarios.

1) Due to the signature verification in Step 91, $\mathcal{A}$ finds

$$\boldsymbol{H}\underbrace{\left[\begin{bmatrix} \left[\sum_{i=1}^{\mathtt{out}} b'_{i,j} - \sum_{i=1}^{\mathtt{in}} b_{i,j} + c_{1,j} - c_{0,j}\right. \\ \left. -2(c_{1,(j+1)} - 2c_{0,(j+1)})\right]_{j=0}^{L-1} \end{bmatrix}, \mathbf{0}^{N-L}\right]}_{\neq\vec{0}^m \text{ (From Equation (9))}}, \vec{0}^{m-1}\Bigg]$$
$$= \mathbf{0}^n$$

2) $\mathcal{A}$ commits invalid carries $(\boldsymbol{c}'_0, \boldsymbol{c}'_1) \neq (\boldsymbol{c}_0, \boldsymbol{c}_1)$. Here, $(\boldsymbol{c}'_0, \boldsymbol{c}'_1)$ cannot be a pair of binary vectors since there is only one binary vector pair that satisfies Equation 2.

From carry proof verification in Step 89, we know that

$$\underbrace{\vec{x}_2^2 \boldsymbol{H} \underbrace{\begin{bmatrix} \vec{0} \\ \sum_{i=0}^{L-1} \mathsf{rot}\left(c'_{1,i}, i\right)\left(\mathsf{rot}(c'_{1,i}, i) - \mathsf{rot}(1, i)\right) \\ \sum_{i=0}^{L-1} \mathsf{rot}(c'_{0,i}, i)\left(\mathsf{rot}(c'_{0,i}, i) - \mathsf{rot}(1, i)\right) \\ \boldsymbol{\mathfrak{G}}^{m-3} \end{bmatrix}}_{\neq \vec{0}^m} = \vec{0}^n}_{\boldsymbol{\mathfrak{k}} = \vec{0}^n} \tag{10}$$

Since $\vec{x}_2^2$ cannot be a zero polynomial, $\boldsymbol{\mathfrak{k}}$ is $\vec{0}^n$.

3) $\mathcal{A}$ breaks the knowledge soundness of $\mathsf{COIN}_{\mathtt{MSIS}}$.

In both scenarios, $\mathcal{A}$'s advantage is,

$$\mathsf{Adv}^{\mathcal{A},\mathtt{carry}}_{\mathsf{CTx},pp_{(\lambda,L)}} \leq \mathsf{Adv}^{\mathtt{MSIS},\mathcal{A}}_{pp_\lambda} + \mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\mathsf{COIN},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathsf{SIG},pp_\lambda}.$$

Therefore, we claim that $\mathsf{CTx}_{\mathtt{MSIS}}$ holds zero coin generation property when the MSIS problem is hard, and $\mathsf{COIN}_{\mathtt{MSIS}}$ is knowledge sound. □

## XII. TRANSACTION AGGREGATION

This section explains transaction aggregation process for both $\mathsf{CTx}_{\mathtt{SIS}}$ and $\mathsf{CTx}_{\mathtt{MSIS}}$. Then we prove the security of the aggregated cash system.

Once the cash system receives a new transaction, it verifies the transaction. If the transaction is valid, then the inputs are in the unspent coin table. Next, the cash system removes those inputs from the unspent coin table and adds new transaction outputs to the table. Also, the cash system adds the number of inputs, the number of outputs, and the proof to the header table. This aggregation preserves cash systems verification due to Equation (2). We state the transaction aggregation and the verification of the aggregated cash system in Figure 16.

Let ACS be an aggregable cash system with the following properties. Here, we consider two tables (Ucoins, Headers) as a cash system.

**Functionality 6**: Aggregable Cash System

- ACS.**aggregate**($pp_{(\lambda,L)}$, Ucoins, Headers, tx): add tx to (Ucoins, Headers) and return the updated tables.
- ACS.**table_ver**($pp_{(\lambda,L)}$, Ucoins, Headers) : return 1 if the cash system is valid, otherwise 0.

ACS is expected to have these security properties after the aggregation; completeness, theft resistance, and zero-coin generation.

The completeness of the aggregate cash system states that the aggregation of honestly generated transactions always outputs a valid cash system. Let $\mathcal{G}$ be a valid random transaction generator. These transactions can be rewards or spending of existing unspent coin bundles. We capture the completeness as follows.

**Protocol 13:** SIS/MSIS Aggregated Cash System

ACS.aggregate($pp_{(\lambda,L)}$, Ucoins, Headers, tx) :
**if** ¬CTx.ver($pp_{(\lambda,L)}$, tx) : return ⊥
// Check whether the inputs are unspent or not
**if** tx.in ∉ Ucoins : return ⊥
// Remove spent coins and add unspent coins
Ucoins = (Ucoins \ tx.in) ⊎ tx.out
// Add proofs to the transaction table
Headers = Headers ⊎ (|tx.in|, |tx.out|, tx.proof)
return (Ucoins, Headers)

