An extended abstract of this paper appears at ACM CCS 2019. This is the full version.

Omniring: Scaling Private Payments Without Trusted Setup

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

Russell W. F. Lai[*], Viktoria Ronge[*], Tim Ruffing[†], Dominique Schröder[*], Sri Aravinda Krishnan Thyagarajan[*], Jiafan Wang[‡]

[*]Friedrich-Alexander University Erlangen-Nuremberg, Germany
[†]Blockstream
[‡]Chinese University of Hong Kong, Hong Kong

April 9, 2020

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.

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, 30, 40, 41, 50, 57]. 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 [7, 9, 24, 25, 29, 37, 38, 42, 49, 51, 53, 59, 60, 65], and multiple cryptocurrencies with a special focus on privacy have emerged [43, 63, 64]. Monero [43] with a market capitalization of $1.6 billion at the time of writing [36] is the largest such privacy-focused cryptocurrency. In contrast to other approaches, most notably Zerocash [7], 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.* [45] 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 [34] which guarantee some form of *anonymity* and enable to detect double-spends simultaneously. The second component of RingCT is Confidential Transactions (CT) [38], 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) [60], 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.* [45], 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 $[0,...,2^\beta-1]$. 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 [12] for zero-knowledge proofs, has been deployed in Monero [56]. 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.* [44] and Kumar *et al.* [31].

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 [52] in Zerocoin [24, 42], 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.* [45] either, even though they generally apply to the RingCT setting as well [52]. While the concrete RingCT

scheme by Noether *et al.* [45] seems not susceptible to such attacks, the fact that the threat has apparently been overlooked in [45] underlines the importance of a rigorous and thorough security model.

Sun *et al.* [58] and Yuen *et al.* [62] propose formalizations of RingCT that improve on the rather informal security notions by Noether *et al.* [45]. 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 [58] 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 [12] 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 [11, 24]. The technique can also be generalized to proving general (bilinear) group arithmetic relations [32], 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 Appendix F 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.* [44] and Kumar *et al.* [31]. 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 [24] or Bootle and Groth [10]) with the most efficient range proof scheme to date (in the Bulletproofs framework [12]), 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 [12], 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 $\vec{\mathbf{R}}=(R_1\|...\|R_n)$ 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 [12] 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 Appendix B, 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 [34]. 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 $\mathsf{OTAccGen}$ 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 $\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}$ 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 $\mathsf{OTAccGen}$ 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 $\mathsf{CheckAmount}$.

Receiving A user can receive funds from a target account acc that it owns by running $\mathsf{Receive}$ on acc which was published (*e.g.*, via a public ledger) as part of a transaction. Using the output of $\mathsf{Receive}$, 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 [61] or burning bug [17]. To ensure that this check is sufficient, the balance property will guarantee that the tag output by $\mathsf{Receive}$ 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 $\mathsf{Receive}$ 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 $\mathsf{poly}(\lambda)$ the set of polynomials in λ and we write $\mathsf{negl}(\lambda)$ 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 $[n]:=\{1,...,n\}$. 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* $(\mathsf{Setup},\mathsf{SAKGen},\mathsf{OTAccGen},\mathsf{Receive},\mathsf{Spend},\mathsf{Vf})$*, and a tuple of auxiliary* PPT *algorithms* $(\mathsf{CheckAmount},\mathsf{CheckTag})$ *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.*

$(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{SAKGen}(\mathsf{pp})$*: 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.*

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

$(\mathsf{ck},a,\mathsf{sk},\mathsf{tag}) \leftarrow \mathsf{Receive}(\mathsf{msk},\mathsf{acc})$*: 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*.*

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

- $\mathcal{R} = \left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}$*: 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* $\mathsf{acc}_{j_i}^{\mathcal{R}}$*)*
- $\mathcal{T} = \left\{(\mathsf{ck}_i^{\mathcal{T}},a_i^{\mathcal{T}},\mathsf{acc}_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|}$*: 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*.*

$b \leftarrow \mathsf{Vf}(\mathsf{tx},\sigma)$*: 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:*

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

$b \leftarrow \mathsf{CheckAmount}(\mathsf{acc},\mathsf{ck},a)$*: 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.*

$b \leftarrow \mathsf{CheckTag}(\mathsf{acc},\mathsf{sk},\mathsf{tag})$*: 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* $\lambda,\alpha,\beta \in \mathbb{N}$*, 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* $a \in \{0,...,2^\beta - 1\}$*, 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*
  - $|\mathcal{T}| \leq 2^\alpha$*,*
  - *for all* $i \in [|\mathcal{T}|]$*,* $a_i^{\mathcal{T}} \in \left\{0,...,2^\beta - 1\right\}$*,*
  - *for all* $i \in [|\mathcal{S}|]$*,* $\mathsf{CheckTag}(\mathsf{acc}_{j_i}^{\mathcal{R}},\mathsf{sk}_i,\mathsf{tag}_i) = 1$*,*
  - *for all* $i \in [|\mathcal{S}|]$*,* $\mathsf{CheckAmount}(\mathsf{acc}_{j_i}^{\mathcal{R}},\mathsf{ck}_i^{\mathcal{S}},a_i^{\mathcal{S}}) = 1$*,*
  - *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 Appendix B we extend the definitions further to account for the tracking and viewing features.

