Omniring: Scaling Private Payments Without Trusted Setup

Formal Foundations and a Construction of Ring Confidential Transactions with Log-size Proofs

Russell W. F. Lai
Friedrich-Alexander University Erlangen-Nuremberg

Viktoria Ronge
Friedrich-Alexander University Erlangen-Nuremberg

Tim Ruffing
Blockstream

Dominique Schröder
Friedrich-Alexander University Erlangen-Nuremberg

Sri Aravinda Krishnan Thyagarajan
Friedrich-Alexander University Erlangen-Nuremberg

Jiafan Wang
Chinese University of Hong Kong

ABSTRACT

Monero is the largest cryptocurrency with built-in cryptographic privacy features. The transactions are authenticated using zero-knowledge spend proofs, which provide a certain level of anonymity by hiding the source accounts from which the funds are sent among a set of other accounts. Due to its similarities to ring signatures, this core cryptographic component is called Ring Confidential Transactions (RingCT). Because of its practical relevance, several works attempt to analyze the security of RingCT. Since RingCT is rather complex, most of them are either informal, miss fundamental functionalities, or introduce undesirable trusted setup assumptions. Regarding efficiency, Monero currently deploys a scheme in which the size of the spend proof is linear in the ring size. This limits the ring size to only a few accounts, which in turn limits the acquired anonymity significantly and facilitates de-anonymization attacks.

As a solution to these problems, we present the first rigorous formalization of RingCT as a cryptographic primitive. We then propose a generic construction of RingCT and prove it secure in our formal security model. By instantiating our generic construction with new efficient zero-knowledge proofs, we obtain *Omniring*, a fully-fledged RingCT scheme in the discrete logarithm setting that provides the highest concrete and asymptotic efficiency as of today. Omniring is the first RingCT scheme which 1) does not require a trusted setup or pairing-friendly elliptic curves, 2) has a proof size *logarithmic* in the size of the ring, and 3) allows to share the same ring between all source accounts in a transaction, thereby enabling significantly improved privacy level without sacrificing performance. Our zero-knowledge proofs rely on novel enhancements to the Bulletproofs framework (S&P 2018), which we believe are of independent interest.

KEYWORDS

Cryptocurrency; Privacy; Anonymity; Monero

ACM Reference Format:
Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. 2019. Omniring: Scaling Private Payments Without Trusted Setup: Formal Foundations and a Construction of Ring Confidential Transactions with Log-size Proofs. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3319535.3345655

1 INTRODUCTION

Modern cryptocurrencies such as Bitcoin and Monero eliminate the need for a trusted central party, giving rise to fully decentralized and publicly verifiable currency systems. A cryptocurrency typically consists of two components: (1) a public ledger, *e.g.*, realized by a blockchain protocol, to publish transactions, and (2) a transaction scheme which specifies the format and validity of transactions. For example, we can think of the simplest transaction type in Bitcoin as simply requiring a signed statement of the form "Pseudonym i pays amount a to pseudonym j". This allows for easy public verification of the ledger but on the flip side, the inherently public nature of cryptocurrencies is a threat for the individual privacy of their users. At first glance, the usage of pseudonyms may give the impression that users are anonymous, but a long list of literature demonstrates that different pseudonyms belonging to the same user can be linked by simple as well as sophisticated heuristics when observing transactions on the public blockchain [2, 4, 25, 34, 35, 44, 50]. As a consequence, transactions can be traced and users can be re-identified.

In order to improve this situation, many privacy-enhancing technologies have been proposed by the academic and the cryptocurrency community [6, 7, 20, 21, 24, 31, 32, 36, 43, 45, 47, 52, 53, 58], and multiple cryptocurrencies with a special focus on privacy have emerged [37, 56, 57]. Monero [37] with a market capitalization of $1.6 billion at the time of writing [30] is the largest such privacy-focused cryptocurrency. In contrast to other approaches, most notably Zerocash [6], which relies on a trusted cryptographic setup to be able to scale to very large anonymity sets, one of the main design goals of Monero is to avoid any form of trusted setup. This approach is arguably much closer to the original spirit of cryptocurrencies whose point is to avoid centralization as much as possible.

For privacy, Monero uses *Ring Confidential Transactions* (RingCT) proposed by Noether *et al.* [39] as its cryptographic core component. The idea behind their scheme is an ad-hoc approach to integrate three privacy-enhancing technologies:

Its first component is similar in spirit to linkable ring signatures [29] which guarantee some form of *anonymity* and enable to detect double-spends simultaneously. The second component of RingCT is Confidential Transactions (CT) [32], which hides the monetary amounts of transactions in homomorphic commitments while still being able to verify that every transaction is *balanced*, *i.e.*, the sum of the amounts sent to target accounts does not exceed the sum of amounts available in the source accounts. The third component is Stealth Addresses (SA) [53], which provides a form of *receiver anonymity.* Given just a single long-term public key of a receiver (called stealth address), a spender can derive an arbitrary number of seemingly unrelated public keys owned by this receiver, which avoids that two transactions paying the same receiver can be linked.

The original RingCT construction of Noether *et al.* [39], which is used in Monero, produces spend proofs of size $O(|\mathcal{S}|\cdot|\mathcal{R}|+|\mathcal{T}|+\beta)$, where $|\mathcal{R}|$, $|\mathcal{S}|$, and $|\mathcal{T}|$ are the size of the ring, the set of source accounts, and the set of target accounts in a transaction respectively, and 2^β is the maximum currency amount that can be sent in a single transaction. A core component in the construction is a zero-knowledge "range proof" which allows a spender to prove that an amount being transferred lies within the pre-defined range $[0,\ldots,2^\beta-1]$. This range proof is a necessary building block to ensure the security of Confidential Transactions. Recently a new range proof system, based on the Bulletproofs framework [10] for zero-knowledge proofs, has been deployed in Monero [49]. This new range proof system reduces the linear dependency on β to a logarithmic one, and the linear dependency on $|\mathcal{S}|\cdot|\mathcal{R}|$ (which includes the ring size) becomes the bottleneck of the current Monero system. This inefficiency incentivizes the use of small ring sizes (currently 11 in Monero) and thus effectively reduces the anonymity set, which facilitates de-anonymization of the spending accounts as shown by Möser *et al.* [38] and Kumar *et al.* [26].

1.1 Our Contributions

Despite its practical importance, there is a lack of theoretical foundations and satisfactory constructions for RingCT. In this work, we overcome both of these shortcomings. First, we provide an extensive formalization of RingCT. Our formalization guarantees security against several realistic attacks which are not covered by the security models of prior work (e.g., due to an unrealistically strong assumption that keys are generated honestly in certain cases), and our formalization covers Stealth Addresses unlike most previous formalizations. Second, we put forward a new construction of RingCT which is significantly better than existing ones in terms of supported features and practical efficiency. The efficiency improvements allow implementations to choose larger parameters that strengthen privacy without sacrificing performance. Our main contributions are summarized as follows.

1.1.1 Rigorous Formalization of RingCT. The necessity of *precise* security models for cryptographic primitives cannot be overstated, as they concisely point out security guarantees, allow comparison, and serve as a guideline for protocol design. An example which highlights this necessity is the denial-of-spending weakness [46] in Zerocoin [20, 36], a different cryptographic approach for privacy in cryptocurrencies. This threat was not captured by the insufficient security model of Zerocoin and lead to vulnerabilities in multiple cryptocurrencies allowing an attacker to destroy funds of honest users. Notably, denial-of-spending attacks are not considered in the RingCT proposal by Noether *et al.* [39] either, even though they generally apply to the RingCT setting as well [46]. While the concrete RingCT scheme by Noether *et al.* [39] seems not susceptible to such attacks, the fact that the threat has apparently been overlooked in [39] underlines the importance of a rigorous and thorough security model.

Sun *et al.* [51] and Yuen *et al.* [55] propose formalizations of RingCT that improve on the rather informal security notions by Noether *et al.* [39]. Unfortunately, the security models in both of these works are still too weak because they fail to cover some realistic attacks, and in the case of [51] the model does not support stealth addresses. In the following, we focus on highlighting the strengths of our model, and we defer the comparison to the two aforementioned formalizations to Section 7.

Capturing Stealth Addresses. Our model is the first one that captures stealth addresses. This is a critical component of the overall security model because it relates directly to receiver anonymity. Moreover, it affects the entire functionality of the primitive and all other security properties.

Non-reliance on External Communication Channels. Our model only assumes a public ledger onto which transactions can be published and does not rely on any external secure channels.

Stronger Security Guarantees. We provide stronger security definitions for balance and spender anonymity. In contrast to prior work, our definition requires balance even if all accounts in the transaction are maliciously generated and in case of spender anonymity, we allow some of the source accounts to be corrupt and still require the other non-corrupt accounts to be anonymous.

Unified Ring for All Source Accounts. All previous RingCT schemes use separate rings for each source accounts. This means that transactions that spend from multiple source accounts (as is common in cryptocurrencies), each source account is anonymous in a *separate* anonymity set. In our model, all source accounts of a single transaction share one ring, hence the name "Omniring". This approach does not only improve efficiency, but it also improves the level of anonymity: Let us consider the case of spending from k source accounts. In the separated-rings approach, each source account is hidden within a different ring of some size n, meaning that each of the k source accounts has at most 1-out-of-n anonymity. On the other hand, in our unified ring approach, having a ring of size kn offers up to k-out-of-kn anonymity.

Now consider for instance the case that one of the real source accounts used for spending is de-anonymized. In the unified ring approach, the other real source accounts now still have $(k-1)$-out-of-$(kn-1)$ anonymity, *i.e.*, all other accounts in the unified ring still count towards the crowd to hide in. However, in the separated-ring approach, the entire ring containing the de-anonymized account would be useless for anonymity after de-anonymization.

1.1.2 Efficient Construction. We propose a new construction of RingCT, Omniring, whose spend proof size is only $O(\log(|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|))$, where $|\mathcal{R}|$, $|\mathcal{S}|$, and $|\mathcal{T}|$ are the size of the ring, the set of source accounts, and the set of target accounts, respectively, and 2^{β} is the maximum allowed currency amount to be transferred. Our scheme is the first scheme that does not require a trusted setup or pairings, supports stealth addresses, and has a logarithmic spend proof size.

Our construction follows the high-level idea of combining a ring signature scheme with a range proof system. We first propose a generic construction of RingCT from signatures of knowledge (SoK) of a certain language. In the second step, we develop techniques to extend the Bulletproofs framework [10] into an argument system for proving knowledge of a discrete logarithm representation, where the exponents in the representation satisfy an arbitrary arithmetic circuit. The statements of the desired language can be expressed in a format which is optimized for the extended Bulletproofs framework. This allows us to exploit the efficiency of Bulletproofs not just for range proofs but also for the spender to prove his ownership of coins and their spendability, and we can combine all proof statements into a single zero-knowledge proof. This leads to a very efficient RingCT construction. Moreover, our extension to Bulletproofs has potentially far-reaching consequences: It leads to a natural construction of logarithmic-size ring signatures, which is competitive with state-of-the-art schemes [9, 20]. The technique can also be generalized to proving general (bilinear) group arithmetic relations [28], which makes it a topic of independent interest.

1.1.3 Adaption to Monero. Our main instantiation presented in Section 4.4 is designed to simplify the language that it induces, and cannot directly be integrated into Monero due to the difference in the format of the linkability tags (or "key images"). To tackle this issue, we formally detail in the full version [27] how our Omniring construction can be adapted and made readily deployable in Monero. The adaption retains essentially the same efficiency as our main instantiation except for a slightly higher computational effort.

1.2 On Ring Selection

Our formalization follows the spirit of ring signatures and does not cover how rings are sampled. We believe that this question of formalizing what a "good" ring sampler is, is orthogonal to formalizing the properties of a RingCT and constructing an efficient scheme. The question of finding good ring sampling strategies also does not seem to be of cryptographic nature, as the ring sampler does not involve any cryptographic keys. We believe that understanding the strategies of ring selection, and hence maximizing the non-deanonymized subsets of sampled rings, are important questions that deserve an in-depth investigation in an independent paper. Our view is motivated by recent attacks against the anonymity provided by cryptocurrency based on RingCT as discussed by Möser *et al.* [38] and Kumar *et al.* [26]. This line of work shows that the ring sampling strategies of the spenders are critical.

What we do formalize is the intuition that, given a ring of accounts selected by some external mechanisms, the source accounts of a transaction are hidden within the *non-deanonymized* subset of the ring.

1.3 Technical Overview

Recall that the statements to be proven in our RingCT construction use signatures of knowledge and consist of two parts. The first part corresponds to knowing the secret key of one of the ring accounts, and the second part guarantees that the amount being transferred lies within a certain range. The first natural idea to construct a RingCT scheme is to combine a state-of-the-art ring signature scheme (*e.g.*, by Groth and Kohlweiss [20] or Bootle and Groth [8]) with the most efficient range proof scheme to date (in the Bulletproofs framework [10]), which has a logarithmic size in the bitlength of the upper limit of the range. However, since both proof systems rely on significantly different techniques, combining them naïvely yields a RingCT scheme with signature size asymptotically equal to that of ours but with worse concrete efficiency.

We then explore the possibility of building a ring signature scheme using the techniques of [10], so that it can be combined natively with all Bulletproofs range proofs into one single zero-knowledge proof of a single combined statement, leveraging the logarithmic size of Bulletproofs as much as possible. In order to understand the challenge that we tackle while exploring this path, recall that Bulletproofs is a framework for proving "inner product relations" between the exponents in a discrete logarithm representation. Although the Bulletproofs framework is expressive enough for capturing arithmetic circuits satisfiability, it is particularly optimized for range proofs. For instance, proving that a committed integer a lies within a range $[0, 2^{\beta}-1]$ yields a proof whose size is only $O(\log \beta)$.

In a Bulletproofs range proof, the prover encodes the binary representation of a in the vector $\vec{\mathbf{a}}$ and sets $\vec{\mathbf{b}} = \vec{\mathbf{a}} - \vec{1}^{|\vec{\mathbf{a}}|}$. It commits to $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ as $A = F^{\alpha}\vec{\mathbf{G}}^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$ with randomness α and public (vector of) group elements F, $\vec{\mathbf{G}}$, and $\vec{\mathbf{H}}$. It then proves that $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ satisfy the Hadamard product relation $\vec{\mathbf{a}} \circ \vec{\mathbf{b}} = \vec{0}^{|\vec{\mathbf{a}}|}$, $\vec{\mathbf{a}} - \vec{\mathbf{b}} = \vec{1}^{|\vec{\mathbf{a}}|}$, and the inner product relation $\langle \vec{\mathbf{a}}, \vec{2}^{|\vec{\mathbf{a}}|}\rangle = a$. These relations guarantee that $\vec{\mathbf{a}}$ is a valid binary representation of a. Moreover, the length $|\vec{\mathbf{a}}|$ guarantees that a must be between 0 and $2^{|\vec{\mathbf{a}}|}-1$. The extractability of the proof crucially depends on the assumption that non-trivial discrete logarithm representations of the identity element with respect to base $(F\|\vec{\mathbf{G}}\|\vec{\mathbf{H}})$ are unknown to the prover. Under such an assumption, one can extract the exponents $(\alpha, \vec{\mathbf{a}}, \vec{\mathbf{b}})$ in the discrete logarithm representation of A.

With the basics above, we can highlight the technical difficulties one encounters when attempting to construct ring membership proofs using techniques of the Bulletproofs framework.

Let the vector of group elements $\vec{\mathbf{R}} = (R_1\|\ldots\|R_n)$ consist of the public keys of the ring members. In a ring membership proof, one would like to prove the knowledge of a tuple (i, x_i) such that $R_i = H^{x_i}$ for a public generator H. Equivalently, the prover would prove its knowledge of a unit vector $\vec{\mathbf{e}}_i$ (whose i-th entry is 1 and zero everywhere else) and an integer $-x_i$ such that $I = H^{-x_i}\vec{\mathbf{R}}^{\vec{\mathbf{e}}_i}$ where I is the identity element; we call this relation the *main equality*.