$\mathsf{ACS}_{\mathtt{SIS}}$.table_ver($pp_{(\lambda,L)}, S$, Ucoins, Headers) :
$\boldsymbol{u} = \boldsymbol{pk} = \boldsymbol{0}^n$; $\boldsymbol{s} = [\mathsf{bin}(S), \boldsymbol{0}^{m-L}] \in \mathbb{Z}^m$ // The supply
**for all** (coin$_i$) ∈ Ucoins:
  **if** ¬COIN.ver($pp_{(\lambda,L)}$, coin$_i$): return 0
  $\boldsymbol{u} = \boldsymbol{u} + \mathtt{coin}_i.\boldsymbol{u} \in \mathbb{Z}_q^n$
**for all** proof$_i$ ∈ Headers:
  **if** ¬CTx.proof_ver($pp_{(\lambda,L)}$, proof): return 0
  $\boldsymbol{pk} = \boldsymbol{pk} + \mathtt{proof}_i.\boldsymbol{pk} \in \mathbb{Z}_q^n$
  $\boldsymbol{u} = \boldsymbol{u} + \mathtt{proof}_i.\boldsymbol{u} \in \mathbb{Z}_q^n$
return $\boldsymbol{pk} \stackrel{?}{=} \boldsymbol{u} - \boldsymbol{H}\boldsymbol{s} \in \mathbb{Z}_q^n$ // Equivalent to Equation 2

$\mathsf{ACS}_{\mathtt{MSIS}}$.table_ver($pp_{(\lambda,L)}, S$, Ucoins, Headers) :
$\vec{\boldsymbol{pk}} = \vec{\boldsymbol{u}} = \vec{0}^n \in \mathcal{R}_q^n$; $\vec{\boldsymbol{s}} = [\mathsf{bin}(S), \vec{0}^{(m-1)}] \in \mathcal{R}_q^m$
**for all** (coin$_i$) ∈ Ucoins:
  **if** ¬COIN.ver($pp_{(\lambda,L)}$, coin$_i$): return 0
  $\vec{\boldsymbol{u}} = \vec{\boldsymbol{u}} + \mathtt{coin}_i.\vec{\boldsymbol{u}} \in \mathcal{R}_q^n$
**for all** proof$_i$ ∈ Headers:
  **if** ¬CTx.proof_ver($pp_{(\lambda,L)}$, proof): return 0
  $\vec{\boldsymbol{pk}} = \vec{\boldsymbol{pk}} + \mathtt{proof}_i.\vec{\boldsymbol{pk}} \in \mathcal{R}_q^n$
  $\vec{\boldsymbol{u}} = \vec{\boldsymbol{u}} + \mathtt{proof}_i.\vec{\boldsymbol{u}} \in \mathcal{R}_q^n$
return $\vec{\boldsymbol{pk}} \stackrel{?}{=} \vec{\boldsymbol{u}} - \vec{\boldsymbol{H}}\vec{\boldsymbol{s}} \in \mathcal{R}_q^n$ // Equivalent to Equation 2

**FIGURE 16.** Transaction aggregation.

**Definition 14 (Completeness):** ACS is complete if,

$$Pr\left[\mathsf{random\_cash\_system}(pp_{(\lambda,L)}, \mathtt{t})\right] \geq 1.$$

Theft resistance of an aggregable cash system ensures that unspent coin bundles cannot be spent without their secret masking keys. We allow the adversarial algorithm to generate the cash system for stronger theft resistance. Then we ask the algorithm to transfer a coin bundle coin′ to us by computing a new transaction. In the new transaction, the algorithm chooses all parameters except the masking key mask′ of coin′. After adding the transaction, we ask the algorithm to update the cash system by spending the coin bundle without mask′. If the updated cash system is valid and the coin bundle is not in the unspent coin table, the algorithm wins the game.

This strong theft resistance considers a environment where everyone is dishonest, except the coin bundle's owner. By allowing the algorithm to choose parameters for the transaction, we let the coin sender be dishonest as well,

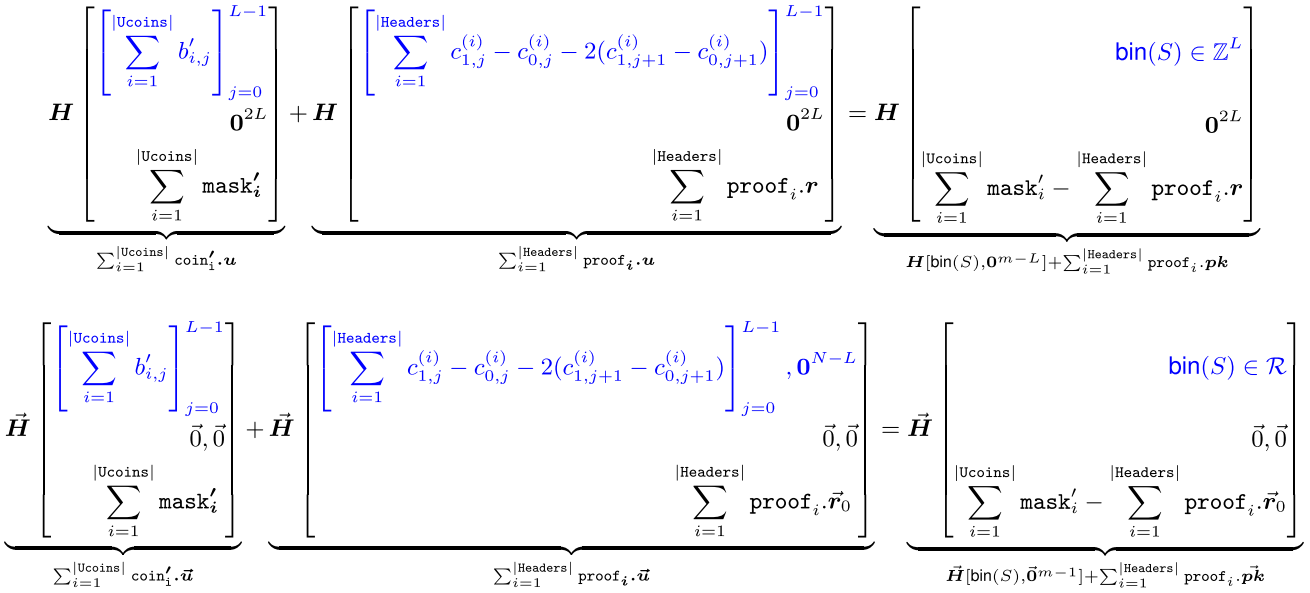


**FIGURE 17.** The structure of aggregate cash systems; $\mathsf{ACS}_{\mathtt{SIS}}$ and $\mathsf{ACS}_{\mathtt{MSIS}}$.