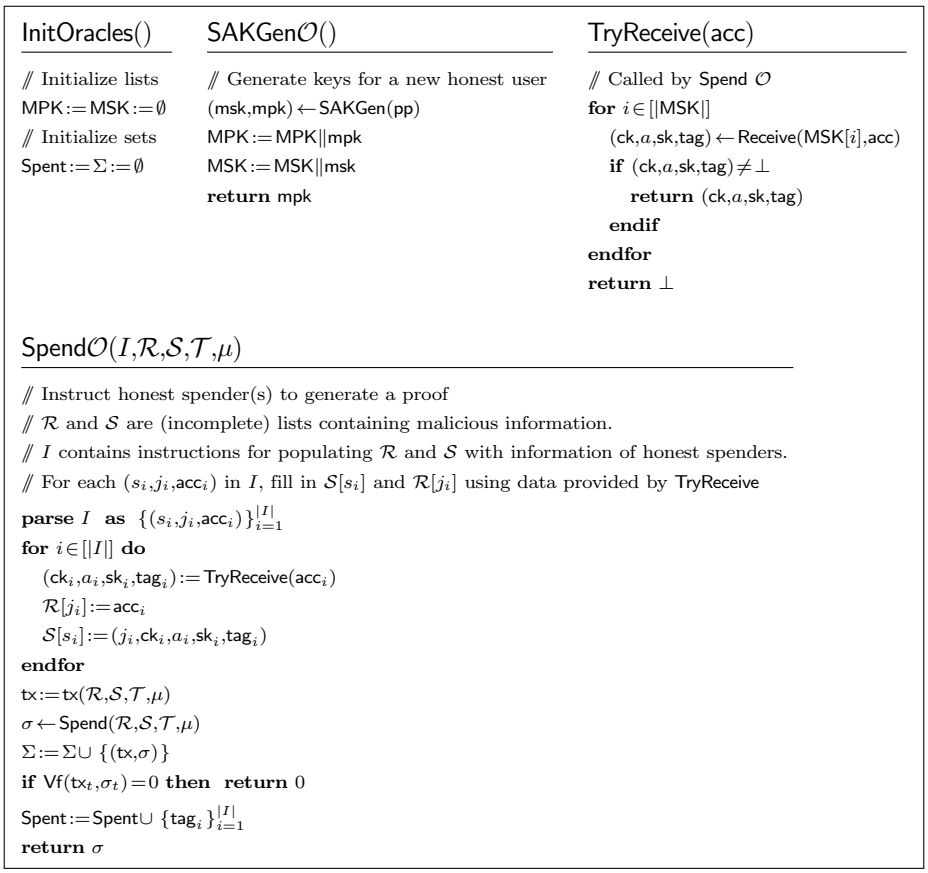

Figure 1: Oracles for security experiments.

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.

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.

$\mathsf{Balance}_{\Omega,\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^{\lambda},1^{\alpha},1^{\beta})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda},1^{\alpha},1^{\beta})$

$(\mathsf{tx},\sigma) \leftarrow \mathcal{A}(\mathsf{pp})$

$(\mathcal{R},\mathcal{S},\mathcal{T},\mu) \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp},\mathsf{tx},\sigma)$

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.

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.*, $\sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} \geq \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}}$.

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 $\mathsf{CheckAmount}$. 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 $\mathsf{CheckTag}$. Therefore double-spending is infeasible.

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

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

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

$\mathsf{Privacy}^{b}_{\Omega,\mathcal{A}}(1^{\lambda},1^{\alpha},1^{\beta})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda},1^{\alpha},1^{\beta})$, $\mathsf{InitOracles}()$
$\mathbb{O} := \{\mathsf{SAKGen}\mathcal{O}, \mathsf{Spend}\mathcal{O}\}$
$(I,J,\mathcal{R},\mathcal{S},\mathcal{T},\mu) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$
$\mathcal{S}_0 := \mathcal{S}_1 := \mathcal{S}$, $\mathcal{T}_0 := \mathcal{T}_1 := \mathcal{T}$
// 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.

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

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

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.

$\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda,1^\alpha,1^\beta)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$, $\mathsf{InitOracles}()$

$(\mathsf{tx}^*,\sigma^*) \leftarrow \mathcal{A}^{\mathsf{SAKGen}\mathcal{O},\mathsf{Spend}\mathcal{O}}(\mathsf{pp})$

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

$b_0 := \mathsf{Vf}(\mathsf{tx}^*,\sigma^*)$

$b_1 := \left((\mathsf{tx}^*,\sigma^*) \notin \Sigma\right)$

$b_2 := \left(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|} \cap \mathsf{Spent} \neq \emptyset \right)$

return $b_0 \wedge b_1 \wedge b_2$

Figure 4: Non-slanderability experiment.

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 $\left\{(s_i,\{j_{t,i},\mathsf{acc}_{t,i}^{\mathcal{S}}\})\right\}_{i=1}^{|I|}$. The experiment retrieves the information required to spend from account $\mathsf{acc}_{t,i}^{\mathcal{S}}$ 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 $\left\{(d_j,\{k_{t,j}^{\mathcal{T}},a_{t,j}^{\mathcal{T}}\}_{t=0}^{1})\right\}_{j=1}^{|J|}$. The experiment creates a one-time account using the master public key of user $k_{t,j}^{\mathcal{T}}$ and the amount $a_{t,j}^{\mathcal{T}}$, 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* $\alpha,\beta \in \mathsf{poly}(\lambda)$*,*

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

where $\mathsf{Privacy}_{\Omega,\mathcal{A}}^{b}$ *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 [52].