A natural idea of using the technique from Bulletproofs for a ring membership proof is to embed the term $H^{-x_i}\vec{\mathbf{R}}^{\vec{\mathbf{e}}_i}$ into a part of the commitment A, show that $\vec{\mathbf{e}}_i$ is indeed a unit vector by defining certain inner product relations, and at the same time show that the main equality holds. Implementing this idea comes with two main challenges: First, we need a way to actually embed the aforementioned term in A. Second, regardless of how the term is embedded, we need to overcome the issue that the prover might know the discrete logarithms between elements in $\vec{\mathbf{R}}$, which forbids to argue soundness in the same way as in Bulletproofs. As a solution to these challenges, we propose a general technique to embed the main equality (or in general any representation of the identity element) into the commitment A, while at the same time avoiding the above problem regarding soundness.

First, we observe that if $\vec{\mathbf{P}}$ is a vector of group elements chosen randomly and independently of $\vec{\mathbf{R}}$, then for any $w \in \mathbb{Z}_q$, the discrete logarithm representation problem base $\vec{\mathbf{G}}_w := (H\|\vec{\mathbf{R}})^w \circ \vec{\mathbf{P}}$ is equivalent to the standard discrete logarithm problem.

Second, let $\vec{\mathbf{a}} = (-x_i\|\vec{\mathbf{e}}_i)$. Note that $\vec{\mathbf{G}}_w^{\vec{\mathbf{a}}} = \vec{\mathbf{G}}_{w'}^{\vec{\mathbf{a}}}$ for any $w, w' \in \mathbb{Z}_q$ due to the main equality (introduced above). Therefore, if $A = F^\alpha \vec{\mathbf{G}}_w^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$ for some $w \in \mathbb{Z}_q$ and some vector of group elements $\vec{\mathbf{b}}$, then for any other $w' \in \mathbb{Z}_q$ it also holds that $A = F^\alpha \vec{\mathbf{G}}_{w'}^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$.

With the above observations, we let the prover run a Bulletproofs-style protocol twice on A with respect to two different bases, *i.e.*, $(F\|\vec{\mathbf{G}}_w\|\vec{\mathbf{H}})$ and $(F\|\vec{\mathbf{G}}_{w'}\|\vec{\mathbf{H}})$. If the prover is able to convince the verifier in both executions, then we can construct an extractor which extracts the exponents $(\alpha, \vec{\mathbf{a}}, \vec{\mathbf{b}})$ such that $A = F^\alpha \vec{\mathbf{G}}_w^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}} = F^\alpha \vec{\mathbf{G}}_{w'}^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$. Dividing the two representations of A yields the main equality.

At this point, we have already obtained a "Bulletproofs-friendly" protocol for ring membership proofs, which can be combined with the range proofs of [10] to construct a very efficient spend algorithm in RingCT. However, this approach requires to execute the Bulletproof protocol twice, which blows up the proof size by a factor of 2. Our third observation allows to compress the two prover executions into a single one. Recall that in the second observation we have $A = F^\alpha \vec{\mathbf{G}}_w^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$ for all $w \in \mathbb{Z}_q$. Therefore the prover can first compute $A = F^\alpha \vec{\mathbf{G}}_0^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$ (*i.e.*, $w = 0$) without knowing w, and then obtain a random w as a challenge by hashing A. If the prover is able to convince the verifier with a randomly chosen w, then in the proof of extractability we can run the prover twice on w and w' respectively, and then apply the analysis mentioned above.

2 FORMALIZING RINGCT

We present an extensive formalization of Ring Confidential Transactions (RingCT), which in particular incorporates the stealth address feature. We first describe the intended use of each algorithm along with some conventions that we adopt (Section 2.1), then provide a formalization of the core syntax (Section 2.2). In the full version [27], we further extend the syntax to provide tracking and viewing features, which enable a user to delegate detection and decoding of incoming transactions.

2.1 Overview

We overview the core functionality of RingCT.

Setup and Joining. A RingCT scheme is initialized by running the Setup algorithm. Anyone can join the system by generating a key-tuple (mpk,msk) using SAKGen, where mpk is the master public key, msk is the master secret key. The master public key is also called a stealth address as it allows the derivation of one-time target accounts for receiving funds.

Transaction. A transaction tx consists of a set of ring accounts $\{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}$, tags $\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}$, target accounts $\{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}$, and some optional arbitrary message μ. The "(linkability) tag" is known in the context of cryptocurrencies as "serial number" (Zcash) or "key image" (Monero), but we follow the terminology of linkable ring signatures [29]. The public key is bound to a secret key sk, while the coin is bound to a secret coin key ck and a secret amount a.

Spending. Let

$$\mathcal{R} = \left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|} \text{ and } \mathcal{S} = \left\{(j_i, \mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{sk}_i, \mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|},$$

where $\mathcal{R}$ is a set of ring accounts sampled by some external mechanism, such that a user knows a set of indices, secret keys, coin keys, amounts and tags $\mathcal{S}$ corresponding to the source accounts $\left\{\mathsf{acc}_{j_i}^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{S}|}$. The user can transfer a batch of amounts $\{a_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}$ to the owners of the stealth addresses $\{\mathsf{mpk}_i\}_{i=1}^{|\mathcal{T}|}$ as follows. It first executes OTAccGen on each $(\mathsf{mpk}_i, a_i^{\mathcal{T}})$ tuple to generate a one-time target account $\mathsf{acc}_i^{\mathcal{T}}$, and a coin key ck_i. This process is sometimes known as minting. The account $\mathsf{acc}_i^{\mathcal{T}}$ will be made publicly available, while the coin key ck_i is kept secret by the spender. Let $\mathcal{T} = \left\{(\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|}$ specifying the target accounts and other relevant information, and μ be some additional message that the user wishes to include as part of the transaction. The user runs Spend on $(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$ to create a proof σ that the transaction tx (defined by $(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$) is valid. The transaction tx and the proof σ are then published (*e.g.*, in a public ledger) for verification. In practice, there may be parts of the source and target accounts, *e.g.* trapdoor information intended to be used by the receiver, that are not necessary for public verification. They can instead be sent to the receiver off-chain; we do not model this explicitly.

Verification. Once a tuple (tx,σ) is published, all parties can run the Vf algorithm to verify its validity. Roughly speaking, a spend proof σ is considered valid for a transaction tx if it demonstrates that the total amounts in the source accounts are equal to that in the target accounts. The infeasibility of forging a proof on an invalid transaction will be formally modeled as the balance property defined in Section 3.1.

In a cryptocurrency system, the verifiers need to perform two additional checks other than the validity of the proof to verify the validity of the transaction in the context of the ledger. First, they need to check that none of the tags in $\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}$ appeared in a previous transaction (double-spending). The balance property will guarantee that this check is sufficient to detect double-spending. Second, they need to check that each of the ring accounts $\{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}$ in the transaction tx has been a target account in some previous transaction or is a "coinbase" account, *i.e.*, an account with which newly mined coins are generated. In the latter case, the miner runs OTAccGen to generate a new account with her stealth address and publishes the created account together with the amount and coin key to claim newly mined coins in a coinbase account; then verifiers can verify the correctness of the claimed amount by CheckAmount.

Receiving. A user can receive funds from a target account acc that it owns by running Receive on acc which was published (*e.g.*, via a public ledger) as part of a transaction. Using the output of Receive, which includes a secret key of the target account, the amount in the account and its tag, the receiver can later spend the received coin in another transaction.

In a cryptocurrency system, the receiver will need to check additionally that he has not already received a different transaction to the same account (with the same tag), because then only one set of received funds will be spendable and the other will be considered a double-spend. This malicious reuse of the account by the spender is known as the faerie gold attack [54] or burning bug [14]. To ensure that this check is sufficient, the balance property will guarantee that the tag output by Receive is indeed the correct tag for the given target account (such that duplicate accounts can be detected by duplicate tags), as well as that the amount output by Receive is the amount which is stored in the account.

2.2 Formal Syntax

Let $\lambda \in \mathbb{N}$ denote the security parameter. We denote by poly (λ) the set of polynomials in λ and we write negl (λ) for a function negligible in λ. PPT means probabilistic polynomial time. Given a set S, $x \leftarrow_\$ S$ means sampling an element x from S uniformly at random. For an algorithm A with input x and output z, we write $z \leftarrow A(x)$. For $n \in \mathbb{N}$ the set $[n]$ is defined as $[n] := \{1,\dots,n\}$. Unless specified otherwise, all sets are implicitly ordered. All algorithms may output $\perp$ upon failure.

Definition 2.1. *A* Ring Confidential Transactions *(RingCT) scheme consists of a tuple of main* PPT *algorithms* (Setup,SAKGen,OTAccGen,Receive,Spend,Vf)*, and a tuple of auxiliary* PPT *algorithms* (CheckAmount, CheckTag) *defined as follows.*

$\underline{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)}$*: The* setup *algorithm takes as inputs the security parameter* 1^λ *and integers* $1^\alpha,1^\beta$*, where* 1^α *represents an upper bound* 2^α *on the number of outputs in a single transaction and* 1^β *an upper bound* 2^β *of amounts to be transferred in a transaction. It outputs the public parameter* pp *to be given to all algorithms implicitly.*

$\underline{(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{SAKGen}(\mathsf{pp})}$*: The* master key generation *algorithm takes as inputs the public parameter* pp*, and outputs a master public key* mpk *and a master secret key* msk*. The master public key* mpk *is also known as a stealth address.*

$\underline{(\mathsf{ck},\mathsf{acc}) \leftarrow \mathsf{OTAccGen}(\mathsf{mpk},a)}$*: The* one-time account generation *algorithm takes as inputs a master public key* mpk *and an amount* $a \in \{0,\dots,2^\beta-1\}$*. It outputs a coin key* ck *and an account* acc.

$\underline{(\mathsf{ck},a,\mathsf{sk},\mathsf{tag}) \leftarrow \mathsf{Receive}(\mathsf{msk},\mathsf{acc})}$*: The* receive *algorithm takes as inputs a master secret key* msk *and an account* acc*. It outputs a coin key* ck*, an amount a, a secret key* sk*, and a* tag.

$\underline{\sigma \leftarrow \mathsf{Spend}(\mathcal{R},\mathcal{S},\mathcal{T},\mu)}$*: The* spend *algorithm takes the following inputs:*

- $\mathcal{R} = \{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}$*: a set of ring accounts* $\mathsf{acc}_i^{\mathcal{R}}$
- $\mathcal{S} = \left\{(j_i,\mathsf{ck}_i^{\mathcal{S}},a_i^{\mathcal{S}},\mathsf{sk}_i,\mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|}$*: a set of tuples consisting of an index* $j_i \in [|\mathcal{R}|]$*, a coin key* $\mathsf{ck}_i^{\mathcal{S}}$*, a secret key* sk_i*, an amount* $a_i^{\mathcal{S}}$*, and a tag* tag_i *(of* $\mathsf{acc}_{j_i}^{\mathcal{R}}$*)*
- $\mathcal{T} = \{(\mathsf{ck}_i^{\mathcal{T}},a_i^{\mathcal{T}},\mathsf{acc}_i^{\mathcal{T}})\}_{i=1}^{|\mathcal{T}|}$*: a set of tuples consisting of a coin key* $\mathsf{ck}_i^{\mathcal{T}}$*, an amount* $a_i^{\mathcal{T}}$*, and a target account* $\mathsf{acc}_i^{\mathcal{T}}$
- μ*: an optional message to be signed*

It outputs a proof σ.

$\underline{b \leftarrow \mathsf{Vf}(\mathsf{tx},\sigma)}$*: The* verify *algorithm takes as inputs a transaction* tx *and a signature* σ*. It outputs a bit b indicating the validity of* σ*. A transaction* tx *defined as follows:*

$$\mathsf{tx}(\mathcal{R},\mathcal{S},\mathcal{T},\mu) := \left(\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \left\{\mathsf{acc}_i^{\mathcal{T}}\right\}_{i=1}^{|\mathcal{T}|}, \mu \right)$$

$\underline{b \leftarrow \mathsf{CheckAmount}(\mathsf{acc},\mathsf{ck},a)}$*: The* amount checking *algorithm takes as inputs an account* acc*, a coin key* ck*, and an amount a. It outputs a bit b indicating the consistency of the inputs.*

$\underline{b \leftarrow \mathsf{CheckTag}(\mathsf{acc},\mathsf{sk},\mathsf{tag})}$*: The* tag checking *algorithm takes as inputs an account* acc*, a secret key* sk*, and a tag* tag*. It outputs a bit b indicating the consistency of the inputs.*

Definition 2.2 (Correctness). *A RingCT scheme is* correct *if the following holds for all* λ, α, $\beta \in \mathbb{N}$*, and all* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$.

- *Honestly generated payments should be received correctly. Concretely, for any* $(\mathsf{mpk},\mathsf{msk}) \in \mathsf{SAKGen}(\mathsf{pp})$*, any amount* $a \in \{0,\dots,2^\beta-1\}$*, any* $(\mathsf{ck},\mathsf{acc}) \in \mathsf{OTAccGen}(\mathsf{mpk},a)$*, and any* $(\mathsf{ck}',a',\mathsf{sk},\mathsf{tag}) \in \mathsf{Receive}(\mathsf{msk},\mathsf{acc})$*, it holds that* $(\mathsf{ck},a) = (\mathsf{ck}',a')$.
- *Correctly received payments should have well-defined amounts and tags. Concretely, for any* $(\mathsf{ck},a,\mathsf{sk},\mathsf{tag}) \in \mathsf{Receive}(\mathsf{msk},\mathsf{acc})$ *(*$\neq \perp$*), it holds that* $\mathsf{CheckAmount}(\mathsf{acc},\mathsf{ck},a) = 1$ *and* $\mathsf{CheckTag}(\mathsf{acc},\mathsf{sk},\mathsf{tag}) = 1$.
- *Honestly generated transactions should be recognized as valid. Concretely, for any tuple* $(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$ *with syntax as defined in Definition 2.1 that satisfies*
  - $|\mathcal{T}| \le 2^\alpha$,
  - *for all* $i \in [|\mathcal{T}|]$, $a_i^{\mathcal{T}} \in \left\{0,\dots,2^\beta-1\right\}$,
  - *for all* $i \in [|\mathcal{S}|]$, $\mathsf{CheckTag}(\mathsf{acc}_{j_i}^{\mathcal{R}},\mathsf{sk}_i,\mathsf{tag}_i) = 1$,
  - *for all* $i \in [|\mathcal{S}|]$, $\mathsf{CheckAmount}(\mathsf{acc}_{j_i}^{\mathcal{R}},\mathsf{ck}_i^{\mathcal{S}},a_i^{\mathcal{S}}) = 1$,

```
InitOracles()
/ Initialize Lists
$\mathsf{MPK} := \mathsf{MSK} := \emptyset$
/ Initialize Sets
$\mathsf{Spent} := \Sigma := \emptyset$

SAKGen$\mathcal{O}$()
/ Generate keys for a new honest user.
$(\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{SAKGen}(\mathsf{pp})$
$\mathsf{MPK} := \mathsf{MPK} \| \mathsf{mpk}$
$\mathsf{MSK} := \mathsf{MSK} \| \mathsf{msk}$
return mpk

TryReceive(acc)
/ Called by Spend $\mathcal{O}$
for $i \in [|\mathsf{MSK}|]$
    $(\mathsf{ck}, a, \mathsf{sk}, \mathsf{tag}) \leftarrow \mathsf{Receive}(\mathsf{MSK}[i], \mathsf{acc})$
    if $(\mathsf{ck}, a, \mathsf{sk}, \mathsf{tag}) \neq \perp$
        return $(\mathsf{ck}, a, \mathsf{sk}, \mathsf{tag})$
    endif
endfor
return $\perp$

Spend$\mathcal{O}(I,\mathcal{R},\mathcal{S},\mathcal{T},\mu)$
/ Instruct honest spender(s) to generate a proof.
/ $\mathcal{R}$ and $\mathcal{S}$ are (incomplete) lists containing malicious information.
/ $I$ contains instructions for populating $\mathcal{R}$ and $\mathcal{S}$ with information of honest spenders.
/ For each $(s_i, j_i, \mathsf{acc}_i)$ in $I$, fill in $\mathcal{S}[s_i]$ and $\mathcal{R}[j_i]$ using data provided by TryReceive.
parse $I$ as $\{(s_i, j_i, \mathsf{acc}_i)\}_{i=1}^{|I|}$
for $i \in [|I|]$ do
    $(\mathsf{ck}_i, a_i, \mathsf{sk}_i, \mathsf{tag}_i) := \mathsf{TryReceive}(\mathsf{acc}_i)$
    $\mathcal{R}[j_i] := \mathsf{acc}_i$
    $\mathcal{S}[s_i] := (j_i, \mathsf{ck}_i, a_i, \mathsf{sk}_i, \mathsf{tag}_i)$
endfor
$\mathsf{tx} := \mathsf{tx}(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$
$\sigma \leftarrow \mathsf{Spend}(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$
$\Sigma := \Sigma \cup \{(\mathsf{tx},\sigma)\}$
if $\mathsf{Vf}(\mathsf{tx}_t, \sigma_t) = 0$ then return 0
$\mathsf{Spent} := \mathsf{Spent} \cup \{\mathsf{tag}_i\}_{i=1}^{|I|}$
return $\sigma$
```

Figure 1: Oracles for security experiments.