**Functionality 7**: Random Cash System Generation

**random_cash_system**($pp_{(\lambda,L)}$, t):
Get any (Ucoins, Headers) such that
ACS.**table_ver**($pp_{(\lambda,L)}$.Ucoins, Headers) = 1 and
T = |Headers|
**for** $i \in [0, \mathtt{t}]$:
$\mathtt{tx}_i \xleftarrow{\$} \mathcal{G}(pp_{(\lambda,L)})$ s.t. CTx.**ver**($pp_{(\lambda,L)}$, $\mathtt{tx}_i$) = 1
(Ucoins, Headers) = ACS.**aggregate**($pp_{(\lambda,L)}$, Ucoins, Headers, tx)
**if** $\mathtt{Headers}_{\mathtt{T}+i} \neq \mathtt{tx.proof}$: return 0
return ACS.**table_ver**($pp_{(\lambda,L)}$, Ucoins, Headers)
$\bigwedge |\mathtt{Headers}| \stackrel{?}{=} \mathtt{T} + \mathtt{t}$

**Game 12:** Theft Resistance

$\mathsf{Game}^{\mathsf{TR},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}}()$:
(Ucoins, Headers) $\leftarrow \mathcal{A}_{step1}(pp_{(\lambda,L)})$
**if** ACS.**table_ver**($pp_{(\lambda,L)}$, Ucoins, Headers): return $\perp$
// Generate tx with $\mathcal{A}$ s.t. $\mathtt{coin}' \in \mathtt{tx}$ and the masking key $\mathtt{mask}'$ of $\mathtt{coin}'$ is unknown to $\mathcal{A}$.
(Ucoins, Headers) = ACS.**aggregate**($pp_{(\lambda,L)}$, Ucoins, Headers, tx)
(Ucoins′, Headers′) $\leftarrow \mathcal{A}_{step2}(pp_{(\lambda,L)}, \mathtt{Ucoins}, \mathtt{Headers})$

return ACS.**table_ver**($pp_{(\lambda,L)}$, Ucoins′, Headers′)
$\bigwedge \mathtt{Headers} \subset \mathtt{Headers}' \bigwedge \mathtt{coin}' \notin \mathtt{Ucoins}'$

because in the real world we cannot expect anyone else to be honest.

**Definition 15 (Theft-Resistance):** ACS is theft-resistant if

$$\mathsf{Adv}^{\mathsf{TR},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}} = Pr\left[\mathsf{Game}^{\mathsf{TR},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}}()\right] \leq \epsilon(\lambda).$$

**Game 13:** Zero-Coin Generation of Aggregation

$\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}}()$:
(Ucoins, Headers) $\leftarrow \mathcal{A}(pp_{(\lambda,L)}, S \in [0, 2^L))$
return ACS.**table_ver**($pp_{(\lambda,L)}$, Ucoins, Headers) $\bigwedge$
$\sum_{i=1}^{|\mathtt{Ucoins}|} \mathcal{E}(\mathtt{Ucoins}_i) - S \stackrel{?}{\neq} 0$

Zero-coin generation of cash system states that no coins can be generated during the transaction aggregation. Note that the extractor $\mathcal{E}$ outputs the coin amount of the confidential coin bundle. Due to the zero-coin generation, the extracted unspent coins are equal to the original supply.

**Definition 16 (Zero-Coin Generation of Aggregate Cash Systems):** An aggregate Cash System ensures zero-coin generation property if

$$\mathsf{Adv}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}} = Pr\left[\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}}()\right] \leq \epsilon(\lambda).$$

We state aggregable cash systems $\mathrm{ACS}_{\mathtt{SIS}}$ and $\mathrm{ACS}_{\mathtt{MSIS}}$ that are compatible with $\mathrm{CTx}_{\mathtt{SIS}}$ and $\mathrm{CTx}_{\mathtt{MSIS}}$, respectively, in Figure 16.

### A SECURITY OF AGGREGATE CASH SYSTEMS

**Theorem 10:** $\mathrm{ACS}_{\mathtt{SIS/MSIS}}$ *are complete, theft resistant, and provide zero-coin generation property if* $\mathrm{CTx}_{\mathtt{SIS/MSIS}}$ *are complete, theft resistant, and provide zero-coin generation property,* $\mathrm{Coin}_{\mathtt{SIS/MSIS}}$ *are complete, hiding, binding, and knowledge sound,* $\mathrm{SIG}_{\mathtt{SIS/MSIS}}$ *are complete and EUF-CMA, and* SIS/MSIS *are hard for* $(n, m, q, \gamma)$ *and* $(n, m, q, N, \gamma)$*, respectively.*

We prove Theorem 10 by showing the validity of each security property.