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.

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

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

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 of RingCT schemes and an efficient instantiation.

4.1 Tagging Scheme

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

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

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

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

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

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

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 $\mathsf{PKE}=(\mathsf{PKESetup},\mathsf{KGen},\mathsf{Enc},\mathsf{Dec})$ be a (labeled) public-key encryption scheme, $\mathsf{HC}=(\mathsf{HCSetup},\mathsf{Com})$ be a homomorphic commitment scheme with message space $(\mathcal{M},+)$ where $\{0,1,\ldots,2^{\alpha+\beta}-1\}\subseteq\mathcal{M}$ and randomness space $(\rho,+)$, $\mathsf{Tag}=(\mathsf{TagSetup},\mathsf{TagEval})$ be a tagging scheme with secret key space space $(\chi,+)$, public key space $(\mathcal{X},+)$, and tag space ψ, and $\mathsf{SoK}=(\mathsf{SoKSetup},\mathsf{SoKSig},\mathsf{SoKVf})$ 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; we recall the definitions of these well-known primitives in Appendix A. 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.

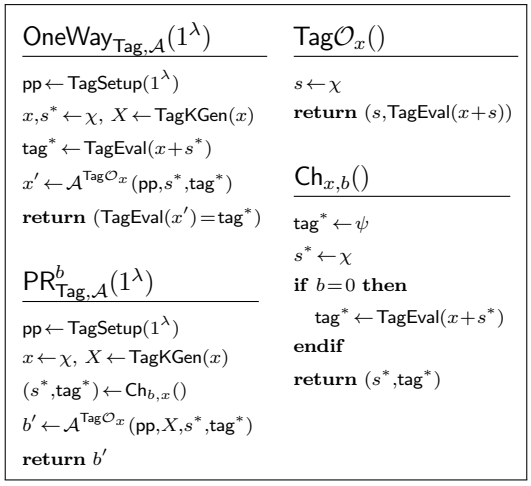

Figure 5: One-wayness and pseudorandomness experiments for tagging schemes.

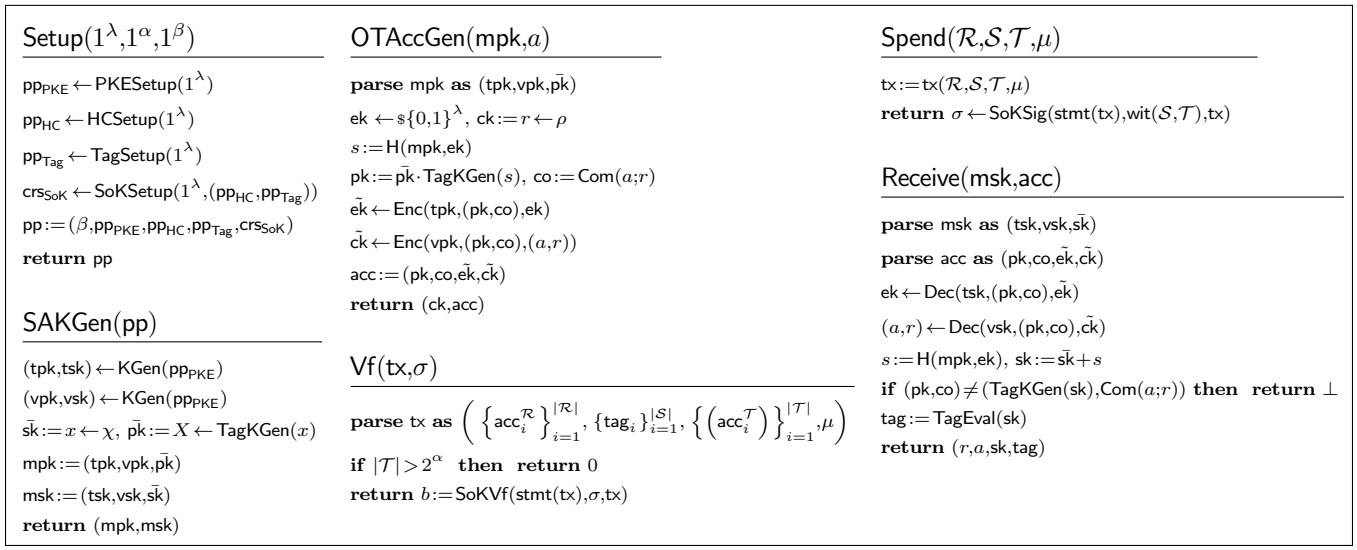

Figure 6: RingCT construction Ω (core components).

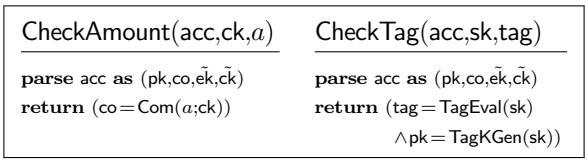

Figure 7: RingCT construction (auxiliary algorithms).

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

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

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

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

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. The key tsk also serves as the tracking secret key.

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

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

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 [6]),* 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 Appendix B, we further extend the construction to provide tracking and viewing features, and provide proofs of the above theorems with respect to the extended definition.

4.4 Concrete Instantiation: Omniring ($\mathcal{O}$)

We propose Omniring, the concrete instantiation $\mathcal{O}$ of our generic construction Ω. We instantiate PKE with a (labeled variant of) ECIES [54], HC with the Pedersen commitment [47], and Tag with the pseudorandom function of Dodis and Yampolsky [19] in a non-black-box manner, which we denote by $\mathsf{Tag}_{\mathcal{O}}$.