– *for all* $i \in [|\mathcal{T}|]$, $\mathsf{CheckAmount}(\mathsf{acc}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}) = 1$, *and*
– $\sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} = \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}}$
and for any proof $\sigma \in \mathsf{Spend}(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$, *it holds that* $\mathsf{Vf}(\mathsf{tx},\sigma) = 1$, *where* $\mathsf{tx} = \mathsf{tx}(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$.

3 SECURITY OF RINGCT

We formalize the security properties of RingCT, namely balance, privacy, and non-slanderability. The oracles used in the security games are shown in Figure 1. Unlike prior work, our definitions take into account stealth addresses. In the full version [27] we extend the definitions further to account for the tracking and viewing features.

3.1 Balance

Balance roughly means that a spender cannot double-spend, or spend more than what it possesses. The formal definition (Definition 3.1) is more complicated than one would initially expect because the amounts being transferred in a transaction are confidential. In more detail, we say that a RingCT scheme is balanced if the following two properties are satisfied.

First, the predicates CheckTag and CheckAmount are required to be "binding" in a sense similar to a commitment scheme. The binding property of CheckTag ensures that a tag is computationally bound to a source account, which in turn ensures that checking for duplicate tags is sufficient to prevent double-spending. Similarly, the binding property of CheckAmount ensures that an amount is computationally bound to an account, which ensures that money cannot be "created out of thin air" by changing the amount of coins in a given account. This formalization does not contradict the mining of new coins, because this is modeled by explicitly creating a new account (see Section 2.1). These binding properties make the balance experiment, defined below, meaningful.

```
Balance$_{\Omega,\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^\lambda, 1^\alpha, 1^\beta)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$
$(\mathsf{tx},\sigma) \leftarrow \mathcal{A}(\mathsf{pp})$
$(\mathcal{R},\mathcal{S},\mathcal{T},\mu) \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp},\mathsf{tx},\sigma)$
parse $\mathcal{R}$ as $\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}$
parse $\mathcal{S}$ as $\left\{(j_i, \mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{sk}_i, \mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|}$
parse $\mathcal{T}$ as $\left\{(\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|}$
$b_0 := \mathsf{Vf}(\mathsf{tx},\sigma)$
$b_1 := (\mathsf{tx} = \mathsf{tx}(\mathcal{R},\mathcal{S},\mathcal{T},\mu))$
$b_2 := \left(\forall i \in [|\mathcal{S}|],\ \mathsf{CheckTag}(\mathsf{acc}_{j_i}^{\mathcal{R}}, \mathsf{sk}_i, \mathsf{tag}_i) = 1\right)$
$b_3 := \left(\forall i \in [|\mathcal{S}|],\ \mathsf{CheckAmount}(\mathsf{acc}_{j_i}^{\mathcal{R}}, \mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}}) = 1\right)$
$b_4 := \left(\forall i \in [|\mathcal{T}|],\ \mathsf{CheckAmount}(\mathsf{acc}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}) = 1\right)$
$b_5 := \left(\sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} \geq \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}}\right)$
return $b_0 \wedge \neg(b_1 \wedge b_2 \wedge b_3 \wedge b_4 \wedge b_5)$
```

Figure 2: Balance experiment.

The second property requires that, for any efficient adversary $\mathcal{A}$ which produces a transaction with a proof, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ such that, if the proof is valid (Event b_0), then the extractor can extract the witness (*e.g.*, secret keys, amounts, *etc.*) leading to the transaction with high probability. More concretely, the latter means that all of the following events (Events b_1 to b_5) must occur with high probability.

Let tx be the transaction output by $\mathcal{A}$, and $(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$ be the tuple extracted by $\mathcal{E}_{\mathcal{A}}$, with the format $\mathcal{R} = \left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}$, $\mathcal{S} = \left\{(j_i, \mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{sk}_i, \mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|}$, and $\mathcal{T} = \left\{(\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|}$.

- Event b_1: The extracted tuple leads to the adversarial transaction, *i.e.*, $\mathsf{tx} = \mathsf{tx}(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$.
- Event b_2: sk_i is consistent with the j_i-th ring account $\mathsf{acc}_{j_i}^{\mathcal{R}}$ and the i-th tag tag_i, *i.e.*, $\mathsf{CheckTag}(\mathsf{acc}_{j_i}^{\mathcal{R}}, \mathsf{sk}_i, \mathsf{tag}_i) = 1$.
- Event b_3: The source coin key and amount $(\mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}})$ are consistent with the j_i-th ring account $\mathsf{acc}_{j_i}^{\mathcal{R}}$, *i.e.*, $\mathsf{CheckAmount}(\mathsf{acc}_{j_i}^{\mathcal{R}}, \mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}}) = 1$.
- Event b_4: The target coin key and amount $(\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}})$ are consistent with the i-th target account $\mathsf{acc}_i^{\mathcal{T}}$, *i.e.*, $\mathsf{CheckAmount}(\mathsf{acc}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}) = 1$.
- Event b_5: The sum of the source amount is at least the sum of the target amount, *i.e.*, $\sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} \geq \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}}$.

```
$\mathsf{Privacy}^b_{\Omega,\mathcal{A}}(1^\lambda,1^\alpha,1^\beta)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$, $\mathsf{InitOracles}()$
$\mathbb{O} := \{\mathsf{SAKGen}\mathcal{O}, \mathsf{Spend}\mathcal{O}\}$
$(I,J,\mathcal{R},\mathcal{S},\mathcal{T},\mu) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$
$\mathcal{S}_0 := \mathcal{S}_1 := \mathcal{S}$, $\mathcal{T}_0 := \mathcal{T}_1 := \mathcal{T}$
/ Preparing honest spenders as instructed by adversary.
parse $I$ as $\left\{(s_i, \{j_{t,i}, \mathsf{acc}_{t,i}\}_{t=0}^1)\right\}_{i=1}^{|I|}$
for $i \in [|I|]$ do
  for $t \in \{0,1\}$ do
    $(\mathsf{ck}^{\mathcal{S}}_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{tag}_{t,i}) := \mathsf{TryReceive}(\mathsf{acc}^{\mathcal{S}}_{t,i})$
    $\mathcal{R}[j_{t,i}] := \mathsf{acc}^{\mathcal{S}}_{t,i}$
    $\mathcal{S}_t[s_i] := (j_{t,i}, \mathsf{ck}^{\mathcal{S}}_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{tag}_{t,i})$
  endfor
  if $\mathsf{tag}_{0,i} \neq \mathsf{tag}_{1,i} \wedge \{\mathsf{tag}_{0,i}, \mathsf{tag}_{1,i}\} \cap \mathsf{Spent} \neq \emptyset$ then return $0$
endfor
/ Preparing honest receivers as instructed by adversary.
parse $J$ as $\left\{(d_j, \{k^{\mathcal{T}}_{t,j}, a^{\mathcal{T}}_{t,j}\}_{t=0}^1)\right\}_{j=1}^{|J|}$
for $j \in [|J|]$ do
  for $t \in \{0,1\}$ do
    $(\mathsf{ck}^{\mathcal{T}}_{t,j}, \mathsf{acc}^{\mathcal{T}}_{t,j}) := \mathsf{OTAccGen}(\mathsf{MPK}[k^{\mathcal{T}}_{t,j}], a^{\mathcal{T}}_{t,j})$
    $\mathcal{T}_t[d_j] := (\mathsf{ck}^{\mathcal{T}}_{t,j}, a^{\mathcal{T}}_{t,j}, \mathsf{acc}^{\mathcal{T}}_{t,j})$
  endfor
endfor
for $t \in \{0,1\}$ do
  $\mathsf{tx}_t := \mathsf{tx}(\mathcal{R}, \mathcal{S}_t, \mathcal{T}_t, \mu)$
  $\sigma_t \leftarrow \mathsf{Spend}(\mathcal{R}, \mathcal{S}_t, \mathcal{T}_t, \mu)$
  if $\mathsf{Vf}(\mathsf{tx}_t, \sigma_t) = 0$ then return $0$
endfor
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{tx}_b, \sigma_b)$
return $b'$
```

Figure 3: Privacy experiment.

```
$\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda,1^\alpha,1^\beta)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$, $\mathsf{InitOracles}()$
$(\mathsf{tx}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SAKGen}\mathcal{O}, \mathsf{Spend}\mathcal{O}}(\mathsf{pp})$
parse $\mathsf{tx}^*$ as $\left(\left\{\mathsf{acc}^{\mathcal{R}}_i\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \left\{\mathsf{acc}^{\mathcal{T}}_i\right\}_{i=1}^{|\mathcal{T}|}, \mu\right)$
$b_0 := \mathsf{Vf}(\mathsf{tx}^*, \sigma^*)$
$b_1 := ((\mathsf{tx}^*, \sigma^*) \notin \Sigma)$
$b_2 := \left(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|} \cap \mathsf{Spent} \neq \emptyset\right)$
return $b_0 \wedge b_1 \wedge b_2$
```

Figure 4: Non-slanderability experiment.

The two properties can be interpreted in the following way. If a spender (*e.g.*, the adversary) can produce a transaction with a valid proof, then it must possess knowledge of balanced input and output amounts, as they can be extracted by $\mathcal{E}_{\mathcal{A}}$. If the actual amounts of the source and target accounts are different from those extracted by the extractor, *e.g.*, the spender attempts to create money out of thin air, then one can break the binding property of CheckAmount. Therefore the amounts that the spender has in mind cannot be different from those extracted by $\mathcal{E}_{\mathcal{A}}$. Similarly, if the spender attempts to spend from the same account twice by producing different tags for the account, then with the spender and the extractor $\mathcal{E}_{\mathcal{A}}$ one can break the binding property of CheckTag. Therefore double-spending is infeasible.

Definition 3.1 (Balance). *A RingCT scheme* Ω *is* balanced *if:*

(1) CheckTag *and* CheckAmount *are binding. That is, for any* PPT *adversary* $\mathcal{A}$*, for all positive integers* $\alpha, \beta \in \mathrm{poly}(\lambda)$*, for* $\mathsf{Chk} \in \{\mathsf{CheckTag}, \mathsf{CheckAmount}\}$*, we have*

$$\Pr\left[\begin{array}{c} \mathsf{Chk}(\mathsf{acc}, k, m) = 1 \\ \mathsf{Chk}(\mathsf{acc}, k', m') = 1 \\ (k, m) \neq (k', m') \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta) \\ (\mathsf{acc}, k, m, k', m') \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \mathrm{negl}(\lambda).$$

(2) *For all* PPT *adversaries* $\mathcal{A}$*, and all positive integers* $\alpha, \beta \in \mathrm{poly}(\lambda)$*, there exists a* PPT *extractor* $\mathcal{E}_{\mathcal{A}}$ *such that*

$$\Pr\left[\mathsf{Balance}_{\Omega,\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^\lambda, 1^\alpha, 1^\beta) = 1\right] \leq \mathrm{negl}(\lambda),$$

where $\mathsf{Balance}_{\Omega,\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^\lambda, 1^\alpha, 1^\beta)$ *is defined in Figure 2.*

3.2 Privacy

Privacy captures anonymity for both the spenders and the receivers, and the confidentiality of the amounts being transferred. The formalization of privacy is inspired by that of anonymity in ring signatures, key-privacy of encryption, and hiding in commitments. While closely related to the anonymity of ring signatures, the spender anonymity aspect of RingCT is significantly more difficult to capture as it must still hold in the presence of stealth addresses, a concept that does not exist for ring signatures.

Roughly speaking, privacy means that an adversary should not be able to distinguish two transactions with the same ring and their proofs, even if the majority of the ring is corrupt and the adversary has prior knowledge about the identities of the spenders and receivers and the amounts being transferred. In more detail, the adversary is allowed to specify a ring with arbitrarily many corrupt accounts, and two honest subsets of the ring which are the potential spenders. The adversary also specify two sets of receivers and the amounts that they are supposed to receive. A transaction is then created using one of the two specifications of the adversary, who should not be able to tell which specification is used to create the transaction.

More concretely, we model privacy in the security experiments $\mathsf{Privacy}^b$ for $b \in \{0,1\}$. Let $\mathcal{A}$ be a PPT adversary who, after several queries to the oracles, produces an *incomplete* input $(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$ to the Spend algorithm, along with two instructions I and J. The incomplete input $(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$ corresponds to all malicious information that will be used to generate a transaction and its proof, while the instructions I and J specify how the sets $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ should be populated by information held by the honest users. Following the adversary's instructions, the experiment duplicates $(\mathcal{S}, \mathcal{T})$ into $(\mathcal{S}_0, \mathcal{T}_0)$ and $(\mathcal{S}_1, \mathcal{T}_1)$, and populates $(\mathcal{R}, \mathcal{S}_t, \mathcal{T}_t)$ for $t \in \{0,1\}$ as follows.

The instruction I corresponds to the source accounts and is of the form $\left\{(s_i, \{j_{t,i}, \mathsf{acc}^{\mathcal{S}}_{t,i}\})\right\}_{i=1}^{|I|}$. The experiment retrieves the information required to spend from account $\mathsf{acc}^{\mathcal{S}}_{t,i}$ by calling TryReceive if possible. It sets $\mathcal{R}[j_{t,i}]$ to this account, and $\mathcal{S}_t[s_i]$ to the retrieved spender information.

Similarly, the instruction J corresponds to the target accounts is of the form $\left\{(d_j, \{k^{\mathcal{T}}_{t,j}, a^{\mathcal{T}}_{t,j}\}^{1}_{t=0})\right\}^{|J|}_{j=1}$. The experiment creates a one-time account using the master public key of user $k^{\mathcal{T}}_{t,j}$ and the amount $a^{\mathcal{T}}_{t,j}$, and set $\mathcal{T}_t[d_j]$ to the appropriate receiver information.

The experiment then proceeds to create the proofs σ_0 and σ_1 for both transactions $\mathsf{tx}(\mathcal{R}, \mathcal{S}_0, \mathcal{T}_0, \mu)$ and $\mathsf{tx}(\mathcal{R}, \mathcal{S}_1, \mathcal{T}_1, \mu)$ respectively. If both proofs are valid, meaning in particular that both transactions created as instructed by the adversary are well-formed, the experiment sends σ_b to the adversary, where b is the parameter of the experiment $\mathsf{Privacy}^b$.