**Lemma 13:** $\mathrm{ACS}_{\mathtt{SIS/MSIS}}$ *are complete if* $\mathrm{CTx}_{\mathtt{SIS/MSIS}}$ *are complete,* $\mathrm{Coin}_{\mathtt{SIS/MSIS}}$ *are complete, and* $\mathrm{SIG}_{\mathtt{SIS/MSIS}}$ *are complete.*

*Proof:* We can directly see the completeness of the aggregable cash systems in Figure 17 due to Equation 2. □

**Lemma 14:** $\mathrm{ACS}_{\mathtt{SIS/MSIS}}$ *of* $(n, m, q, \gamma)$ *and* $(n, m, q, N, \gamma)$ *are theft resistant if* $\mathrm{CTx}_{\mathtt{SIS/MSIS}}$ *are theft resistant,* $\mathrm{Coin}_{\mathtt{SIS/MSIS}}$ *are binding, and* $\mathrm{SIG}_{\mathtt{SIS/MSIS}}$ *of* $(n, m-3L, q, \gamma)$ *and* $(n, m-3, q, N, \gamma)$ *are EUF-CMA.*

*Proof:* Let there be an algorithm $\mathcal{A}$ that wins the theft-resistance game. Then, we can simulate $\mathcal{A}$ to break one of the above properties. Therefore, we can reduce $\mathcal{A}$'s advantages such that $\mathsf{Adv}^{\mathsf{TR},\mathcal{A}}_{\mathrm{ACS}_{\mathtt{SIS}},pp_{(\lambda,L)}}$ is,

$$\mathsf{Adv}^{\mathsf{TR},\mathcal{A}}_{\mathrm{CTx}_{\mathtt{SIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{BND},\mathcal{A}}_{\mathrm{COIN}_{\mathtt{SIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG}_{\mathtt{SIS}},pp_{\lambda}}$$

and $\mathsf{Adv}^{\mathsf{TR},\mathcal{A}}_{\mathrm{ACS}_{\mathtt{MSIS}},pp_{(\lambda,L)}}$ is,

$$\mathsf{Adv}^{\mathsf{TR},\mathcal{A}}_{\mathrm{CTx}_{\mathtt{MSIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{BND},\mathcal{A}}_{\mathrm{COIN}_{\mathtt{MSIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG}_{\mathtt{MSIS}},pp_{\lambda}}$$

We claim Lemma 14 is true. □

**Lemma 15:** $\mathrm{ACS}_{\mathtt{SIS/MSIS}}$ *of* $(n, m, q, \gamma)$ *and* $(n, m, q, N, \gamma)$ *provide zero-coin generation property if* $\mathrm{CTx}_{\mathtt{SIS/MSIS}}$ *provide zero-coin generation property,* $\mathrm{Coin}_{\mathtt{SIS/MSIS}}$ *are knowledge sound,* $\mathrm{SIG}_{\mathtt{SIS/MSIS}}$ *of* $(n, m-3L, q, \gamma)$ *and* $(n, m-3, q, N, \gamma)$ *are EUF-CMA, and* $\mathtt{SIS/MSIS}$ *are hard.*

*Proof:* Let there be an algorithm $\mathcal{A}$ which wins $\mathsf{Game}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{ACS},pp_{(\lambda,L)}}$ of $\mathrm{ACS}_{\mathtt{SIS/MSIS}}$. Then $\mathcal{A}$ creates a cash system where the extractor $\mathcal{E}$ outputs $[v]_{i=0}^{|\mathtt{Ucoins}|}$ such that $\sum_{i=1}^{|\mathtt{Ucoins}|} v_i \neq S$. Since $\mathrm{CTx}_{\mathtt{SIS/MSIS}}$ provide zero-coin generation property, $\mathrm{Coin}_{\mathtt{SIS/MSIS}}$ are knowledge sound, and $\mathrm{SIG}_{\mathtt{SIS/MSIS}}$ are EUF-CMA, $\mathcal{A}$ finds the followings, for $\mathrm{ACS}_{\mathtt{SIS}}$

$$\underbrace{\boldsymbol{H}\left[\left[\sum_{i=1}^{|\mathtt{Ucoins}|}\mathsf{bin}(v) - \mathsf{bin}(S)\right], 0^{m-L}\right]}_{\boldsymbol{s}_{\mathtt{SIS}}\neq\mathbf{0}^m} = \mathbf{0}^n,$$

and for $\mathrm{ACS}_{\mathtt{MSIS}}$,

$$\underbrace{\vec{\boldsymbol{H}}\left[\left[\sum_{i=1}^{|\mathtt{Ucoins}|}\mathsf{bin}(v) - \mathsf{bin}(S)\right], \vec{0}^{m-1}\right]}_{\vec{\boldsymbol{s}}_{\mathtt{MSIS}}\neq\vec{\mathbf{0}}^m} = \vec{\mathbf{0}}^n$$

Here, $\boldsymbol{s}_{\mathtt{SIS}}$ and $\vec{\boldsymbol{s}}_{\mathtt{MSIS}}$ are solutions to $\mathtt{SIS/MSIS}$ problems. Therefore, $\mathcal{A}$'s advantages; $\mathsf{Adv}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{ACS}_{\mathtt{SIS}},pp_{(\lambda,L)}}$ is,

$$\mathsf{Adv}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx}_{\mathtt{SIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN}_{\mathtt{SIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG}_{\mathtt{SIS}},pp_{\lambda}} + \mathsf{Adv}^{\mathtt{SIS},\mathcal{A}}_{pp_{\lambda}},$$

and $\mathsf{Adv}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{ACS}_{\mathtt{MSIS}},pp_{(\lambda,L)}}$ is,

$$\mathsf{Adv}^{\mathsf{ZCG},\mathcal{A}}_{\mathrm{CTx}_{\mathtt{MSIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{KS},\mathcal{A}}_{\mathrm{COIN}_{\mathtt{MSIS}},pp_{(\lambda,L)}} + \mathsf{Adv}^{\mathsf{EUF},\mathcal{A}}_{\mathrm{SIG}_{\mathtt{MSIS}},pp_{\lambda}} + \mathsf{Adv}^{\mathtt{MSIS}}_{pp_{\lambda}}.$$

Therefore, we claim Lemma 15 is true. □

## XIII. IMPLEMENTATION OF MSIS CONFIDENTIAL TRANSACTIONS

We implement $\mathrm{COIN}_{\mathtt{MSIS}}$ and $\mathrm{CTx}_{\mathtt{MSIS}}$ protocols using the C language targeting root Hermite factor $\delta = 1.004$. It is widely believed that $\delta \leq 1.0045$ is sufficient to withstand known attacks of any chosen security level [33] (we chose $\lambda = 128$).