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.1 for details). Let $H\in\mathbb{G}$ be another random generator of $\mathbb{G}$. We set $\mathsf{pp}_{\mathsf{HC}}:=(\mathbb{G},q,G,H)$ which defines $\mathcal{M}:=\mathbb{Z}_q$ and $\rho:=\mathbb{Z}_q$. We also set $\mathsf{pp}_{\mathsf{Tag}}:=(\mathbb{G},q,G,H)$ so that the secret key, public key, and tag spaces of the tagging scheme Tag is $\chi:=\mathbb{Z}_q^*$, $\mathcal{X}:=\mathbb{G}\setminus\{G^0\}$ and $\psi:=\mathbb{G}$ respectively. For $a,r\in\mathbb{Z}_q$, we define $\mathsf{Com}(a;r):=G^aH^r$. 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 E.

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

$$\mathsf{stmt}=\Big(\ \big\{\mathsf{pk}_i^{\mathcal{R}},\mathsf{co}_i^{\mathcal{R}}\big\}_{i=1}^{|\mathcal{R}|},\ \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|},\ \big\{\mathsf{co}_i^{\mathcal{T}}\big\}_{i=1}^{|\mathcal{T}|}\Big),$$
$$\mathsf{wit}=\Big(\ \big\{(j_i,x_i,a_i^{\mathcal{S}},r_i^{\mathcal{S}})\big\}_{i=1}^{|\mathcal{S}|},\ \big\{(a_i^{\mathcal{T}},r_i^{\mathcal{T}})\big\}_{i=1}^{|\mathcal{T}|}\Big),$$

we define the following notation:

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

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

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

The language becomes:

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

Finally, we instantiate SoK for the language $\mathcal{L}[\mathbb{G},q,G,H]$ by applying the Fiat-Shamir transform [21] 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 $\mathcal{L}_{\mathcal{U}}[\mathbb{G},q,G,H]$. 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 [12] 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 and is not specified unless there is an ambiguity. We use "$\|$" as an operator for concatenating vectors. Inner products are denoted by $\langle\vec{\mathbf{u}},\vec{\mathbf{v}}\rangle=\sum_{i=1}^{n}u_iv_i$. Let $\vec{\mathbf{G}}\in\mathbb{G}^n$ be a vector of group elements, and $x\in\mathbb{N}$. We define the following operations between vectors of group elements and (vectors of) integers:

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

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

- Hadamard Products: $\vec{\mathbf{u}}\circ\vec{\mathbf{v}}:=(u_1v_1,...,u_nv_n)$
- Hadamard Inverse: $\vec{\mathbf{u}}^{\circ-1}:=(v_1,...,v_n)$ where $v_i=u_i^{-1}$ if $u_i\neq0$, and $v_i=1$ otherwise.
- Kronecker Products: $\vec{\mathbf{u}}\otimes\vec{\mathbf{v}}:=(u_1\vec{\mathbf{v}},...,u_n\vec{\mathbf{v}})$
- Consecutive Powers: $\vec{x}^n=(1,x,...,x^{n-1})$

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

5.1 Our Basic Protocol

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

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

$\underline{\langle\mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle}$:
$\mathcal{V}$:

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

$$\vec{\mathbf{G}}_w := ((G \| H \| \hat{T} \| \hat{\mathbf{Y}})^{\circ w} \circ \vec{\mathbf{P}} \| \vec{\mathbf{G}}') \tag{1}$$

$\mathcal{P}$:

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 $I = G^\xi H^\eta \hat{T}^\tau \hat{\mathbf{Y}}^{\hat{\mathbf{e}}}$. Thus $A = F^{r_A} \vec{\mathbf{G}}_w^{\vec{\mathbf{c}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{c}}_R}$ for all $w \in \mathbb{Z}_q$.

$\mathcal{P} \to \mathcal{V}$: A
$\mathcal{V}$: $w \leftarrow_\$ \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V}$: w
$\mathcal{P}$:

1. $r_S \leftarrow_\$ \mathbb{Z}_q$, $\vec{\mathbf{s}}_L \leftarrow_\$ \mathbb{Z}_q^m$, $\vec{\mathbf{s}}_R \leftarrow_\$ \{\vec{\mathbf{s}} = (s_1, ..., s_m) \in \mathbb{Z}_q^m : \forall i \in [m],\ \vec{\mathbf{c}}_R[i] = 0 \implies s_i = 0\}$.
2. $S := F^{r_S} \vec{\mathbf{G}}_w^{\vec{\mathbf{s}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{s}}_R}$.