Definition 3.2 (Privacy). *A RingCT scheme* Ω *is* private *if for all* PPT *adversaries* $\mathcal{A}$ *and all positive integers* $\alpha, \beta \in \mathsf{poly}(\lambda)$,

$$\left|\Pr\left[\mathsf{Privacy}^0_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1\right] - \Pr\left[\mathsf{Privacy}^1_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Privacy}^b_{\Omega,\mathcal{A}}$ *is defined in Figure 3.*

Remark. One may think of a seemingly stronger privacy game in which the adversary gets an additional corruption oracle $\mathsf{Corrupt}\mathcal{O}$ that is used to obtain the master secret keys of honest parties before and after being given the challenge. To show that this game is not stronger than the current version, we give an informal argument. We show that one can construct an adversary $\mathcal{A}$ against Privacy that uses an adversary $\mathcal{B}$ against Privacy with $\mathsf{Corrupt}\mathcal{O}$. $\mathcal{A}$ guesses two users that will be honest and own two source accounts acc_{0,i_0} and acc_{1,i_1} respectively provided by $\mathcal{B}$. With inverse-polynomial probability, $\mathcal{A}$ guesses correctly. To generate a well-formed challenge for $\mathcal{B}$, the algorithm $\mathcal{A}$ guesses the hidden bit b of its challenger and provides the corresponding secret keys for other "honest" users, except for the account acc_{b,i_b}. With probability $1/2$, $\mathcal{A}$ guesses correctly and the challenge proof returned from its challenger is a valid challenge for $\mathcal{B}$. Overall, the advantage of $\mathcal{A}$ is only a polynomial factor lower than that of $\mathcal{B}$.

3.3 Non-Slanderability (and Unforgeability)

In the context of RingCT, slandering is an act of producing a valid proof on behalf of another user. Note that a proof authenticates a transaction which specifies a set of tags bound to a set of source accounts. If the owner of one of the source accounts later attempts to spend from the account, the proof will not be accepted because the tag corresponding to the account has already been published in the slandering transaction. Slandering thus effectively causes the owners of these source accounts to lose money. Non-slanderability is a property that prevents the above attack, which is known as denial-of-spending attack in the literature [46].

Formally, we model non-slanderability by defining a security experiment in which the adversary produces a transaction-proof tuple, after several queries to the oracles (Figure 1). The adversary is successful if the tuple is valid and not produced by the spend oracle, and some of the tags specified in the slandering transaction collide with those that are signed by the spend oracle.

```
OneWay_{Tag,A}(1^λ)
  $pp \leftarrow \mathsf{TagSetup}(1^\lambda)$
  $x, s^* \leftarrow \chi,\ X \leftarrow \mathsf{TagKGen}(x)$
  $\mathsf{tag}^* \leftarrow \mathsf{TagEval}(x + s^*)$
  $x' \leftarrow \mathcal{A}^{\mathsf{Tag}\mathcal{O}_x}(pp, s^*, \mathsf{tag}^*)$
  return $(\mathsf{TagEval}(x') = \mathsf{tag}^*)$

PR^b_{Tag,A}(1^λ)
  $pp \leftarrow \mathsf{TagSetup}(1^\lambda)$
  $x \leftarrow \chi,\ X \leftarrow \mathsf{TagKGen}(x)$
  $(s^*, \mathsf{tag}^*) \leftarrow \mathsf{Ch}_{b,x}()$
  $b' \leftarrow \mathcal{A}^{\mathsf{Tag}\mathcal{O}_x}(pp, X, s^*, \mathsf{tag}^*)$
  return $b'$

TagO_x()
  $s \leftarrow \chi$
  return $(s, \mathsf{TagEval}(x + s))$

Ch_{x,b}()
  $\mathsf{tag}^* \leftarrow \psi$
  $s^* \leftarrow \chi$
  if $b = 0$ then
    $\mathsf{tag}^* \leftarrow \mathsf{TagEval}(x + s^*)$
  endif
  return $(s^*, \mathsf{tag}^*)$
```

Figure 5: One-wayness and pseudorandomness experiments for tagging schemes.

Definition 3.3 (Non-slanderability). *A RingCT scheme* Ω *is* non-slanderable *if for all* PPT *adversaries* $\mathcal{A}$ *and all* $\alpha, \beta \in \mathsf{poly}(\lambda)$,

$$\Pr\left[\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1\right] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{NSland}_{\Omega,\mathcal{A}}$ *is defined in Figure 4.*

Since a tag is computationally bound to a unique account (as required by the balance property), non-slanderability (which states that no adversary can forge under a tag of a honest user account), naturally captures that no adversary can forge spend proofs for honest accounts. As a consequence, we do not need to define an unforgeability property explicitly.

4 RINGCT CONSTRUCTION

We present a generic construction Ω for a RingCT scheme and an efficient instantiation $\mho$.

4.1 Tagging Scheme

Our generic construction Ω depends on a new primitive called tagging scheme. Roughly speaking, a tagging scheme is a one-way permutation over group elements.

Formally, a tagging scheme Tag = (TagSetup, TagKGen, TagEval) consists of a PPT setup algorithm TagSetup, an efficient *bijection* TagKGen, and an efficient *deterministic* algorithm TagEval. TagSetup inputs the security parameter 1^λ and outputs the public parameter pp, which defines a secret key space $(\chi, +)$, which is a group equipped with the operation $+$, a key space $(\mathcal{X}, \cdot)$, which is a group equipped with the operation $\cdot$, and a tag space ψ. TagKGen inputs $x \in \chi$ and outputs a public key $X \in \mathcal{X}$. Furthermore, Tag is homomorphic, *i.e.*, for any $x, x' \in \chi$, $\mathsf{TagKGen}(x) \cdot \mathsf{TagKGen}(x') = \mathsf{TagKGen}(x + x')$. TagEval inputs $x \in \chi$ and outputs a tag $\mathsf{tag} \in \psi$.

We require a tagging scheme which satisfies (related-input) one-wayness and pseudorandomness, defined as follows.

```
Setup(1^λ, 1^α, 1^β)
    $\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKESetup}(1^\lambda)$
    $\mathsf{pp}_{\mathsf{HC}} \leftarrow \mathsf{HCSetup}(1^\lambda)$
    $\mathsf{pp}_{\mathsf{Tag}} \leftarrow \mathsf{TagSetup}(1^\lambda)$
    $\mathsf{pp}_{\mathsf{SoK}} \leftarrow \mathsf{SoKSetup}(1^\lambda, (\mathsf{pp}_{\mathsf{HC}}, \mathsf{pp}_{\mathsf{Tag}}))$
    $\mathsf{pp} := (\beta, \mathsf{pp}_{\mathsf{PKE}}, \mathsf{pp}_{\mathsf{HC}}, \mathsf{pp}_{\mathsf{Tag}}, \mathsf{pp}_{\mathsf{SoK}})$
    return pp

SAKGen(pp)
    $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathsf{PKE}})$
    $(\mathsf{vpk}, \mathsf{vsk}) \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathsf{PKE}})$
    $\bar{\mathsf{sk}} := x \leftarrow \chi,\ \bar{\mathsf{pk}} := X \leftarrow \mathsf{TagKGen}(x)$
    $\mathsf{mpk} := (\mathsf{tpk}, \mathsf{vpk}, \bar{\mathsf{pk}})$
    $\mathsf{msk} := (\mathsf{tsk}, \mathsf{vsk}, \bar{\mathsf{sk}})$
    return (mpk, msk)

OTAccGen(mpk, a)
    parse mpk as $(\mathsf{tpk}, \mathsf{vpk}, \bar{\mathsf{pk}})$
    $\mathsf{ek} \leftarrow \$\{0,1\}^\lambda,\ \mathsf{ck} := r \leftarrow \rho$
    $s := \mathsf{H}(\mathsf{mpk}, \mathsf{ek})$
    $\mathsf{pk} := \bar{\mathsf{pk}} \cdot \mathsf{TagKGen}(s),\ \mathsf{co} := \mathsf{Com}(a;r)$
    $\tilde{\mathsf{ek}} \leftarrow \mathsf{Enc}(\mathsf{tpk}, (\mathsf{pk}, \mathsf{co}), \mathsf{ek})$
    $\tilde{\mathsf{ck}} \leftarrow \mathsf{Enc}(\mathsf{vpk}, (\mathsf{pk}, \mathsf{co}), (a, r))$
    $\mathsf{acc} := (\mathsf{pk}, \mathsf{co}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
    return (ck, acc)

Vf(tx, σ)
    parse tx as $\left(\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \left\{\left(\mathsf{acc}_i^{\mathcal{T}}\right)\right\}_{i=1}^{|\mathcal{T}|}, \mu\right)$
    if $|\mathcal{T}| > 2^\alpha$ then return 0
    return $b := \mathsf{SoKVf}(\mathsf{stmt}(\mathsf{tx}), \sigma, \mathsf{tx})$

Spend(R, S, T, μ)
    $\mathsf{tx} := \mathsf{tx}(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$
    return $\sigma \leftarrow \mathsf{SoKSig}(\mathsf{stmt}(\mathsf{tx}), \mathsf{wit}(\mathcal{S}, \mathcal{T}), \mathsf{tx})$

Receive(msk, acc)
    parse msk as $(\mathsf{tsk}, \mathsf{vsk}, \bar{\mathsf{sk}})$
    parse acc as $(\mathsf{pk}, \mathsf{co}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
    $\mathsf{ek} \leftarrow \mathsf{Dec}(\mathsf{tsk}, (\mathsf{pk}, \mathsf{co}), \tilde{\mathsf{ek}})$
    $(a, r) \leftarrow \mathsf{Dec}(\mathsf{vsk}, (\mathsf{pk}, \mathsf{co}), \tilde{\mathsf{ck}})$
    $s := \mathsf{H}(\mathsf{mpk}, \mathsf{ek}),\ \mathsf{sk} := \bar{\mathsf{sk}} + s$
    if $(\mathsf{pk}, \mathsf{co}) \neq (\mathsf{TagKGen}(\mathsf{sk}), \mathsf{Com}(a;r))$ then return $\perp$
    $\mathsf{tag} := \mathsf{TagEval}(\mathsf{sk})$
    return $(r, a, \mathsf{sk}, \mathsf{tag})$
```

Figure 6: RingCT construction Ω (core components).

```
CheckAmount(acc, ck, a)
    parse acc as (pk, co, ek, ck)
    return $(\mathsf{co} = \mathsf{Com}(a;\mathsf{ck}))$

CheckTag(acc, sk, tag)
    parse acc as (pk, co, ek, ck)
    return $(\mathsf{tag} = \mathsf{TagEval}(\mathsf{sk})$
        $\wedge\, \mathsf{pk} = \mathsf{TagKGen}(\mathsf{sk}))$
```

Figure 7: RingCT construction (auxiliary algorithms).

Definition 4.1 (Security of Tagging Schemes). *A tagging scheme* Tag *is* related-input one-way *if for any* PPT *adversary* $\mathcal{A}$,

$$\Pr\left[\mathsf{OneWay}_{\mathsf{Tag},\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}\ (\lambda).$$

It is related-input pseudorandom *if for any* PPT *adversary* $\mathcal{A}$,

$$\left|\Pr\left[\mathsf{PR}^0_{\mathsf{Tag},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{PR}^1_{\mathsf{Tag},\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}\ (\lambda),$$

where $\mathsf{PR}^b_{\mathsf{Tag},\mathcal{A}}$ *and* $\mathsf{OneWay}_{\mathsf{Tag},\mathcal{A}}$ *are defined in Figure 5.*

4.2 Scheme Description

Let $\beta \in \mathbb{N}$. Let PKE = (PKESetup, KGen, Enc, Dec) be a (labeled) public-key encryption scheme, HC = (HCSetup, Com) be a homomorphic commitment scheme with message space $(\mathcal{M}, +)$ where $\{0, 1, \ldots, 2^{\alpha+\beta} - 1\} \subseteq \mathcal{M}$ and randomness space $(\rho, +)$, Tag = (TagSetup, TagEval) be a tagging scheme with secret key space space $(\chi, +)$, public key space $(\mathcal{X}, +)$, and tag space ψ, and SoK = (SoKSetup, SoKSig, SoKVf) be a signature of knowledge scheme for the language $\mathcal{L}[\mathsf{pp}_{\mathsf{HC}}, \mathsf{pp}_{\mathsf{Tag}}]$ (parameterized by the public parameters of HC and Tag) to be defined below. Let $\mathsf{H} : \{0,1\}^* \to \chi$ be a hash function modeled as a random oracle. We give a generic construction Ω of RingCT in Figure 6. An overview of the construction is as follows.

Recall that given the sets $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$, and a message μ, where

$$\mathcal{R} = \left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|} = \left\{(\mathsf{pk}_i^{\mathcal{R}}, \mathsf{co}_i^{\mathcal{R}}, \tilde{\mathsf{ek}}_i^{\mathcal{R}}, \tilde{\mathsf{ck}}_i^{\mathcal{R}})\right\}_{i=1}^{|\mathcal{R}|},$$

$$\mathcal{S} = \left\{(j_i, \mathsf{ck}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{sk}_i, \mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|} = \left\{(j_i, r_i^{\mathcal{S}}, a_i^{\mathcal{S}}, x_i, \mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|},$$

$$\mathcal{T} = \left\{(\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}}\right\}_{i=1}^{|\mathcal{T}|} = \left\{(r_i^{\mathcal{T}}, a_i^{\mathcal{T}}, (\mathsf{pk}_i^{\mathcal{T}}, \mathsf{co}_i^{\mathcal{T}}, \tilde{\mathsf{ek}}_i^{\mathcal{T}}, \tilde{\mathsf{ck}}_i^{\mathcal{T}}))\right\}_{i=1}^{|\mathcal{T}|},$$

we defined the corresponding transaction to be

$$\mathsf{tx}(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu) := \left(\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \left\{\left(\mathsf{acc}_i^{\mathcal{T}}\right)\right\}_{i=1}^{|\mathcal{T}|}, \mu\right).$$

Given $\mathsf{tx} = \mathsf{tx}(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$, we further define the statement

$$\mathsf{stmt}(\mathsf{tx}) := \left(\left\{\mathsf{pk}_i^{\mathcal{R}}, \mathsf{co}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \left\{\mathsf{co}_i^{\mathcal{T}}\right\}_{i=1}^{|\mathcal{T}|}\right).$$

The witness to the above statement is defined as

$$\mathsf{wit}(\mathcal{S}, \mathcal{T}) := \left(\left\{(j_i, x_i, a_i^{\mathcal{S}}, r_i^{\mathcal{S}})\right\}_{i=1}^{|\mathcal{S}|}, \left\{(a_i^{\mathcal{T}}, r_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|}\right).$$

Setup. The setup algorithm generates and outputs the public parameters for all the underlying primitives.

Stealth Address Generation. The master public key mpk consists of the two PKE public keys tpk and vpk, and a commitment $X = \mathsf{TagKGen}(x)$ to 0 with randomness x. The master secret key msk consists of the two PKE secret keys tsk and vsk, and the value x.

One-Time Account Generation. The algorithm commits to the amount a as $\mathsf{co} := \mathsf{Com}(a;r)$ with some randomness r. It then generates a random bit-string as an *ephemeral key* ek and hashes it with mpk to get a random exponent $s \in \chi$. A one-time public key pk is then derived as $\mathsf{pk} = X \cdot \mathsf{TagKGen}(s)$. Next it encrypts ek and ck using the appropriate instances of PKE and obtains $\tilde{\mathsf{ek}}$ and $\tilde{\mathsf{ck}}$ respectively as ciphertexts. These four elements are assembled to the *account* $\mathsf{acc} = (\mathsf{pk}, \mathsf{co}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$ and output together with the coin key, *i.e.*, the randomness r.