We always work with coin values in $[0, 2^{64})$ and polynomial space $\mathcal{R}_q = \mathbb{Z}_q[X]/[X^{256}+1]$. We state the parameters used for the implementation in Table 3. Note that the challenges are taken from the "hashing to a ball" method where the hash function is SHAKE-256 [43]. We only use uniform distributions for secret vectors and do not use any Gaussian distributions making it easy to secure against side channel attacks.

We use two polynomial multiplication methods.

1) Generic polynomial multiplication using Number Theoretic Transform (NTT) on 64 bit signed integers similar to [44], and
2) An easy multiplication for polynomial of $\mathsf{rot}()$.

Recall that we chose $q$ to have 512-th root of unity[13] $r \pmod q$. Therefore, $X^{256}+1$ splits into linear factors $X - r^i \pmod q$ with $i = 1, 3, 5, .., 511$. As a result, $\mathbb{Z}_q[X]/(X - r^i) \cong \mathbb{Z}_q$ (the Chinese remainder theorem). We can easily transform the polynomials in $\mathcal{R}_q$ to $\prod_i \mathbb{Z}_q[X]/(X + r^i)$ using the Fast Fourier Transform (or NTT since our field is finite) which we denote as,

$$\vec{a} \mapsto (a(r), a(r^3), a(r^5), .., a(r^{511}))$$
$$\mathcal{R}_q \to \prod_i \mathbb{Z}_q[X]/(X + r^i)$$

Since $X^{256} + 1 = X^{256} - r^{256}$, $\mathbb{Z}_q[X]/[X^{256}+1]$ is isomorphic to

$$\mathbb{Z}_q[X]/[X^{128} - r^{128}] \times \mathbb{Z}_q[X]/[X^{128} + r^{128}]$$
$$\mathbb{Z}_q[X]/[X^{128} - r^{128}] \times \mathbb{Z}_q[X]/[X^{128} - r^{384}].$$

At this stage, polynomial multiplication is point-wise. Note that natural NTT implementation does not output $a(r)$ directly. Therefore, following [45],

$$\mathsf{NTT}(\vec{a}) = (a(r^{\mathsf{brv}(128+i)}), a(-r^{\mathsf{brv}(128+i)}))_{i=0}^{127}$$

when $\mathsf{brv}$ is the bit-reversal algorithm. Now, we can denote $\vec{a}\vec{b} = \mathsf{NTT}^{-1}(\mathsf{NTT}(\vec{a}) \circ \mathsf{NTT}(\vec{b}))$ where $\circ$ is the point-wise multiplication. We use Cooley-Tukey butterflies for the forward transform [46], Gentleman-Sande butterflies for the inverse-transform [47], and Montgomery point-wise multiplications [48].

Our second multiplication is for polynomial formed from $\mathsf{rot}()$. Recall that $\vec{a} = \mathsf{rot}(j, i)$ outputs a polynomial such that $\vec{a}_i = j$ and other coefficients are zero. Therefore, we simply multiply all coefficient of $\vec{b}$ by $j$, change signs of the coefficients after the $i$th index, and rotate $i$ times to get $\vec{a}\vec{b}$.

[13] $r = 5834101087838$.

**TABLE 3.** Concrete parameters.

| Parameter | Value | Description |
|---|---|---|
| $N$ | 256 | |
| $q$ | $2^{50} - 2^{14} + 1$ | $q \equiv 1 \pmod{2N}$ |
| $\beta$ | 60 | $\log_2 \binom{N}{\beta} + \beta \geq 128 \times 2$ |
| $n$ | 6 | |
| $m$ | 5 | |
| $\gamma$ | $2^{32}$ | $q \gg \gamma$ |
| $\alpha$ | $2^{11} - 1$ | $\alpha \gg 2$ |
| $\tau$ | $2^2 - 1$ | $(m-3)N \log_2(2\tau) > 128 \times 6$ |
| $\tau_1$ | $2^7 - 1$ | $\tau_1 > \tau$ |
| $\tau_2$ | $2^{32} - 1$ | $\tau_2 \gg \beta^2\tau + \beta\tau_1$ |
| $\tau_3$ | $2^{16} - 1$ | $\tau_3 \gg 5\beta\tau$ |

**TABLE 4.** Component sizes of MSIS-CTx protocol.

| Symbol | $\vec{z}$ | $\vec{r}$ | $\vec{u}/\vec{t}_1/\vec{t}_2/\vec{pk}$ | $\vec{\sigma}$ for $(2 \Rightarrow 2)$ |
|---|---|---|---|---|
| Size | 22.5 KB | 2.0 KB | 9.6 KB | 4.1 KB |