$\mathcal{P} \to \mathcal{V}$: 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):

$$\begin{aligned} l(X) &:= \vec{\mathbf{c}}_L + \vec{\alpha} + \vec{\mathbf{s}}_L \cdot X \\ r(X) &:= \vec{\theta} \circ (\vec{\mathbf{c}}_R + \vec{\mathbf{s}}_R \cdot X) + \vec{\mu} \\ t(X) &:= \langle l(X), r(X) \rangle = t_2 X^2 + t_1 X + t_0 \end{aligned}$$

   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. $T_1 := G^{t_1} F^{\tau_1}$, $T_2 := G^{t_2} F^{\tau_2}$

$\mathcal{P} \to \mathcal{V}$: 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. $(\vec{l}, \vec{r}, t) := (l(x), r(x), t(x))$

$\mathcal{P} \to \mathcal{V}$: $\tau, r, \vec{l}, \vec{r}, t$
$\mathcal{V}$: Check if the following relations hold:

$$t = \langle \vec{l}, \vec{r} \rangle \tag{2}$$

$$F^r \vec{\mathbf{G}}_w^{\vec{l}} \vec{\mathbf{H}}^{\vec{\theta}^{\circ -1} \circ \vec{r}} = A S^x \vec{\mathbf{G}}_w^{\vec{\alpha}} \vec{\mathbf{H}}^{\vec{\beta}} \tag{3}$$

$$G^t H^\tau = G^\delta \vec{\mathbf{C}}_{\mathcal{T}}^{z^2 \cdot \vec{y}^{|\mathcal{T}|}} T_1^x T_2^{x^2} \tag{4}$$

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 Appendix D.

5.2 Inner Product Argument

We next recall the argument of knowledge for inner product relations in [12]. 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:

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

We denote the argument of knowledge protocol of [12] 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 [12]. 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 (Modified from [12])**.** *Let $\vec{\mathbf{G}}$ and $\vec{\mathbf{H}}$ be sampled as in Corollary 1 such 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.*

The proof of the above theorem is almost identical to that given in [12] and is therefore 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 (1), 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 $F^r P = A S^x \vec{\mathbf{G}}_w^{\vec{\alpha}} \vec{\mathbf{H}}^{\vec{\beta}}$ 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 [12], 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.

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

Figure 8: Honest encoding of witness.

6 Optimizations and Performance

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

Notation	*Description*
$\hat{\mathbf{Y}}=\hat{\mathbf{Y}}(u):=\vec{\mathbf{R}}\circ\vec{\mathbf{C}}_{\mathcal{R}}^{\circ u}$	Vector of compressed public keys and coins with randomness $u\in\mathbb{Z}_q$.
$\hat{\mathbf{e}}=\hat{\mathbf{e}}(v):=\vec{v}^{\lvert\mathcal{S}\rvert}\mathbf{E}$	Vector of compressed unit vectors with randomness $v\in\mathbb{Z}_q$.
$\hat{T}=\hat{T}(u,v):=\vec{\mathbf{T}}^{u^2\vec{v}^{\lvert\mathcal{S}\rvert}}$	Compressed tag with randomness $u,v\in\mathbb{Z}_q$.
$\xi=\xi(u,v):=-\langle\vec{v}^{\lvert\mathcal{S}\rvert},u\cdot\vec{\mathbf{a}}^{\mathcal{S}}+u^2\cdot\vec{\mathbf{x}}^{\circ-1}\rangle$	Compressed secrets with randomness $u,v\in\mathbb{Z}_q$.
$\eta=\eta(u,v):=-\langle\vec{v}^{\lvert\mathcal{S}\rvert},\vec{\mathbf{x}}+u\cdot\vec{\mathbf{r}}^{\mathcal{S}}\rangle$	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.
$m=3+\lvert\mathcal{R}\rvert+\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert+3\lvert\mathcal{S}\rvert$	Length of $\vec{\mathbf{c}}_L$ and $\vec{\mathbf{c}}_R$.
$(\vec{\mathbf{v}}_0,...,\vec{\mathbf{v}}_8,\vec{\mathbf{u}}_4)=(\vec{\mathbf{v}}_0,...,\vec{\mathbf{v}}_8,\vec{\mathbf{u}}_4)(u,v,y)$	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.

$$\begin{bmatrix}\vec{\mathbf{v}}_0\\ \vec{\mathbf{v}}_1\\ \vec{\mathbf{v}}_2\\ \vec{\mathbf{v}}_3\\ \vec{\mathbf{v}}_4\\ \vec{\mathbf{v}}_5\\ \vec{\mathbf{v}}_6\\ \vec{\mathbf{v}}_7\\ \vec{\mathbf{v}}_8\\ \vec{\mathbf{u}}_4\end{bmatrix} := \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert} & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{S}\rvert} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{T}\rvert}\otimes\vec{2}^{\beta} & \cdot & \cdot & \cdot \\ \cdot & \cdot & y^{\lvert\mathcal{S}\rvert} & \cdot & \vec{y}^{\lvert\mathcal{S}\rvert}\otimes\vec{1}^{\lvert\mathcal{R}\rvert} & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\cdot\vec{v}^{\lvert\mathcal{S}\rvert} & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\cdot\vec{v}^{\lvert\mathcal{S}\rvert} & \vec{v}^{\lvert\mathcal{S}\rvert} \\ \cdot & \cdot & \cdot & -\vec{y}^{\lvert\mathcal{R}\rvert} & \vec{v}^{\lvert\mathcal{S}\rvert}\otimes\vec{y}^{\lvert\mathcal{R}\rvert} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \vec{1}^{\lvert\mathcal{T}\rvert}\otimes\vec{2}^{\beta} & -\vec{1}^{\lvert\mathcal{S}\rvert} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert} & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & u^2\cdot\vec{v}^{\lvert\mathcal{S}\rvert} \end{bmatrix}$$

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

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 [12], where $m=3+\lvert\mathcal{R}\rvert+\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert+3\lvert\mathcal{S}\rvert$. 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 [12], we refer the reader to Bünz *et al.* [12].