Receiving. The algorithm decrypts both ciphertexts $\tilde{\mathsf{ek}}$ and $\tilde{\mathsf{ck}}$ in $\mathsf{acc} = (\mathsf{pk}, \mathsf{co}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$ to obtain ek and (a, r), checks if $\mathsf{co} = \mathsf{Com}(a;r)$, derives the (one-time) secret key of the account as $x' = x + s$, and checks if $\mathsf{pk} = \mathsf{TagKGen}(x')$. It also generates the *tag* of the account as $\mathsf{tag} := \mathsf{TagEval}(x')$.

Spending. The algorithm derives the transaction tx, the statement stmt and the witness wit, and creates a signature of knowledge of the statement $\mathsf{stmt} \in \mathcal{L}[\mathsf{pp}_{\mathsf{HC}}, \mathsf{pp}_{\mathsf{Tag}}]$ with message tx, where

$$\mathcal{L}[\mathsf{pp_{HC}},\mathsf{pp_{Tag}}] := \left\{ \begin{array}{l} \mathsf{stmt} = \left(\{\mathsf{pk}_i^{\mathcal{R}},\mathsf{co}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{co}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|} \right): \\ \exists \mathsf{wit} = \left(\left\{(j_i,x_i,a_i^{\mathcal{S}},r_i^{\mathcal{S}})\right\}_{i=1}^{|\mathcal{S}|}, \{(a_i^{\mathcal{T}},r_i^{\mathcal{T}})\}_{i=1}^{|\mathcal{T}|} \right) \ s.t. \\ \forall i \in [|\mathcal{S}|], \begin{cases} \mathsf{pk}_{j_i}^{\mathcal{R}} = \mathsf{TagKGen}(x_i) \\ \mathsf{co}_{j_i}^{\mathcal{R}} = \mathsf{Com}(a_i^{\mathcal{S}};r_i^{\mathcal{S}}) \\ \mathsf{tag}_i = \mathsf{TagEval}(x_i) \end{cases} \\ \forall i \in [|\mathcal{T}|], \begin{cases} \mathsf{co}_i^{\mathcal{T}} = \mathsf{Com}(a_i^{\mathcal{T}};r_i^{\mathcal{T}}) \\ a_i^{\mathcal{T}} \in \left\{0,\ldots,2^{\beta}-1\right\} \end{cases} \\ \sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} = \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}} \end{array} \right.$$

Verify. Given a transaction tx and a proof σ, the verifier derives the statement stmt from tx and verifies if σ is a valid signature of knowledge of stmt with message tx.

4.3 Analysis

The correctness of the construction is obvious. Below we state the security results.

Theorem 4.2 (Balance). *If* HC *is computationally binding, and* SoK *is extractable, then the construction* Ω *is balanced.*

Theorem 4.3 (Privacy). *If* HC *is perfectly hiding and computationally binding,* PKE *is IND-CCA-secure and key-private (IK-CCA [5]),* H *is modeled as a random oracle,* SoK *is simulatable, and* Tag *is related-input pseudorandom, then the construction* Ω *is private.*

Theorem 4.4 (Non-slanderability). *If* SoK *is extractable and simulatable,* Tag *is related-input one-way, and* H *is modeled as a random oracle, then the construction* Ω *is non-slanderable.*

In the full version [27] we further extend the construction to provide tracking and viewing features, and provide proofs of the above theorems with respect to the extended definitions.

4.4 Concrete Instantiation: Omniring ($\mho$)

We propose Omniring, the concrete instantiation $\mho$ of our generic construction Ω. We instantiate PKE with a (labeled variant of) ECIES [48], HC with the Pedersen commitment [41], and Tag with the pseudorandom function of Dodis and Yampolsky [16] in a non-black-box manner, which we denote by $\mathsf{Tag}_{\mho}$.

Concretely, let $\mathcal{G} = (\mathbb{G},q,G)$ be the description of a cyclic group $\mathbb{G}$ of prime order $q \geq 2^{\alpha+\beta}$ with generator G, where certain Diffie-Hellman-types assumptions hold (see Appendix A for details). Let $H \in \mathbb{G}$ be another random generator of $\mathbb{G}$. We set $\mathsf{pp_{HC}} := (\mathbb{G},q,G,H)$ which defines $\mathcal{M} := \mathbb{Z}_q$ and $\rho := \mathbb{Z}_q$. We also set $\mathsf{pp_{Tag}} := (\mathbb{G},q,G,H)$ so that the secret key, public key, and tag spaces of the tagging scheme Tag is $\chi := \mathbb{Z}_q^*$, $\mathcal{X} := \mathbb{G} \setminus \left\{G^0\right\}$ and $\psi := \mathbb{G}$ respectively. For $a,r \in \mathbb{Z}_q$, we define $\mathsf{Com}(a;r) := G^a H^r$. For $x \in \mathbb{Z}_q^*$, we define $\mathsf{TagKGen}(x) = H^x$ and $\mathsf{TagEval}(x) := G^{\frac{1}{x}}$. More details of these constructions can be found in Appendix B.

With the above choices of HC and Tag fixed, we introduce the following notation for describing the language $\mathcal{L}[\mathsf{pp_{HC}},\mathsf{pp_{Tag}}]$ more conveniently. Given the statement and witness

$$\mathsf{stmt} = \left(\left\{\mathsf{pk}_i^{\mathcal{R}},\mathsf{co}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \left\{\mathsf{co}_i^{\mathcal{T}}\right\}_{i=1}^{|\mathcal{T}|} \right),$$

$$\mathsf{wit} = \left(\left\{(j_i,x_i,a_i^{\mathcal{S}},r_i^{\mathcal{S}})\right\}_{i=1}^{|\mathcal{S}|}, \left\{(a_i^{\mathcal{T}},r_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|} \right),$$

we define the following notation:

$$\begin{array}{rlrl} \vec{\mathbf{R}} &:= (\mathsf{pk}_1^{\mathcal{R}},\ldots,\mathsf{pk}_{|\mathcal{R}|}^{\mathcal{R}}) & \vec{\mathbf{C}}_{\mathcal{R}} &:= (\mathsf{co}_1^{\mathcal{R}},\ldots,\mathsf{co}_{|\mathcal{R}|}^{\mathcal{R}}) \\ \vec{\mathbf{T}} &:= (\mathsf{tag}_1,\ldots,\mathsf{tag}_{|\mathcal{S}|}) & \vec{\mathbf{C}}_{\mathcal{T}} &:= (\mathsf{co}_1^{\mathcal{T}},\ldots,\mathsf{co}_{|\mathcal{T}|}^{\mathcal{T}}) \\ \vec{\mathbf{x}} &:= (x_1,\ldots,x_{|\mathcal{S}|}) & \vec{\mathbf{a}}^{\mathcal{S}} &:= (a_1^{\mathcal{S}},\ldots,a_{|\mathcal{S}|}^{\mathcal{S}}) \\ \vec{\mathbf{x}}^{\circ-1} &:= (x_1^{-1},\ldots,x_{|\mathcal{S}|}^{-1}) & \vec{\mathbf{a}}^{\mathcal{T}} &:= (a_1^{\mathcal{T}},\ldots,a_{|\mathcal{T}|}^{\mathcal{T}}) \\ \vec{\mathbf{r}}^{\mathcal{S}} &:= (r_1^{\mathcal{S}},\ldots,r_{|\mathcal{S}|}^{\mathcal{S}}) & \vec{\mathbf{r}}^{\mathcal{T}} &:= (r_1^{\mathcal{T}},\ldots,r_{|\mathcal{T}|}^{\mathcal{T}}). \end{array}$$

Furthermore, let $\vec{\mathbf{e}}_i$ be the $|\mathcal{R}|$-dimensional unit vector with 1 at the j_i-th position and 0 everywhere else, and let $\vec{\mathbf{b}}_i$ be the binary representation of $\vec{\mathbf{a}}_i^{\mathcal{T}}$. We define their concatenations as the matrices $\mathbf{E}$ and $\mathbf{B}$ respectively. That is, the i-th row of $\mathbf{E}$ and $\mathbf{B}$ are $\vec{\mathbf{e}}_i$ and $\vec{\mathbf{b}}_i$ respectively. We write the (row) vectorizations of $\mathbf{E}$ and $\mathbf{B}$ as

$$\mathrm{vec}(\mathbf{E}) := (\vec{\mathbf{e}}_1,\ldots,\vec{\mathbf{e}}_{|\mathcal{S}|}) \qquad \mathrm{vec}(\mathbf{B}) := (\vec{\mathbf{b}}_1,\ldots,\vec{\mathbf{b}}_{|\mathcal{T}|}).$$

The language becomes:

$$\mathcal{L}_{\mho}[\mathbb{G},q,G,H] := \left\{ \begin{array}{l} \mathsf{stmt} = \left(\vec{\mathbf{R}},\vec{\mathbf{C}}_{\mathcal{R}},\vec{\mathbf{T}},\vec{\mathbf{C}}_{\mathcal{T}}\right): \\ \exists \mathsf{wit} = \left(\mathbf{E},\vec{\mathbf{x}},\vec{\mathbf{a}}^{\mathcal{S}},\vec{\mathbf{r}}^{\mathcal{S}},\mathbf{B},\vec{\mathbf{a}}^{\mathcal{T}},\vec{\mathbf{r}}^{\mathcal{T}}\right) \ s.t. \\ \forall i \in [|\mathcal{S}|], \begin{cases} \vec{\mathbf{e}}_i \text{ is a unit vector of length } |\mathcal{R}| \\ \vec{\mathbf{R}}^{\vec{\mathbf{e}}_i} = H^{x_i} \\ \vec{\mathbf{C}}_{\mathcal{R}}^{\vec{\mathbf{e}}_i} = G^{a_i^{\mathcal{S}}} H^{r_i^{\mathcal{S}}} \\ \mathsf{tag}_i = G^{x_i^{-1}} \end{cases} \\ \forall i \in [|\mathcal{T}|], \begin{cases} \vec{\mathbf{b}}_i \text{ is the binary rep. of } a_i^{\mathcal{T}} \text{ of length } \beta \\ \mathsf{co}_i^{\mathcal{T}} = G^{a_i^{\mathcal{T}}} H^{r_i^{\mathcal{T}}} \end{cases} \\ \sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} = \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}} \end{array} \right.$$

Finally, we instantiate SoK for the language $\mathcal{L}[\mathbb{G},q,G,H]$ by applying the Fiat-Shamir transform [17] to the argument of knowledge scheme for $\mathcal{L}[\mathbb{G}, q, G, H]$ to be constructed in Section 5.

We remark that with the above instantiations, all public parameters can be generated using *public coins*, i.e., without trusted setup.

5 ARGUMENT OF KNOWLEDGE

Below we construct a logarithmic-round argument of knowledge scheme for the language $\mathcal{L}_{\mho}[\mathbb{G}, q, G, H]$. In the basic protocol described below, the total size of the messages sent by the prover is bounded by $O(|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|)$. We can then replace part of the protocol with the argument of knowledge for inner product relations $\mathcal{L}_{\mathsf{IP}}$ (defined below) of [10] as a black-box, in the same fashion of their range proof construction. This squashes the communication to $O(\log(|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|))$.

In the following, vectors (of integers and group elements) are always written as row vectors unless specified. The actual orientation of a vector in a matrix-vector product is implicit

Notation	*Description*
$\hat{\mathbf{Y}}=\hat{\mathbf{Y}}(u):=\vec{\mathbf{R}}\circ\vec{\mathbf{C}}^{\circ u}_{\mathcal{R}}$	Vector of compressed public keys and coins with randomness $u\in\mathbb{Z}_q$.
$\hat{\mathbf{e}}=\hat{\mathbf{e}}(v):=\vec{v}^{\,|\mathcal{S}|}\mathbf{E}$	Vector of compressed unit vectors with randomness $v\in\mathbb{Z}_q$.
$\hat{T}=\hat{T}(u,v):=\vec{\mathbf{T}}^{u^2\vec{v}^{|\mathcal{S}|}}$	Compressed tag with randomness $u,v\in\mathbb{Z}_q$.
$\xi=\xi(u,v):=-\langle\vec{v}^{\,|\mathcal{S}|},u\cdot\vec{\mathbf{a}}^{\mathcal{S}}+u^2\cdot\vec{\mathbf{x}}^{\circ-1}\rangle$	Compressed secrets with randomness $u,v\in\mathbb{Z}_q$.
$\eta=\eta(u,v):=-\langle\vec{v}^{\,|\mathcal{S}|},\vec{\mathbf{x}}+u\cdot\vec{\mathbf{r}}^{\mathcal{S}}\rangle$	Note that $(\xi,\eta,\hat{\mathbf{e}})$ satisfies $I=G^{\xi}H^{\eta}\hat{T}\hat{\mathbf{Y}}^{\hat{\mathbf{e}}}$.
$\vec{\mathbf{c}}_L,\vec{\mathbf{c}}_R$	Encoding of witness by honest prover dependent on u and v, see Figure 8.
$m=3+|\mathcal{R}|+|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|+3|\mathcal{S}|$	Length of $\vec{\mathbf{c}}_L$ and $\vec{\mathbf{c}}_R$.
$(\vec{\mathbf{v}}_0,\ldots,\vec{\mathbf{v}}_8,\vec{\mathbf{u}}_4)=(\vec{\mathbf{v}}_0,\ldots,\vec{\mathbf{v}}_8,\vec{\mathbf{u}}_4)(u,v,y)$	Constraint vectors parameterized by the randomness $u,v,y\in\mathbb{Z}_q$, see Figure 9.
$(\vec{\alpha},\vec{\beta},\vec{\delta},\vec{\theta},\vec{\zeta},\vec{\mu},\vec{\nu},\vec{\omega})$ $=(\vec{\alpha},\vec{\beta},\vec{\delta},\vec{\theta},\vec{\zeta},\vec{\mu},\vec{\nu},\vec{\omega})(u,v,y,z)$	Compressed constraint vectors parameterized by the randomness $u,v,y,z\in\mathbb{Z}_q$, see Figure 9 and Figure 10.
$\mathsf{EQ}=\mathsf{EQ}[\vec{\mathbf{a}}^{\mathcal{T}},u,v,y]$	System of equations parameterized by the amounts $\vec{\mathbf{a}}^{\mathcal{T}}$ and randomness $u,v,y\in\mathbb{Z}_q$, see Figure 11.