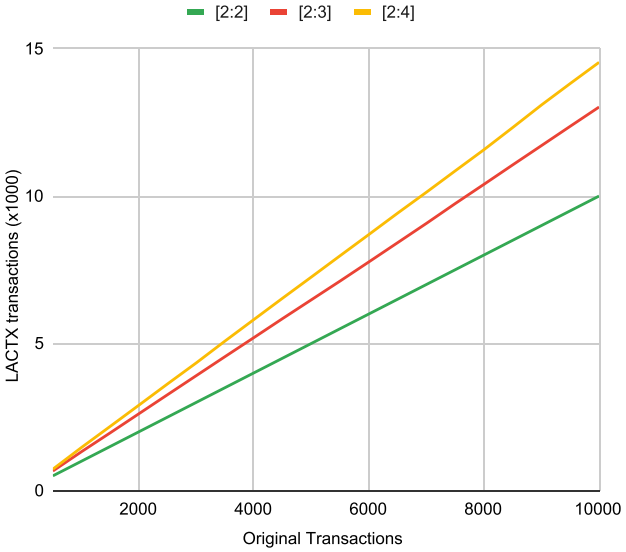



**FIGURE 18.** Transaction increment with the maximum 2 inputs and 2 outputs.

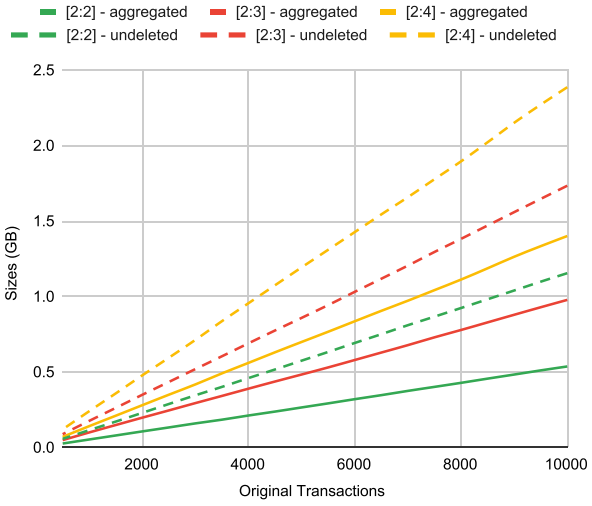



**FIGURE 19.** Size reductions with transaction aggregation.

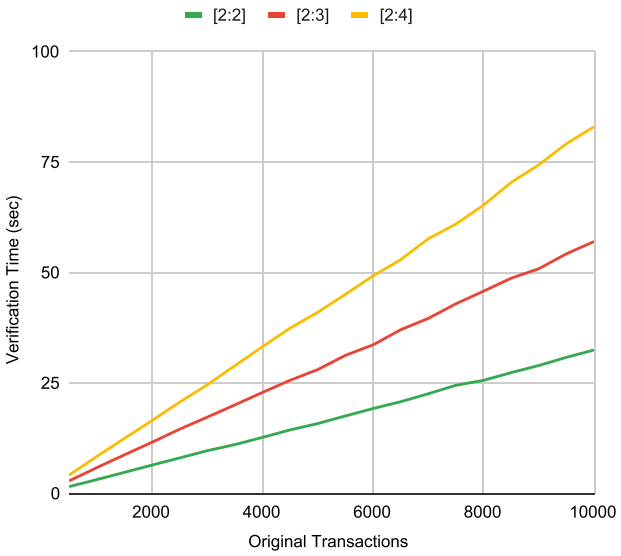



**FIGURE 20.** Verification time of aggregated cash systems.

### A. BENCHMARKS

We run benchmarks for different average input and output rates; [2:2], [2:3], and [2:4] since the statistics show that the average input and output rate is [2:3]. We uniformly choose input sizes from $[1, x]$ and output sizes from $[1, y]$ when the rate is $[x : y]$. For the benchmarks, we use SQLite3 [49] as the database of the cash system. When we measure the size of the cash system, we actually measure the size of the database. The measured size of the cash system is slightly higher than the theoretical size due to the identifiers and index tables used for faster searching. Also, we transfer 2000 coins from the coinbase per 10 transactions. Therefore, 2 million coins circulate in the system at the end of each benchmark.

First, we check what happens when we limit the maximum number of inputs and outputs to 2. We use, "original transactions" for any real transaction with average [input:output] proportions. Later we separate them into the transactions of maximum 2 inputs and 2 outputs which we call "LACTX transactions". Figure 18 shows the number of LACTX transactions when the numbers of inputs and outputs of original transactions are chosen uniformly with average input and output rates; [2:2], [2:3], and [2:4].

Next, we check the cash systems' size reductions after the aggregation. Figure 19 shows the cash systems' sizes before and after the transaction aggregation. According to the figure, we see that all input and output rates have significant size reductions even if the LACTX transactions are limited to 2 inputs and 2 outputs.

We check the verification times of the aggregated cash systems. We use an i7-1065G7 CPU @1.30GHz to measure verification times. From Figure 20, we see that the average input-output rate affects the verification time. However, the

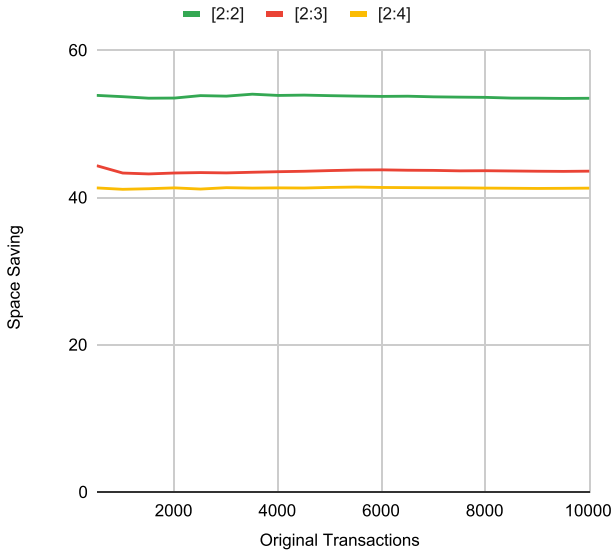



**FIGURE 21.** Space savings of aggregated cash systems.

verification time is shorter after the aggregation since there are no spent coin bundles.