6.2 Log-size Transactions

While Omniring produces spend proofs of logarithmic size, the spender needs to communicate the set of destination accounts $\left\{\mathsf{acc}_i^{\mathcal{T}},\mathsf{info}_i\right\}_{i=1}^{\lvert\mathcal{T}\rvert}$, the set of tags $\left\{\mathsf{tag}_i\right\}_{i=1}^{\lvert\mathcal{S}\rvert}$, and a set of ring accounts $\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{\lvert\mathcal{R}\rvert}$ within the transaction to allow the cryptocurrency network to verify the transaction. Since $\lvert\mathcal{T}\rvert$ and $\lvert\mathcal{S}\rvert$ are typically small (we typically have $\lvert\mathcal{T}\rvert=2$ and $\lvert\mathcal{S}\rvert<5$), they can be safely neglected. However, the ring size $\lvert\mathcal{R}\rvert$ is a problem if a high level of privacy is desired. The obvious solution to include $\lvert\mathcal{R}\rvert$ ring members in the transaction needs $O(\lvert\mathcal{R}\rvert)$ space which quickly becomes impractical. However, using the *recovery sampling* technique by Chator and Green [15], which is built for this exact purpose, the description of the set of ring members can be as short as $O(\log\lvert\mathcal{R}\rvert)$, yielding a overall transaction size (not only proof size) logarithmic in m.

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

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

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

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

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

$$\mathsf{EQ}(\vec{\gamma}_L, \vec{\gamma}_R) = 0 \iff$$

$$\begin{cases} \langle \vec{\gamma}_L, \vec{\gamma}_R \circ \vec{\mathbf{v}}_0 \rangle & = 0 & (5) \\ \langle \vec{\gamma}_L, \vec{\gamma}_R \circ \vec{\mathbf{v}}_1 \rangle & = \langle \vec{1}^{|\mathcal{S}|}, \vec{y}^{|\mathcal{S}|} \rangle & (6) \\ \langle \vec{\gamma}_L, \vec{\mathbf{v}}_2 \rangle & = \langle \vec{\mathbf{a}}^{\mathcal{T}}, \vec{y}^{|\mathcal{T}|} \rangle & (7) \\ \langle \vec{\gamma}_L, \vec{\mathbf{v}}_3 \rangle & = \langle \vec{1}^{|\mathcal{S}|+1}, \vec{y}^{|\mathcal{S}|+1} \rangle & (8) \\ \langle \vec{\gamma}_L, \vec{\mathbf{v}}_4 \rangle + \langle \vec{\gamma}_R, \vec{\mathbf{u}}_4 \rangle & = 0 & (9) \\ \langle \vec{\gamma}_L, \vec{\mathbf{v}}_5 \rangle & = 0 & (10) \\ \langle \vec{\gamma}_L, \vec{\mathbf{v}}_6 \rangle & = 0 & (11) \\ \langle \vec{\gamma}_L, \vec{\mathbf{v}}_7 \rangle & = 0 & (12) \\ \langle \vec{\gamma}_L - \vec{\gamma}_R - \vec{1}^m, \vec{\mathbf{v}}_8 \rangle & = 0 & (13) \end{cases}$$

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

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.* [45] with a minor modification [27], together with Bulletproofs [12] 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 [62] 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 [58] 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.

[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.

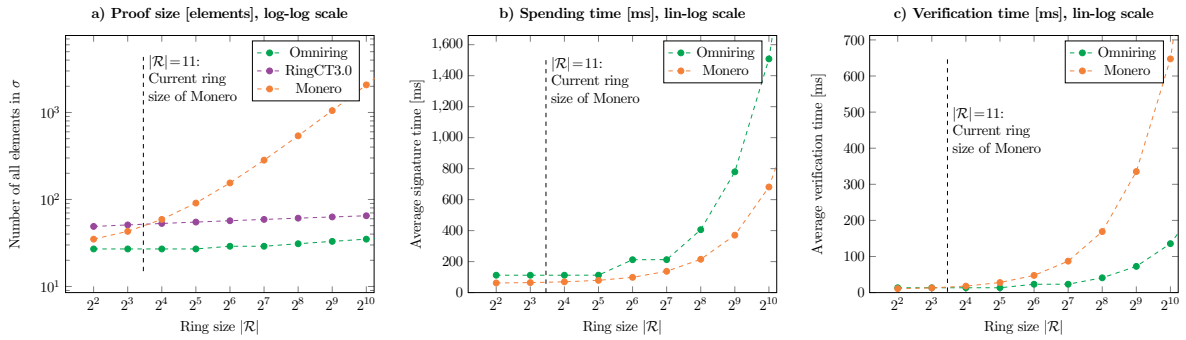

Figure 12: Performance comparison.

Scheme	Spend proof size (in elements)	Pairing	Trusted setup
Monero [27, 45] with Bulletproofs range proofs	$O(\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\log(\beta\lvert\mathcal{T}\rvert))$	No	No
RingCT 2.0 [58] with Bulletproofs range proofs	$O(\lvert\mathcal{S}\rvert+\log(\beta\lvert\mathcal{T}\rvert))$	Yes	Yes
RingCT 3.0 [62] with Bulletproofs range proofs	$O(\lvert\mathcal{S}\rvert\log\lvert\mathcal{R}\rvert+\log(\beta\lvert\mathcal{T}\rvert))$	No	No
Omniring	$O(\log(\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert))$	No	No

Figure 13: Comparison of RingCT schemes.

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 [16]. 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.

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.