Table 1: Notation for signatures of knowledge construction.

$$
\begin{array}{llccccc}
\vec{\mathbf{c}}_L := (& \xi \parallel \eta \parallel 1 \parallel \hat{\mathbf{e}} \parallel & \mathrm{vec}(\mathbf{E}) & \parallel \; \mathrm{vec}(\mathbf{B}) & \parallel \vec{\mathbf{a}}^{\mathcal{S}} \parallel \vec{\mathbf{r}}^{\mathcal{S}} \parallel & \vec{\mathbf{x}} &) \\
\vec{\mathbf{c}}_R := (& \vec{0}^{3+|\mathcal{R}|} & \parallel \mathrm{vec}(\mathbf{E})-\vec{1}^{|\mathcal{R}||\mathcal{S}|} & \parallel \mathrm{vec}(\mathbf{B})-\vec{1}^{\beta|\mathcal{T}|} & \parallel \; \vec{0}^{2|\mathcal{S}|} & \parallel \vec{\mathbf{x}}^{\circ-1} &)
\end{array}
$$

Figure 8: Honest encoding of witness.

$$
\begin{bmatrix}\vec{\mathbf{v}}_0\\ \vec{\mathbf{v}}_1\\ \vec{\mathbf{v}}_2\\ \vec{\mathbf{v}}_3\\ \vec{\mathbf{v}}_4\\ \vec{\mathbf{v}}_5\\ \vec{\mathbf{v}}_6\\ \vec{\mathbf{v}}_7\\ \vec{\mathbf{v}}_8\\ \vec{\mathbf{u}}_4\end{bmatrix}
:=
\left[\begin{array}{ccccccccc}
\cdot & \cdot & \cdot & \cdot & \multicolumn{2}{c}{\vec{y}^{\,|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|}} & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vec{y}^{\,|\mathcal{S}|} \\
\cdot & \cdot & \cdot & \cdot & \cdot & \vec{y}^{\,|\mathcal{T}|}\otimes\vec{2}^{\beta} & \cdot & \cdot & \cdot \\
\cdot & \cdot & y^{|\mathcal{S}|} & \cdot & \vec{y}^{\,|\mathcal{S}|}\otimes\vec{1}^{|\mathcal{R}|} & \cdot & \cdot & \cdot & \cdot \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\cdot\vec{v}^{\,|\mathcal{S}|} & \cdot & \cdot \\
\cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\cdot\vec{v}^{\,|\mathcal{S}|} & \vec{v}^{\,|\mathcal{S}|} \\
\cdot & \cdot & \cdot & -\vec{y}^{\,|\mathcal{R}|} & \vec{v}^{\,|\mathcal{S}|}\otimes\vec{y}^{\,|\mathcal{R}|} & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \vec{1}^{|\mathcal{T}|}\otimes\vec{2}^{\beta} & -\vec{1}^{|\mathcal{S}|} & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \multicolumn{2}{c}{\vec{y}^{\,|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|}} & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & u^2\cdot\vec{v}^{\,|\mathcal{S}|}
\end{array}\right]
$$

Figure 9: Definitions of constraint vectors. (Dots mean zeros.)

$$
\vec{\theta}:=\sum_{i=0}^{1}z^i\cdot\vec{\mathbf{v}}_i \qquad \vec{\zeta}:=\sum_{i=2}^{7}z^i\cdot\vec{\mathbf{v}}_i \qquad \vec{\mu}:=\sum_{i=2}^{8}z^i\cdot\vec{\mathbf{v}}_i
$$

$$
\vec{\nu}:=z^8\cdot\vec{\mathbf{v}}_8 \qquad \vec{\omega}:=z^4\cdot\vec{\mathbf{u}}_4
$$

$$
\vec{\alpha}:=\vec{\theta}^{\circ-1}\circ(\vec{\omega}-\vec{\nu}) \qquad \vec{\beta}:=\vec{\theta}^{\circ-1}\circ\vec{\mu}
$$

$$
\delta:=z\cdot\langle\vec{1}^{|\mathcal{S}|},\vec{y}^{\,|\mathcal{S}|}\rangle+z^3\cdot\langle\vec{1}^{|\mathcal{S}|+1},\vec{y}^{\,|\mathcal{S}|+1}\rangle+\langle\vec{\alpha},\vec{\mu}\rangle+\langle\vec{1}^{m},\vec{\nu}\rangle
$$

Figure 10: Definitions of constraint vectors (cont.).

$$
\mathsf{EQ}(\vec{\gamma}_L,\vec{\gamma}_R)=0 \iff
\left\{\begin{array}{lll}
\langle\vec{\gamma}_L,\vec{\gamma}_R\circ\vec{\mathbf{v}}_0\rangle & =0 & (1)\\
\langle\vec{\gamma}_L,\vec{\gamma}_R\circ\vec{\mathbf{v}}_1\rangle & =\langle\vec{1}^{|\mathcal{S}|},\vec{y}^{\,|\mathcal{S}|}\rangle & (2)\\
\langle\vec{\gamma}_L,\vec{\mathbf{v}}_2\rangle & =\langle\vec{\mathbf{a}}^{\mathcal{T}},\vec{y}^{\,|\mathcal{T}|}\rangle & (3)\\
\langle\vec{\gamma}_L,\vec{\mathbf{v}}_3\rangle & =\langle\vec{1}^{|\mathcal{S}|+1},\vec{y}^{\,|\mathcal{S}|+1}\rangle & (4)\\
\langle\vec{\gamma}_L,\vec{\mathbf{v}}_4\rangle+\langle\vec{\gamma}_R,\vec{\mathbf{u}}_4\rangle & =0 & (5)\\
\langle\vec{\gamma}_L,\vec{\mathbf{v}}_5\rangle & =0 & (6)\\
\langle\vec{\gamma}_L,\vec{\mathbf{v}}_6\rangle & =0 & (7)\\
\langle\vec{\gamma}_L,\vec{\mathbf{v}}_7\rangle & =0 & (8)\\
\langle\vec{\gamma}_L-\vec{\gamma}_R-\vec{1}^{m},\vec{\mathbf{v}}_8\rangle & =0 & (9)
\end{array}\right.
$$

Figure 11: System of equations guaranteeing the integrity of the encoding of witness.

and is not specified unless there is an ambiguity. We use "$\|$" as an operator for concatenating vectors. Inner products are denoted by $\langle \vec{\mathbf{u}}, \vec{\mathbf{v}} \rangle = \sum_{i=1}^{n} u_i v_i$. Let $\vec{\mathbf{G}} \in \mathbb{G}^n$ be a vector of group elements, and $x \in \mathbb{Z}$. We define the following operations between vectors of group elements and (vectors of) integers:

- Hadamard Powers: $\vec{\mathbf{G}}^{\circ x} := (g_1^x, \ldots, g_n^x)$
- Hadamard Products: $\vec{\mathbf{G}}^{\vec{\mathbf{v}}} := (g_1^{v_1}, \ldots, g_n^{v_n})$

We also define the operations between (vectors of) integers:

- Hadamard Products: $\vec{\mathbf{u}} \circ \vec{\mathbf{v}} := (u_1 v_1, \ldots, u_n v_n)$
- Kronecker Products: $\vec{\mathbf{u}} \otimes \vec{\mathbf{v}} := (u_1 \vec{\mathbf{v}}, \ldots, u_n \vec{\mathbf{v}})$
- Consecutive Powers: $\vec{x}^{n} = (1, x, \ldots, x^{n-1})$

Given a matrix $E \in \mathbb{Z}_q^{m \times n}$, its (row) vectorization is defined as $\mathrm{vec}(\mathbf{E}) := (\vec{\mathbf{e}}_1, \ldots, \vec{\mathbf{e}}_m)$, where $\vec{\mathbf{e}}_i$ is the i-th row of E. Conversely, we write $\mathbf{E} = \mathrm{vec}^{-1}(\vec{\mathbf{e}}_1, \ldots, \vec{\mathbf{e}}_m)$.

5.1 Our Basic Protocol

Below we describe our basic protocol $\Pi_{\mho} = (\mathsf{Setup}, \langle \mathcal{P}, \mathcal{V} \rangle)$ and state its security properties. The notation used within is defined in Section 4.4, Figures 8 to 10 and Table 1.

```
Setup(1^λ, $\mathcal{L}_\mho$):
Recall that $\mathcal{L}_\mho$ is specified by a tuple $(\mathbb{G}, q, G, H)$.
Output $\mathsf{crs} = (\mathbb{G}, q, G, H)$.
```

```
$\langle \mathcal{P}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle$:
$\mathcal{V}$:
    (1) $u, v \leftarrow_\$ \mathbb{Z}_q$
    (2) $F \leftarrow_\$ \mathbb{G}$, $\vec{\mathbf{P}} \leftarrow_\$ \mathbb{G}^{3+|\mathcal{R}|}$, $\vec{\mathbf{G}}' \leftarrow_\$ \mathbb{G}^{m-|\mathcal{R}|-3}$,
        $\vec{\mathbf{H}} \leftarrow_\$ \mathbb{G}^m$
$\mathcal{P} \leftarrow \mathcal{V} : u, v, F, \vec{\mathbf{P}}, \vec{\mathbf{G}}', \vec{\mathbf{H}}$
$\mathcal{P}, \mathcal{V}$:
    (1) $\hat{\mathbf{Y}} := \vec{\mathbf{R}} \circ \vec{\mathbf{C}}_{\mathcal{R}}^{\circ u}$
    (2) $\hat{T} := \vec{\mathbf{T}}^{u^2 \vec{v}^{|\mathcal{S}|}}$
    (3) For $w \in \mathbb{Z}_q$, denote
            $\vec{\mathbf{G}}_w := ((G \| H \| \hat{T} \| \hat{\mathbf{Y}})^{\circ w} \circ \vec{\mathbf{P}} \| \vec{\mathbf{G}}')$        (10)
$\mathcal{P}$:
    (1) $r_A \leftarrow_\$ \mathbb{Z}_q$
    (2) $A := F^{r_A} \vec{\mathbf{G}}_0^{\vec{\mathbf{c}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{c}}_R}$
        Note that $\vec{\mathbf{G}}_w^{\vec{\mathbf{c}}_L} = \vec{\mathbf{G}}_{w'}^{\vec{\mathbf{c}}_L}$ for all $w, w' \in \mathbb{Z}_q$ since $I =$
        $G^{\xi} H^{\eta} \hat{T} \hat{\mathbf{Y}}^{\hat{\mathbf{e}}}$. Thus $A = F^{r_A} \vec{\mathbf{G}}_w^{\vec{\mathbf{c}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{c}}_R}$ for all $w \in \mathbb{Z}_q$.
$\mathcal{P} \rightarrow \mathcal{V} : A$
$\mathcal{V} : w \leftarrow_\$ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V} : w$
$\mathcal{P}$:
    (1) $r_S \leftarrow_\$ \mathbb{Z}_q$, $\vec{\mathbf{s}}_L, \vec{\mathbf{s}}_R \leftarrow_\$ \mathbb{Z}_q^m$
    (2) $S := F^{r_S} \vec{\mathbf{G}}_w^{\vec{\mathbf{s}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{s}}_R}$.
$\mathcal{P} \rightarrow \mathcal{V} : S$
$\mathcal{V} : y, z \leftarrow_\$ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V} : y, z$
$\mathcal{P}$:
    (1) Define the following polynomials (in $X$):
            $l(X) := \vec{\mathbf{c}}_L + \vec{\alpha} + \vec{\mathbf{s}}_L \cdot X$
            $r(X) := \vec{\theta} \circ (\vec{\mathbf{c}}_R + \vec{\mathbf{s}}_R \cdot X) + \vec{\mu}$
            $t(X) := \langle l(X), r(X) \rangle = t_2 X^2 + t_1 X + t_0$
        for some $t_0, t_1, t_2 \in \mathbb{Z}_q$. In particular
            $t_0 = z^2 \cdot \langle \vec{\mathbf{a}}^{\mathcal{T}}, \vec{y}^{|\mathcal{T}|} \rangle + \delta$
    (2) $\tau_1, \tau_2 \leftarrow_\$ \mathbb{Z}_q$
    (3) $T_1 := G^{t_1} F^{\tau_1}$, $T_2 := G^{t_2} F^{\tau_2}$
$\mathcal{P} \rightarrow \mathcal{V} : T_1, T_2$
$\mathcal{V} : x \leftarrow_\$ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V} : x$
$\mathcal{P}$:
    (1) $\tau := z^2 \cdot \langle \vec{\mathbf{r}}^{\mathcal{T}}, \vec{y}^{|\mathcal{T}|} \rangle + \tau_1 x + \tau_2 x^2$
    (2) $r := r_A + r_S x$
    (3) $(\vec{l}, \vec{r}, t) := (l(x), r(x), t(x))$
$\mathcal{P} \rightarrow \mathcal{V} : \tau, r, \vec{l}, \vec{r}, t$
$\mathcal{V}$: Check if the following relations hold:
            $t = \langle \vec{l}, \vec{r} \rangle$        (11)
            $F^r \vec{\mathbf{G}}_w^{\vec{l}} \vec{\mathbf{H}}^{\vec{\theta}^{\circ -1} \circ \vec{r}} = A S^x \vec{\mathbf{G}}_w^{\vec{\alpha}} \vec{\mathbf{H}}^{\vec{\beta}}$        (12)
            $G^t H^{\tau} = G^{\delta} \vec{\mathbf{C}}_{\mathcal{T}}^{z^2 \cdot \vec{y}^{|\mathcal{T}|}} T_1^x T_2^{x^2}$        (13)
```

THEOREM 5.1. *The verifier $\mathcal{V}$ is public-coin. $\Pi_{\mho}$ is constant-round, perfectly complete, and perfect special honest-verifier zero-knowledge.*

THEOREM 5.2. *Assuming the discrete logarithm assumption holds over $\mathcal{G}$, $\Pi_{\mho}$ has computational witness-extended emulation.*

The proofs of the above theorems can be found in the full version [27].

5.2 Inner Product Argument

We next recall the argument of knowledge for inner product relations in [10]. Formally, given a group description $\mathcal{G} = (\mathbb{G}, q, G)$, an integer $m \in \mathbb{N}$, and two vectors of group elements $\vec{\mathbf{G}}, \vec{\mathbf{H}} \in \mathbb{G}^m$, we define the following inner product relation:

$$\mathcal{L}_{\mathsf{IP}}[\vec{\mathbf{G}}, \vec{\mathbf{H}}] := \left\{ \begin{array}{c} (P, t) \in \mathbb{G} \times \mathbb{Z}_q : \\ \exists \vec{l}, \vec{r} \in \mathbb{Z}_q^m \ s.t. \ P = \vec{\mathbf{G}}^{\vec{l}} \vec{\mathbf{H}}^{\vec{r}} \wedge \langle \vec{l}, \vec{r} \rangle = t \end{array} \right.$$

We denote the argument of knowledge protocol of [10] by Π_{IP}.

It is shown that if finding a non-trivial discrete logarithm representation of the identity element in $\mathbb{G}$ with base $\vec{\mathbf{G}} \| \vec{\mathbf{H}}$ is hard, then Π_{IP} has computational witness-extended emulation [10]. In their security proof, it is implicitly assumed that $\vec{\mathbf{G}}$ and $\vec{\mathbf{H}}$ are uniformly sampled from $\mathbb{G}^m$. For our purpose, we require a slightly stronger theorem which states that the argument of knowledge has computational witness-extended emulation even if the adversary has certain control over the values of $\vec{\mathbf{G}}$ and $\vec{\mathbf{H}}$.

THEOREM 5.3 (INFORMAL, MODIFIED FROM [10]). *Let $\vec{\mathbf{G}}$ and $\vec{\mathbf{H}}$ be sampled in such a way that it is hard to find $\vec{\mathbf{a}}, \vec{\mathbf{b}}$ with $I = \vec{\mathbf{G}}^{\vec{\mathbf{a}}} \vec{\mathbf{H}}^{\vec{\mathbf{b}}}$. Then Π_{IP} has computational witness-extended emulation.*

For the formal theorem we refer to the full version [27]. Its proof is almost identical to that given in [10] and is thus omitted.

5.3 Squashing Prover Communication

To squash the prover communication in our basic protocol from linear to logarithmic, we modify the protocol as follows. In the last message that $\mathcal{P}$ sends to $\mathcal{V}$, instead of sending $\vec{l}$ and $\vec{r}$ in plain, $\mathcal{P}$ commits to them as $P = \vec{\mathbf{G}}_w^{\vec{l}} \vec{\mathbf{H}}^{\vec{\theta}^{\circ -1} \circ \vec{r}}$, where $\vec{\mathbf{G}}_w$ is defined in Equation (10), and sends P to $\mathcal{V}$. Then $\mathcal{P}$ and $\mathcal{V}$ engage in the argument of knowledge Π_{IP} for the inner product relation, which convinces $\mathcal{V}$ that P is computed correctly and indeed $t = \langle \vec{l}, \vec{r} \rangle$. Finally, $\mathcal{V}$ proceeds to check if $F^r P = AS^x \vec{\mathbf{G}}_w^{\vec{\alpha}} \vec{\mathbf{H}}^{\vec{\beta}}$ and $G^t H^\tau = G^\delta \vec{\mathbf{C}}_{\mathcal{T}}^{z^2 \cdot \vec{y}^{|\mathcal{T}|}} T_1^x T_2^{x^2}$. As shown in [10], the following holds: $\mathcal{V}_{\mathsf{IP}}$ is public-coin; Π_{IP} has $\lceil \log_2 m \rceil$ rounds; and Π_{IP} is perfectly complete, perfect special honest-verifier zero-knowledge. As stated in Theorem 5.3, Π_{IP} has computational witness-extended emulation if finding non-trivial discrete logarithm relations among $\vec{\mathbf{G}}_w$ and $\vec{\mathbf{H}}$ is hard. Consequently, after the aforementioned changes, Theorem 5.1 and Theorem 5.2 still hold for the resulting protocol, except that the latter now consists of $\lceil \log_2 m \rceil + O(1)$ number of rounds.