Finally, we show the space savings of our aggregate cash systems for different input and output rates in Figure 21. Here, space saving is computed as,

$$\texttt{spacesaving} = \frac{\texttt{size} - \texttt{aggregated\_size}}{\texttt{size}} \times 100\%$$

## XIV. CONCLUSION

Post-quantum zero-knowledge (PQ-ZK) protocols from lattices have a potential for unrivalled security and privacy, but, in such massively multi-party systems as cryptocurrencies, they tend to be less practical than their quantum-vulnerable discrete-logarithm counterparts. To reconcile the appeal of post-quantum security and confidentiality with the need for efficiency and scalability in truly decentralized cash systems, we introduce two efficient lattice-based aggregable confidential transaction (CT) protocols that respectively rely on the (plain) SIS and (ring-like) MSIS lattice problems. We use them to construct a stateless or histoty-free efficient blockchain-based cash system, where old coin records can be safely deleted once spent. Our protocols achieve a higher efficiency through the use of commitments in binary form, while retaining the ability to perform correct arithmetic in zero-knowledge through the judicious use of carries. Our experiments with our MSIS-based CT implementation show 40%–54% size reductions from ZK transaction aggregation.

## REFERENCES

[1] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed. Berlin, Germany: Springer, 2013, pp. 34–51.

[2] M. Spagnuolo, F. Maggi, and S. Zanero, "BitIodine: Extracting intelligence from the Bitcoin network," in *Financial Cryptography and Data Security*, N. Christin and R. Safavi-Naini, Eds. Berlin, Germany: Springer, 2014, pp. 457–468.

[3] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," 2015, *arXiv:1502.01657*.

[4] F. Reid and M. Harrigan, "An analysis of anonymity in the Bitcoin system," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust IEEE 3rd Int. Conf. Social Comput.*, 2011, pp. 1318–1326, doi: 10.1109/PASSAT/SocialCom.2011.79.

[5] J. Herrera-Joancomartí, "Research and challenges on Bitcoin anonymity," in *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, J. Garcia-Alfaro, J. Herrera-Joancomartí, E. Lupu, J. Posegga, A. Aldini, F. Martinelli, and N. Suri, Eds. Cham, Switzerland: Springer, 2015, pp. 3–16.

[6] M. C. K. Khalilov and A. Levi, "A survey on anonymity and privacy in bitcoin-like digital cash systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2543–2585, 3rd Quart., 2018.

[7] L. Morris, "Anonymity analysis of cryptocurrencies," M.S. thesis, Rochester Inst. Technol., Rochester, NY, USA, 2015. [Online]. Available: https://scholarworks.rit.edu/theses/8616/

[8] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[10] S. Noether and S. Noether. (2014). *Monero is Not That Mysterious*. [Online]. Available: https://web.getmonero.org/ru/resources/research-lab/pubs/MRL-0003.pdf

[11] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 397–411.

[12] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 459–474.

[13] M. Divya and N. B. Biradar, "IOTA-next generation block chain," *Int. J. Eng. Comput. Sci.*, vol. 7, no. 4, pp. 23823–23826, Apr. 2018.

[14] Bitcoin Wiki. *Simplified Payment Verification*. Accessed: Nov. 15, 2021. [Online]. Available: https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification

[15] T. E. Jedusor. (2016). *MimbleWimble*. [Online]. Available: https://docs.beam.mw/Mimblewimble.pdf

[16] A. Poelstra. (2016). *2016-10-06 (Commit e9f45ec)*. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf

[17] G. Fuchsbauer, M. Orrù, and Y. Seurin, "Aggregate cash systems: A cryptographic investigation of mimblewimble," in *Advances in Cryptology—EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham, Switzerland: Springer, 2019, pp. 657–689.

[18] J. Alupotha and X. Boyen, "Origami store: UC-secure foldable datachains for the quantum era," *IEEE Access*, vol. 9, pp. 81454–81484, 2021.

[19] S. Noether and A. Mackenzie, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, Dec. 2016.

[20] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *Computer Security—ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham, Switzerland: Springer, 2017, pp. 456–474.

[21] M. F. Esgin, R. K. Zhao, R. Steinfeld, J. K. Liu, and D. Liu, "MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 567–584.

[22] G. Maxwell. (2015). *Confidential Transactions*. Accessed: Jan. 9, 2021. [Online]. Available: https://people.xiph.org/~greg/confidential_values.txt

[23] *Inputs Per TX-Daily Median Input Statistics Per Transaction and Block, Excluding Coinbase Transaction (Miner Reward)*. Accessed: Jan. 9, 2021. [Online]. Available: https://bitcoinvisuals.com/chain-input-count-tx

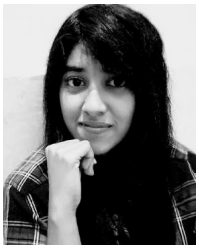
[24] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO '91*, J. Feigenbaum, Ed. Berlin, Germany: Springer, 1992, pp. 129–140.

[25] H. Zhang, F. Zhang, B. Wei, and Y. Du, "Implementing confidential transactions with lattice techniques," *IET Inf. Secur.*, vol. 14, no. 1, pp. 30–38, Jan. 2020.

[26] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 186–194.