Potential Issues of Leaking One-Time Secret Keys For completeness, we point out a potential issue which seems costly to avoid. We observe that in Noether *et al.* [45], leaking a one-time secret key $x=x_2+s$ is almost as bad as leaking the master secret key as s is known to the spender who generates the corresponding one-time public key. With the knowledge of $x_2 = x - s$, the spender can get a "refund" for any future transaction bounded to the same receiver: Suppose in a future transaction the spender creates a new one-time account of the receiver with one-time secret key $x'=x_2+s'$ for some s' known by the spender. Since the spender knows x_2, it can compute x' and spend from this account "on behalf of the receiver".

We do not see any simple solution to this potential issue, since the only one-time public keys a spender can publicly derive are affine functions of the master public key, assuming that the spender only performs generic group operations and does not use pairing. That is to say, our construction also does not protect against this potential attack. If one is willing to use computationally more expensive tools, an option is to use (techniques related to) identity-based encryption which can be constructed with pairing (*e.g.*, [8]) or garbled circuits [20]. Fortunately, although devastating in theory, the attack seems impractical as the victim can easily discover it.

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 [60], 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 [34], 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.* [45] 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 [38] 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 [23]. CT further has been partially formalized by Poelstra *et al.* [49], and fully formalized in the context of the Mimblewimble cryptocurrency design [26, 48] by Fuchsbauer *et al.* [22].

A variant [27] 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 [56].

To our best knowledge, the concept of "stealth addresses" first appeared in CryptoNote v2.0 [60] in the name "public user keys" without formalization. Meiklejohn and Mercer [39] 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.* [58].

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.* [58] 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.* [58] 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.* [62] 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 [42] and Zerocash [7] are designs for cryptocurrencies aiming to provide anonymity and the privacy of amounts based on zero-knowledge proofs. Ring signatures in CryptoNote v2.0 [60] (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 [63]. 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 [24].

8 Acknowledgments

We thank the Monero Research Lab and the anonymous CCS'19 referees for their valuable comments and helpful suggestions.

We also thank the Monero Research Lab and Tsz Hon Yuen for pointing out a bug in the argument of knowledge construction, which is now fixed.

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 Preliminaries

A.1 Computational Hardness 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 $\mathcal{A}$*

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

where the game ℓ-$\mathsf{DL}_{\mathcal{A}}(\mathcal{G})$ is defined in Figure 14. For $\ell=1$, this is the (standard) discrete logarithm assumption.

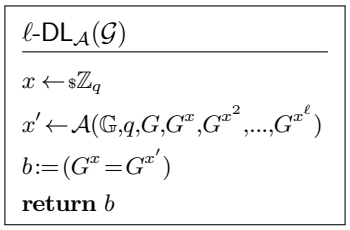

Figure 14: Security game for GDL.

Definition A.2 (Gap Diffie-Hellman Assumption (GapDH) [46])**.** *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.

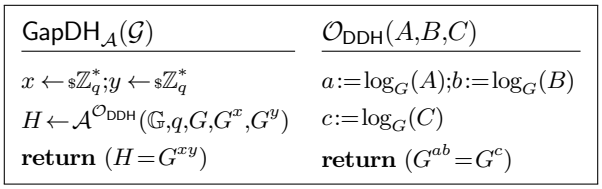

Figure 15: Security game for GapDH.

Definition A.3 (Strong Decisional Diffie-Hellman Inversion Assumption (SDDHI) [13])**.** *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.

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

Figure 16: Security game for SDDHI.

$\mathsf{DDH}^b_{\mathcal{A}}(\mathcal{G})$
$x, y, z \leftarrow_{\$} \mathbb{Z}_q^*$
$b \leftarrow_{\$} \{0,1\}$
$Z_0 = G^{xy}$
$Z_1 = G^z$
$b' \leftarrow \mathcal{A}(\mathbb{G}, q, G, G^x, G^y, Z_b)$
$b_0 := (b = b')$
return b_0

Figure 17: Security Game for DDH.

Definition A.4 (Decisional Diffie-Hellman (DDH))**.** *We say that the DDH assumption according to holds over $\mathcal{G}$ if for all $\ell \in \mathsf{poly}(\lambda)$, every PPT adversary $\mathcal{A}$*

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

where the game $\mathsf{DDH}_{\mathcal{A}}(\mathcal{G})$ is defined in Figure 17.

We now define an interactive variant of the discrete logarithm (representation) assumption, and show that it is equivalent to the standard discrete logarithm assumption.

Definition A.5 (Interactive Discrete Logarithm (iDL) Assumption)**.** *We say that the interactive discrete logarithm assumption holds over $\mathcal{G}$ if for all $\ell \in \mathsf{poly}(\lambda)$, every PPT adversary $\mathcal{A}$*

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

where the game ℓ-$\mathsf{iDL}_{\mathcal{A}}(\mathcal{G})$ is defined in Figure 18.

Theorem A.6. *The interactive discrete logarithm assumption holds over $\mathcal{G}$ if and only if the discrete logarithm assumption (1-DL) holds over $\mathcal{G}$.*

Proof. The forward direction is trivial. For the backward direction, suppose there exists a PPT adversary $\mathcal{A}$ which solves the interactive discrete logarithm problem for some $\ell \in \mathbb{N}$. We construct a PPT algorithm $\mathcal{B}$ for the discrete logarithm problem as follows.