6 OPTIMIZATIONS AND PERFORMANCE

We discuss several optimization techniques and compare the efficiency of Omniring with that of Monero.

6.1 Efficient Verification

An Omniring transaction is computationally efficient to verify, as it can be reduced to a single multi-exponentiation of size $2m + \log(m) + O(1)$ using the technique of [10], where $m = 3 + |\mathcal{R}| + |\mathcal{R}||\mathcal{S}| + \beta|\mathcal{T}| + 3|\mathcal{S}|$. Since multi-exponentiations can be computed more efficiently than performing an equivalent number of individual exponentiations, they enable large savings. As the required techniques are exactly the same as in [10], we refer the reader to Bünz *et al.* [10].

6.2 Log-size Transactions

While Omniring produces spend proofs of logarithmic size, the spender needs to communicate the set of destination accounts $\{\mathsf{acc}_i^{\mathcal{T}}, \mathsf{info}_i\}_{i=1}^{|\mathcal{T}|}$, the set of tags $\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}$, and a set of ring accounts $\{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}$ within the transaction to allow the cryptocurrency network to verify the transaction. Since $|\mathcal{T}|$ and $|\mathcal{S}|$ are typically small (we typically have $|\mathcal{T}| = 2$ and $|\mathcal{S}| < 5$), they can be safely neglected. However, the ring size $|\mathcal{R}|$ is a problem if a high level of privacy is desired. The obvious solution to include $|\mathcal{R}|$ ring members in the transaction needs $O(|\mathcal{R}|)$ space which quickly becomes impractical. However, using the *recovery sampling* technique by Chator and Green [12], which is built for this exact purpose, the description of the set of ring members can be as short as $O(\log|\mathcal{R}|)$, yielding a overall transaction size (not only proof size) logarithmic in m.

6.3 Performance Comparison

We compare the performance of Omniring with the RingCT scheme currently employed in Monero, *i.e.*, the scheme by Noether *et al.* [39] with a minor modification [23], together with Bulletproofs [10] range proofs (all of the range proofs in a transaction aggregated into a single Bulletproof). For conciseness we simply use Monero to refer to this scheme. We consider the typical case of $|\mathcal{T}| = 2$,[1] and the amount range $\beta = 64$ used in Monero. For a fair comparison we also consider only transactions with one source account ($|\mathcal{S}| = 1$), to exclude the advantages that our model of RingCT provides for $|\mathcal{S}| > 1$ (see Section 1.1.1).

Proof Size. In Figure 13 we compare the proof size of known RingCT schemes. We assume (non-pairing) elliptic curves as in Monero and therefore do not differentiate between group elements and scalars because they have roughly the same size. The proof size of Omniring is

$$2\lceil \log_2(3 + |\mathcal{R}| + |\mathcal{R}||\mathcal{S}| + \beta|\mathcal{T}| + 3|\mathcal{S}|) \rceil + 9,$$

while that of Monero is

$$(|\mathcal{R}| + 2)(|\mathcal{S}| + 1) + \log_2(|\mathcal{T}|\beta) + 9.$$

RingCT 3.0 proposed in a concurrent work [55] can be instantiated in the same setting, and has proof size

$$|\mathcal{S}|(2\lceil \log_2 |\mathcal{R}| \rceil + 17) + 2\lceil \log_2(\beta|\mathcal{T}|) \rceil + 2.$$

The proof size of RingCT 2.0 [51] is $O(|\mathcal{S}| + \log(\beta|\mathcal{T}|))$ elements where the hidden constant is in the hundreds. The concrete count is incomparable to other schemes and is omitted, as the scheme is based on pairing groups and has a trusted setup.

Figure 12a shows the number of elements in the proof against the size of the ring. Note that even when $|\mathcal{R}|$ is as small as 11, which is the ring size currently enforced in Monero, the proof size of Omniring is already significantly smaller than that of Monero, and for larger $|\mathcal{R}|$, the difference in proof size grows further. Finally, we remark that although our comparison only considers $|\mathcal{S}| = 1$ and $|\mathcal{T}| = 2$, for general $|\mathcal{S}|$ and $|\mathcal{T}|$, the gap in proof size would only be larger as the proof size of Monero scales linearly with $|\mathcal{S}|$ and $|\mathcal{T}|$ while Omniring's only scales logarithmically.

Running Time. To compare the running time, we make use of the fact that our spend algorithm is very similar in structure to a Bulletproofs range proof, which has been implemented in Monero; the only difference significant for performance is the size of the vectors in the inner product proof. By modifying the Monero benchmark suite to run Bulletproofs with larger vectors, we can obtain estimates for the running time of Omniring, and compare them with running times for the RingCT scheme used in Monero. Our estimates are suitable for a comparison with Monero because they rely on the same C++ implementation of Bulletproofs, i.e., on the same elliptic curve using the same optimizations. Our modified code is available online [13]. All experiments were run on a single core of an Intel Core i7-7600U (Kaby Lake) CPU with TurboBoost disabled to get more consistent results.

[1] A typical transaction uses one destination account to pay to the receiver and one *change account* to pay the remaining funds from the source accounts back to a new account of the sender. This model, introduced by Bitcoin, is common in cryptocurrencies and it is actually fundamental to RingCT because the spend proof only reveals that the sum of the source amounts equals the sum of the output amounts. Partial spends of a source account would require accounting for the exact amount that has been spent.

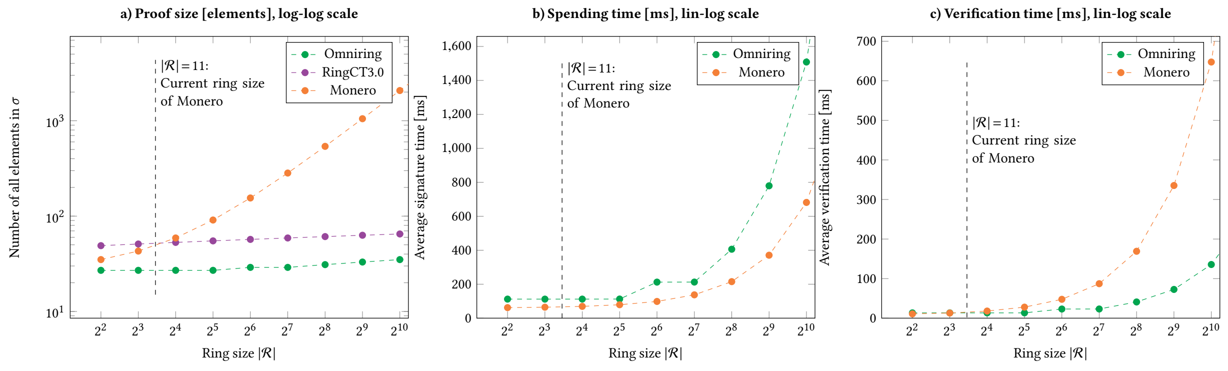

Figure 12: Performance comparison.

Scheme	Spend proof size (in elements)	Pairing	Trusted setup
Monero [23, 39] with Bulletproofs range proofs	$O(\|\mathcal{R}\|\|\mathcal{S}\|+\log(\beta\|\mathcal{T}\|))$	No	No
RingCT 2.0 [51] with Bulletproofs range proofs	$O(\|\mathcal{S}\|+\log(\beta\|\mathcal{T}\|))$	Yes	Yes
RingCT 3.0 [55] with Bulletproofs range proofs	$O(\|\mathcal{S}\|\log\|\mathcal{R}\|+\log(\beta\|\mathcal{T}\|))$	No	No
Omniring	$O(\log(\|\mathcal{R}\|\|\mathcal{S}\|+\beta\|\mathcal{T}\|))$	No	No

Figure 13: Comparison of RingCT schemes.

Figure 12b and Figure 12c show the estimated time needed for generating and verifying a proof, respectively, against the size of the ring. The verification time is particularly important as each proof on the blockchain needs to be verified by virtually all nodes in the cryptocurrency network. The time needed for generating proofs in Omniring is about twice of that in Monero. Omniring, however, has considerably faster verification than Monero does for higher $|\mathcal{R}|$. For instance, at $|\mathcal{R}| = 128$, verifying an Omniring transaction is 4 times faster than verifying a Monero transaction.

7 RELATED WORK

We start our discussion of related work by a brief historical overview.

7.1 Brief History of RingCT and Monero

To ensure anonymity in cryptocurrencies while still preventing double-spending, van Saberhagen mentioned the use of linkable ring signatures in a cryptocurrency called CryptoNote v2.0 [53], which ensures that messages signed by the same sender are linkable, independently of the rings or messages. The construction is a slight modification of the scheme by Liu, Wei, and Wong [29], however the security analysis is not detailed and is carried out with respect to informal definitions.

Back [3] observed how to improve CryptoNote v2.0 relying on ideas from [1]. Noether *et al.* [39] generalized Back's scheme using the name Ring Confidential Transactions (RingCT) to allow for batch spending with improved confidentiality (by using Confidential Transactions (CT)) and anonymity guarantees (Stealth Addresses). While the original proposal by Maxwell [32] leaves out many cryptographic details, which could just be found in the source code, the concept of CT for confidentiality of amounts is explained in detail by Gibson [19]. CT further has been partially formalized by Poelstra *et al.* [43], and fully formalized in the context of the Mimblewimble cryptocurrency design [22, 42] by Fuchsbauer *et al.* [18].

A variant [23] of the scheme by Noether *et al.* is used in Monero since its inception. In 2018 Monero switched to Bulletproofs range proofs to reduce the size of the spend proofs [49].

To our best knowledge, the concept of "stealth addresses" first appeared in CryptoNote v2.0 [53] in the name "public user keys" without formalization. Meiklejohn and Mercer [33] formalize stealth addresses by requiring that the one-time public keys are identically distributed as randomly chosen ones, in the view of external parties. This requirement is necessary but not sufficient in our application because the spender who derives the one-time public key may know extra information about the key which can be used to link signatures (hence breaking spender anonymity).

7.2 Comparison with RingCT 2.0

We compare our formalization and construction with "RingCT 2.0" by Sun *et al.* [51].

Formalizing RingCT. The formal model in RingCT 2.0 does not formalize the central property of stealth addresses nor receiver anonymity. While both our model and that of RingCT 2.0 define balance and spender anonymity, our definitions assume stronger adversaries. In the case of balance, we require the property to hold even if all accounts in the transaction are maliciously generated. In contrast, RingCT 2.0 considers only honestly generated accounts. While RingCT 2.0 only considers spender anonymity in the case where all source accounts are not corrupt, we allow some of the source accounts to be corrupt and still require the non-corrupt accounts to be anonymous.

Moreover, related to the support of stealth addresses, we allow the source accounts to be the target accounts in previous transactions created by the adversary. This naturally makes it non-trivial to define spender anonymity in contrast to previous works that only relied on a ring signature style anonymity definition. Therefore our model offers stronger security guarantees.

Non-slanderability $\not\Rightarrow$ Linkability. Sun *et al.* [51] claim that non-slanderability implies linkability. Since their claim is informal, it is unclear whether the implication is claimed just for RingCT systems or also for linkable signatures. We show that either case is not true by giving an intuition for constructing counter examples. Consider a RingCT or a linkable ring signature scheme where the tag is a commitment of the signer secret key. A signature consists of a proof where, among other relations, the tag is a commitment of a secret key, and the public key corresponding to the secret key is a member of the ring. It is clear that such a construction can be made unforgeable and anonymous (and balanced in the case of RingCT) when instantiated with appropriate proof system and commitment scheme. In particular, suppose the proof system is perfectly zero-knowledge and is a PoK, and the commitment scheme is perfectly hiding. In this case, we observe that the scheme can be made non-slanderable yet non linkable, falsifying the claim in RingCT 2.0 that non-slanderability implies linkability.

Obviously, the scheme is not linkable since the tag is completely independent of the secret key. Intuitively, the scheme is non-slanderable, since given a tag (a commitment of some secret key sk) along with a proof about it, sk is information-theoretically hidden from the adversary due to the perfectly zero-knowledge and perfectly hiding property. If the adversary manages to produce a proof that the tag is a commitment of a secret key sk', with overwhelming probability we would have $\mathsf{sk} \neq \mathsf{sk}'$. Since the proof system is a proof of knowledge, we can extract sk' and thus break the binding property of the commitment scheme. We remark that the RingCT 2.0 construction of Sun *et al.* [51] is nevertheless linkable since a tag is uniquely determined by an account.

Constructing RingCT. The RingCT 2.0 scheme is an accumulator-based construction which features a signature size independent of $|\mathcal{T}|$ and $|\mathcal{R}|$. However, this scheme relies on a trusted setup and a pairing-friendly elliptic curve over which operations are computationally more expensive than non-pairing-friendly ones. While setup-free accumulators are known based on unknown order groups, they require considerably larger parameters to be secure. Moreover, the RingCT 2.0 construction does not support stealth addresses.

7.3 Comparison with RingCT 3.0

A very recent concurrent work by Yuen *et al.* [55] proposes "RingCT 3.0", which uses a syntax similar to ours and improves upon RingCT 2.0 mainly by supporting stealth addresses and getting rid of the trusted setup.

Formalizing RingCT. Regarding their security model, RingCT 3.0 suggests a more restricted definition of balance that forces the adversary to generate its transactions using oracles provided by the experiment. This is necessary to learn the amounts corresponding to the adversarial transaction by witnessing the oracle queries. We believe that this notion is too restrictive because it does not cover adversaries that simply forge proofs of false statements and do not use the oracle at all. Our approach is different and more general, allowing arbitrary adversaries and requiring only the existence of an extractor which extracts the amounts by running code of the adversary.

The second weakness in their formal security model is their definition of anonymity, which is split into *anonymity against receivers* and *anonymity against ring insiders.* First, the two properties together do not seem to imply the combined property, *i.e.*, anonymity against a coalition of receivers and ring members. Second, their definition assumes honestly generated source accounts. We believe that this notion is too weak because it rules out natural real-world attacks where a curious user, who has transferred some money to a one-time account of the victim, tries to determine if the victim's account is used in a given transaction. Third, their definition only covers spender but not receiver anonymity.

Constructing RingCT. Aside from the definitional issues, the construction seems also less efficient compared to our "unified ring" construction because RingCT 3.0 requires separate rings for separate source accounts like all other previous RingCT schemes. Moreover, range proofs are not directly integrated in their construction. Instead, for real-world applications, one would need to compose their construction with a separate range proof system. While this is acceptable from a theoretical point of view, it incurs unnecessary computational and communication overheads which impact concrete efficiency.

7.4 Comparison with Zerocoin and Zerocash

Zerocoin [36] and Zerocash [6] are designs for cryptocurrencies aiming to provide anonymity and the privacy of amounts based on zero-knowledge proofs. Ring signatures in CryptoNote v2.0 [53] (the underlying scheme of Monero) serve the same purpose as the non-interactive zero-knowledge proofs in Zerocoin. In fact, the zero-knowledge proofs can be seen as a form of ring signatures. Zerocoin was developed as an extension to Bitcoin, and Zerocash is designed as an independent currency and has been implemented in Zcash [56]. Both Zerocoin and Zerocash use a trusted setup, and Zerocash uses zero-knowledge succinct non-interactive arguments of knowledge (ZK-SNARKs) to prove the integrity of computations. Due to the trusted setup, the system per se cannot be considered as completely decentralized. By using cryptographic accumulators which give constant-size membership proofs, Zerocoin and Zerocash can handle very large anonymity sets; the anonymity set is the set of all coins ever created. In contrast, Monero scales only to medium-size anonymity sets but does not require a trusted setup; the same is true for another instantiation of Zerocoin proposed by Groth and Kohlweiss [20].