[27] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung, "Design validations for discrete logarithm based signature schemes," in *Public Key Cryptography*, H. Imai and Y. Zheng, Eds. Berlin, Germany: Springer, 2000, pp. 276–292.

[28] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, Mar. 2000.
[29] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre, "From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security," in *Advances in Cryptology—EUROCRYPT 2002*, L. R. Knudsen, Ed. Berlin, Germany: Springer, 2002, pp. 418–433.
[30] A. Kawachi, K. Tanaka, and K. Xagawa, "Concurrently secure identification schemes based on the worst-case hardness of lattice problems," in *Advances in Cryptology—ASIACRYPT 2008*, J. Pieprzyk, Ed. Berlin, Germany: Springer, 2008, pp. 372–389.
[31] V. Lyubashevsky, "Lattice-based identification schemes secure under active attacks," in *Public Key Cryptography—PKC 2008*, R. Cramer, Ed. Berlin, Germany: Springer, 2008, pp. 162–179.
[32] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *Advances in Cryptology—ASIACRYPT 2009*, M. Matsui, Ed. Berlin, Germany: Springer, 2009, pp. 598–616.
[33] M. R. Albrecht. *LWE Estimator*. Accessed: Oct. 22, 2021. [Online]. Available: https://lwe-estimator.readthedocs.io/en/latest/readme_link.html
[34] T. H. Yuen, S.-F. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Springer, 2020, pp. 464–483.
[35] J. Alupotha, X. Boyen, and E. Foo, "Compact multi-party confidential transactions," in *Cryptology and Network Security*, S. Krenn, H. Shulman, and S. Vaudenay, Eds. Cham, Switzerland: Springer, 2020, pp. 430–452.
[36] H. Zhang, F. Zhang, H. Tian, and M. H. Au, "Anonymous post-quantum cryptocash," in *Financial Cryptography and Data Security*, S. Meiklejohn and K. Sako, Eds. Berlin, Germany: Springer, 2018, pp. 461–479.
[37] W. A. A. Torres, R. Steinfeld, A. Sakzad, J. K. Liu, V. Kuchta, N. Bhattacharjee, M. H. Au, and J. Cheng, "Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1.0)," in *Information Security and Privacy*, W. Susilo and G. Yang, Eds. Cham, Switzerland: Springer, 2018, pp. 558–576.
[38] W. A. Torres, V. Kuchta, R. Steinfeld, A. Sakzad, J. K. Liu, and J. Cheng, "Lattice RingCT V2.0 with multiple input and multiple output wallets," in *Information Security and Privacy*, J. Jang-Jaccard and F. Guo, Eds. Cham, Switzerland: Springer, 2019, pp. 156–175.
[39] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Efficient range proofs for confidential transactions," *IEEE Security Privacy*, pp. 315–334, 2018, doi: 10.1109/SP.2018.00020.
[40] G. Maxwell and A. Poelstra. (2015). *Borromean Ring Signatures*. [Online]. Available: https://raw.githubusercontent.com/Blockstream/borromean_paper/master/borromean_draft_0.01_34241bb.pdf
[41] A. Poelstra, A. Back, M. Friedenbach, G. Maxwell, and P. Wuille, "Confidential assets," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Springer, 2018, pp. 43–63.
[42] R. Impagliazzo, L. A. Levin, and M. Luby, "Pseudo-random generation from one-way functions," in *Proc. 21st Annu. ACM Symp. Theory Comput.*, 1989, pp. 12–24.
[43] M. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," Federal Inf. Process. Standards, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Aug. 2015, doi: 10.6028/NIST.FIPS.202.
[44] PQ-Crystals. (2019). *Dilithium Signature Scheme*. [Online]. Available: https://github.com/pq-crystals/dilithium
[45] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—A new hope," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*, 2016, pp. 327–343.
[46] C. F. Gauss, *Nachlass: Theoria Interpolationis Methodo Nova Tractata*, vol. 3. Göttingen, Germany: Koeniglichen Gesellschaft der Wissenchaften Goettingen, 1866, pp. 265–327.
[47] W. M. Gentleman and G. Sande, "Fast Fourier transforms: For fun and profit," in *Proc. Fall Joint Comput. Conf.*, 1966, pp. 563–578.
[48] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, pp. 519–521, 1985.
[49] SQLite. *What is SQLite?* Accessed: Sep. 10, 2020. [Online]. Available: https://sqlite.org



**JAYAMINE ALUPOTHA** is currently pursuing the Ph.D. degree in computer science with the Queensland University of Technology, Australia. Her research interest includes cryptography, mainly blockchain and post-quantum cryptography.

**XAVIER BOYEN** received the Ph.D. degree from Stanford University. He is an Associate Professor at the Queensland University of Technology (QUT). He is the creator of multiple cryptographic protocols extensively used in real-world software and hardware. His research interests include efficient and decentralized cryptography as an enabler of individual freedoms and privacy, especially in relation with human limitations. He was the recipient of multiple awards, the author of about 100 papers with 10,000–20,000 citations, and a Future Fellow of the ARC.

**MATTHEW MCKAGUE** received the B.Sc. degree (Hons.) in mathematics from the University of Regina, Regina, SK, Canada, in 2004, and the M.Math. and Ph.D. degrees in combinatorics and optimization from the University of Waterloo, Waterloo, ON, Canada, in 2005 and 2010, respectively. He was a Research Fellow with the Centre for Quantum Technologies, Singapore, and a Lecturer with the Computer Science Department, University of Otago, Dunedin, New Zealand. He is currently a Lecturer in cryptography with the Queensland University of Technology, Brisbane, QLD, Australia.