ACKNOWLEDGMENTS

We thank the Monero Research Lab and the anonymous referees for their valuable comments and helpful suggestions.

This work is supported by the German Research Foundation under Grant No.: 272573906 and 393541319, by the German

Academic Exchange Service under Grant No.: PPP-HK 57391915, and by the University Grants Committee of Hong Kong under Grant No.: G-CUHK406/17, CUHK 14210217, and CUHK 14209918. This work is also supported by the state of Bavaria at the Nuremberg Campus of Technology (NCT). NCT is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN).

A COMPUTATIONAL ASSUMPTIONS

Let $\mathcal{G} = (\mathbb{G}, q, G)$ be the description of a cyclic group $\mathbb{G}$ of prime order q with generator G. In the following, we recall the (general) discrete logarithm assumption, the strong decisional Diffie-Hellman inversion (SDDHI) assumption, and the Gap Diffie-Hellman assumption.

Definition A.1 (General Discrete Logarithm Assumption (GDL)). *We say that the general logarithm assumption holds over $\mathcal{G}$ if for all $\ell \in \mathsf{poly}(\lambda)$, every* PPT *adversary $\mathcal{A}$*

$$\Pr[\ell\text{-}\mathsf{DL}_{\mathcal{A}}(\mathcal{G})] \leq \mathsf{negl}(\lambda),$$

where the game ℓ-$\mathsf{DL}_{\mathcal{A}}(\mathcal{G})$ is defined in Figure 14.

```
$\ell\text{-DL}_{\mathcal{A}}(\mathcal{G})$
$x \leftarrow_\$ \mathbb{Z}_q$
$x' \leftarrow \mathcal{A}(\mathbb{G}, q, G, G^x, G^{x^2}, \ldots, G^{x^\ell})$
$b := (G^x = G^{x'})$
return $b$
```

Figure 14: Security game for GDL.

Definition A.2 (Gap Diffie-Hellman Assumption (GapDH) [40]). *We say that the GapDH assumption holds over $\mathcal{G}$ if for all $\ell \in \mathsf{poly}(\lambda)$, every* PPT *adversary $\mathcal{A}$*

$$\Pr\left[\mathsf{GapDH}_{\mathcal{A}}(\mathcal{G}) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{GapDH}_{\mathcal{A}}(\mathcal{G})$ is defined in Figure 15.

```
$\mathsf{GapDH}_{\mathcal{A}}(\mathcal{G})$
$x \leftarrow_\$ \mathbb{Z}_q^*; y \leftarrow_\$ \mathbb{Z}_q^*$
$H \leftarrow \mathcal{A}^{O_{\mathsf{DDH}}}(\mathbb{G}, q, G, G^x, G^y)$
return $(H = G^{xy})$

$O_{\mathsf{DDH}}(A,B,C)$
$a := \log_G(A); b := \log_G(B)$
$c := \log_G(C)$
return $(G^{ab} = G^c)$
```

Figure 15: Security game for GapDH.

Definition A.3 (Strong Decisional Diffie-Hellman Inversion Assumption (SDDHI) [11]). *We say that the SDDHI assumption holds over $\mathcal{G}$ if for all $\ell \in \mathsf{poly}(\lambda)$, every* PPT *adversary $\mathcal{A}$*

$$\left|\Pr\left[\mathsf{SDDHI}^0_{\mathcal{A}}(\mathcal{G}) = 1\right] - \Pr\left[\mathsf{SDDHI}^1_{\mathcal{A}}(\mathcal{G}) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where the game $\mathsf{SDDHI}_{\mathcal{A}}(\mathcal{G})$ is defined in Figure 16.

```
$\mathsf{SDDHI}^b_{\mathcal{A}}(\mathcal{G})$
$Z := \emptyset$
$x \leftarrow_\$ \mathbb{Z}_q^*$
$s \leftarrow \mathcal{A}_0^{O_x}(\mathbb{G}, q, G, G^x)$
$b \leftarrow_\$ \{0,1\}$
$y_0 = G^{\frac{1}{x+s}}$
$y_1 \leftarrow_\$ \mathbb{G}$
$b' \leftarrow \mathcal{A}_1^{O_x}(y_b)$
if $s \in Z$ then return $0$
return $b'$

$O_x(z)$
$Z := Z \cup \{z\}$
return $G^{\frac{1}{x+z}}$
```

Figure 16: Security game for SDDHI.

B DETAILED INSTANTIATION

We describe the details of the instantiation mentioned in Section 4.4.

B.1 Tag Function

We instantiate the tagging scheme $\mathsf{Tag} = (\mathsf{TagSetup}, \mathsf{TagKGen}, \mathsf{TagEval})$ as $(\mathbb{G}, q, G, H) \leftarrow \mathsf{TagSetup}(1^\lambda)$, $\mathsf{TagKGen}(x) := H^x$, and $\mathsf{TagEval}(x) := G^{\frac{1}{x}}$.

Theorem B.1. *If the general discrete logarithm assumption and the strong decisional Diffie-Hellman inversion assumption hold over $\mathcal{G} = (\mathbb{G}, q, G)$, then the tagging scheme* Tag *instantiated as above is a secure tagging scheme according to Definition 4.1.*

B.2 Homomorphic Commitments

The Pedersen commitment [41] can commit to a vector of messages $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_q^n$ by picking group elements $G_1, \ldots, G_n \leftarrow \mathbb{G}$ and computing $\mathsf{Com}_{\mathsf{crs}}(\mathbf{m}; r) := H^r \prod_{i=1}^n G_i^{m_i}$. The Pedersen commitment is naturally homomorphic. This commitment scheme is perfectly hiding, and it is computationally binding under the discrete logarithm assumption.

B.3 Labeled Encryption

The Elliptic Curve Integrated Encryption Scheme (ECIES) [48] is a practical hybrid encryption scheme on elliptic curves. We provide below an abstract description of a labeled variant over generic groups. Below, let $H : \{0,1\}^* \rightarrow \{0,1\}^{2\lambda}$ be a hash function modeled as a random oracle, and let SKE and MAC be a symmetric key encryption scheme and a message authentication code scheme respectively both with key space $\{0,1\}^\lambda$.

$\underline{\mathsf{Setup}(1^\lambda)}$: On input the security parameter 1^λ, it outputs public parameters pp consisting of a description of a cyclic group $\mathbb{G}$ of order q together with a group elements $G \in \mathbb{G}$.

$\underline{\mathsf{KGen}(\mathsf{pp})}$: On input the public parameters pp, the algorithm samples $x \leftarrow \mathbb{Z}_q$ and computes $H := G^x$. Then H is the public key pk and x is the secret key sk.

$\underline{\mathsf{Enc}(\mathsf{pk}, \tau, m)}$: On input the public key pk, a label τ, and a message $m \in \mathbb{G}$, the algorithm samples $r \leftarrow \mathbb{Z}_q$ and computes $R := G^r$, $P := H^r$, $(\mathsf{sk}_{\mathsf{SKE}}, \mathsf{sk}_{\mathsf{MAC}}) := H(P, \tau)$, $e \leftarrow \mathsf{SKE.Enc}(\mathsf{sk}_{\mathsf{SKE}}, m)$, and $\sigma \leftarrow \mathsf{MAC.Sig}(\mathsf{sk}_{\mathsf{MAC}}, e)$. The ciphertext is then $c := (R, e, \sigma)$.

Dec(sk,τ,c): On input the secret key sk, a label τ, and a ciphertext c, the algorithm computes $P := R^x$ and $(\mathsf{sk}_{\mathsf{SKE}},\mathsf{sk}_{\mathsf{MAC}}) := H(P,\tau)$. It checks if $\mathsf{MAC.Vf}(\mathsf{sk}_{\mathsf{MAC}},e,\sigma) = 1$. If so, it outputs $m := \mathsf{SKE.Dec}(\mathsf{sk}_{\mathsf{SKE}},e)$. Otherwise, it outputs $\perp$.

It is well-known that if the GapDH assumption holds in $\mathbb{G}$, SKE is IND-CPA and MAC is a strongly unforgeable MAC, then ECIES is IND-CCA in the random oracle model [15, 48]; this can easily be extended to key-privacy (IK-CCA [5]).

REFERENCES

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 2002. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT'02*.

[2] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2013. Evaluating user privacy in Bitcoin. In *FC'13*.

[3] Adam Back. 2015. Ring signature efficiency. Post in Bitcoin forum. https://bitcointalk.org/index.php?topic=972541.msg10619684.

[4] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. 2012. Bitter to better. How to make Bitcoin a better currency. In *FC'12*.

[5] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. 2001. Key-privacy in public-key encryption. In *ASIACRYPT'01*.

[6] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: decentralized anonymous payments from Bitcoin. In *S&P'14*.

[7] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. 2014. Mixcoin: anonymity for Bitcoin with accountable mixes. In *FC'14*.

[8] Jonathan Bootle and Jens Groth. 2018. Efficient batch zero-knowledge arguments for low degree polynomials. In *PKC'18*.

[9] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. 2015. Short accountable ring signatures based on DDH. In *ESORICS'15*.

[10] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: short proofs for confidential transactions and more. In *S&P'18*.

[11] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. 2006. How to win the clonewars: efficient periodic n-times anonymous authentication. In *CCS'06*.

[12] Alishah Chator and Matthew Green. 2018. How to squeeze a crowd: reducing bandwidth in mixing cryptocurrencies. In *S&B'18*.

[13] Benchmark code for this paper. 2019. https://github.com/real-or-random/monero/commit/152e545558e52bec65ce68e58b5c69258c824d0e.

[14] dEBRUYNE. 2018. A post mortem of the burning bug. https://getmonero.org/2018/09/25/a-post-mortum-of-the-burning-bug.html.

[15] Alexander W. Dent. 2002. ECIES-KEM vs. PSEC-KEM. Technical report. NES/DOC/RHU/WP5/028/2. https://www.cosic.esat.kuleuven.be/nessie/reports/phase2/evalv2.pdf.

[16] Yevgeniy Dodis and Aleksandr Yampolskiy. 2005. A verifiable random function with short proofs and keys. In *PKC'05*.

[17] Amos Fiat and Adi Shamir. 1987. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*.

[18] Georg Fuchsbauer, Michele Orrù, and Yannick Seurin. 2019. Aggregate cash systems: a cryptographic investigation of Mimblewimble. In *EUROCRYPT'19*.

[19] Adam Gibson. 2016. An investigation into confidential transactions. https://github.com/AdamISZ/ConfidentialTransactionsDoc/blob/master/essayonCT.pdf.

[20] Jens Groth and Markulf Kohlweiss. 2015. One-out-of-many proofs: or how to leak a secret and spend a coin. In *EUROCRYPT'15*.

[21] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub. In *NDSS'17*.

[22] Tom Elvis Jedusor. 2016. Mimblewimble. https://scalingbitcoin.org/papers/mimblewimble.txt.

[23] kenshi84. 2016. Tag-linkable Ring-CT with multiple inputs and one-time keys. https://www.overleaf.com/read/xyhymkfjfqmn.

[24] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In *S&P'16*.

[25] Philip Koshy, Diana Koshy, and Patrick McDaniel. 2014. An analysis of anonymity in Bitcoin using P2P network traffic. In *FC'14*.

[26] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. 2017. A traceability analysis of Monero's blockchain. In *ESORICS'17*.

[27] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. 2019. Omniring: scaling private payments without trusted setup formal foundations and a construction of ring confidential transactions with log-size proofs. Full version of this paper. IACR Cryptology ePrint Archive, Report 2019/580. https://eprint.iacr.org/2019/580.

[28] Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. 2019. Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography. In *CCS'19*.

[29] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. 2004. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP'04*.

[30] Cryptocurrency market capitalizations Monero. 2019. visited Aug, 8th 2019. https://coinmarketcap.com/currencies/monero/.

[31] Greg Maxwell. 2013. CoinJoin: Bitcoin privacy for the real world. Post on Bitcoin Forum. https://bitcointalk.org/index.php?topic=279249. (2013).

[32] Greg Maxwell. 2015. Confidential transactions. https://people.xiph.org/~greg/confidential_values.txt.

[33] Sarah Meiklejohn and Rebekah Mercer. 2018. Möbius: Trustless tumbling for transaction privacy. *PoPETs*, 2018, 2.

[34] Sarah Meiklejohn and Claudio Orlandi. 2015. Privacy-enhancing overlays in Bitcoin. In *BITCOIN'15*.

[35] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *IMC'13*.

[36] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. 2013. Zerocoin: anonymous distributed e-cash from Bitcoin. In *S&P'13*.

[37] Monero. 2014. https://getmonero.org/.

[38] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, *et al.* 2018. An empirical analysis of traceability in the Monero blockchain. *PoPETs*, 2018, 3.

[39] Shen Noether, Adam Mackenzie, and the Monero Research Lab. 2016. Ring confidential transactions. *Ledger*, 1. https://www.ledgerjournal.org/ojs/index.php/ledger/article/view/34.

[40] Tatsuaki Okamoto and David Pointcheval. 2001. The gap-problems: a new class of problems for the security of cryptographic schemes. In *PKC'01*.

[41] Torben P. Pedersen. 1992. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*.

[42] Andrew Poelstra. 2006. Mimblewimble. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf.

[43] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. 2017. Confidential assets. In *BITCOIN'17*.

[44] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the Bitcoin system. In *SXSW'13*.

[45] Tim Ruffing and Pedro Moreno-Sanchez. 2017. ValueShuffle: mixing confidential transactions for comprehensive transaction privacy in Bitcoin. In *BITCOIN'17*.

[46] Tim Ruffing, Sri Aravinda Krishnan Thyagarajan, Viktoria Ronge, and Dominique Schröder. 2018. Burning zerocoins for fun and for profit: a cryptographic denial-of-spending attack on the Zerocoin protocol. In *CVCBT'18*.

[47] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. 2017. P2P mixing and unlinkable Bitcoin transactions. In *NDSS'17*.

[48] Victor Shoup. 2001. A proposal for an ISO standard for public key encryption (version 2.1). https://www.shoup.net/papers/iso-2_1.pdf.

[49] Riccardo Spagni. 2018. Monero 0.13.0 "Beryllium Bullet" release. https://getmonero.org/2018/10/11/monero-0.13.0-released.html.

[50] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. 2014. BitIodine: extracting intelligence from the Bitcoin network. In *FC'14*.

[51] Shi-Feng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. 2017. RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero. In *ESORICS'17*.

[52] Luke Valenta and Brendan Rowan. 2015. Blindcoin: blinded, accountable mixes for Bitcoin. In *BITCOIN'15*.

[53] Nicolas van Saberhagen. 2013. Cryptonote v 2.0. https://cryptonote.org/whitepaper.pdf.

[54] Nathan Wilcox. 2015. Specify a mitigation of the faerie gold vulnerability. Zcash Issue on GitHub. https://github.com/zcash/zcash/issues/98.

[55] Tsz Hon Yuen, Shi-feng Sun, Joseph K. Liu, Man Ho Au, Muhammed F. Esgin, Qingzhao Zhang, and Dawu Gu. 2019. RingCT 3.0 for blockchain confidential transaction: shorter size and stronger security. https://eprint.iacr.org/2019/508.

[56] Zcash. 2016. https://z.cash/.

[57] Zcoin. 2016. https://zcoin.io/.

[58] Jan Henrik Ziegeldorf, Fred Grossmann, Martin Henze, Nicolas Inden, and Klaus Wehrle. 2015. CoinParty: secure multi-party mixing of bitcoins. In *CODASPY'15*.