$\mathcal{B}$ receives the discrete logarithm instance G^x for some unknown x. It runs $\mathcal{A}$ up to the point where $\mathcal{A}$ outputs a vector of group elements $\vec{\mathbf{G}}$. Let $m = |\vec{\mathbf{G}}| \in \mathsf{poly}(\lambda)$ be the length of $\vec{\mathbf{G}}$. $\mathcal{B}$ forks the execution of $\mathcal{A}$ into $m+1$ parallel instances. Let $i^* \leftarrow_{\$} [\ell]$ and $j^* \leftarrow_{\$} [m+1]$. For the j-th instance, $\mathcal{B}$ does the following:

ℓ-iDL$_{\mathcal{A}}(\mathcal{G})$

$(\vec{\mathbf{G}},\mathsf{st}) \leftarrow \mathcal{A}(\mathbb{G},q,G)$
$\vec{\mathbf{H}} \leftarrow_{\$} \mathbb{G}^{\ell}$
$(\vec{\mathbf{a}},\vec{\mathbf{b}}) \leftarrow \mathcal{A}(\mathsf{st},\vec{\mathbf{H}})$
$b_0 := (I = \vec{\mathbf{G}}^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}})$
$b_1 := (\vec{\mathbf{b}} \neq \vec{0}^{\ell})$
return $b_0 \wedge b_1$

Figure 18: Security Game for iDL

If $j \neq j^*$, $\mathcal{B}$ samples $\vec{\mathbf{x}}_j \leftarrow_{\$} \mathbb{Z}_q^{\ell}$. If $j = j^*$, $\mathcal{B}$ chooses a random index $i^* \leftarrow_{\$} [\ell]$ and writes symbolically $x_{i^*,j^*} := x$. For $i \in [\ell] \setminus \{i^*\}$, it samples $x_{i,j^*} \leftarrow_{\$} \mathbb{Z}_q$. In either case, it sends $\vec{\mathbf{H}}_j := G^{\vec{\mathbf{x}}_j}$ to $\mathcal{A}$, who responds with $(\vec{\mathbf{a}}_j,\vec{\mathbf{b}}_j)$ with $I = \vec{\mathbf{G}}^{\vec{\mathbf{a}}_j}\vec{\mathbf{H}}_j^{\vec{\mathbf{b}}_j}$ and $\vec{\mathbf{b}}_j \neq \vec{0}^{\ell}$. We can write

$$G^{-\langle \vec{\mathbf{b}}_j,\vec{\mathbf{x}}_j \rangle} = \vec{\mathbf{G}}^{\vec{\mathbf{a}}_j}.$$

Note that $|\vec{\mathbf{a}}_j| = m$ for all $j \in [m+1]$ and therefore the set $\{\vec{\mathbf{a}}_1,...,\vec{\mathbf{a}}_{m+1}\}$ must be linearly dependent, *i.e.*, there exists $c_1,...,c_{m+1}$ not all zero such that

$$c_1 \cdot \vec{\mathbf{a}}_1 + ... + c_{m+1} \cdot \vec{\mathbf{a}}_{m+1} = \vec{0}^m.$$

Since j^* is chosen randomly, with probability at least $1/(m+1)$, we have $c_{j^*} \neq 0$. We can therefore write

$$\vec{\mathbf{a}}_{j^*} = \sum_{j \in [m+1] \setminus \{j^*\}} c'_j \cdot \vec{\mathbf{a}}_j$$

for some $c'_1,...,c'_{j^*-1},c'_{j^*+1},...,c'_{m+1}$. This implies

$$\vec{\mathbf{G}}^{\vec{\mathbf{a}}_{j^*}} = \prod_{j \in [m+1] \setminus \{j^*\}} \left(\vec{\mathbf{G}}^{\vec{\mathbf{a}}_j}\right)^{c'_j},$$

$$G^{\langle \vec{\mathbf{b}}_{j^*},\vec{\mathbf{x}}_{j^*} \rangle} = \prod_{j \in [m+1] \setminus \{j^*\}} \left(G^{\langle \vec{\mathbf{b}}_j,\vec{\mathbf{x}}_j \rangle}\right)^{c'_j},$$

$$G^{b_{i^*,j^*}x} = G^{\sum_{j \in [m+1] \setminus \{j^*\}} c'_j \cdot \langle \vec{\mathbf{b}}_j,\vec{\mathbf{x}}_j \rangle - \sum_{i \in [\ell] \setminus \{i^*\}} b_{i,j^*}x_{i,j^*}}.$$

Since i^* is chosen randomly, with probability at least $1/\ell$, we have $b_{i^*,j} \neq 0$. Hence we have

$$x = (b_{i^*,j})^{-1} \cdot \left(\sum_{j \in [m+1] \setminus \{j^*\}} c'_j \cdot \langle \vec{\mathbf{b}}_j,\vec{\mathbf{x}}_j \rangle - \sum_{i \in [\ell] \setminus \{i^*\}} b_{i,j^*}x_{i,j^*} \right),$$

which is a solution to the discrete logarithm problem instance.

□

A.2 Arguments of Knowledge

Definition A.7 (Arguments of Knowledge)**.** *A triple* $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ *is called an* argument of knowledge *for a NP relation* $\mathcal{R}$ *for a language* $\mathcal{L}_R = \{\mathsf{stmt} \mid \exists \mathsf{wit} : \mathcal{R}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}) = 1\}$ *if it satisfies the following two definitions.*

On input 1^{λ} the setup algorithm Setup produces a common reference string crs. When interacting the prover $\mathcal{P}$ and verifier $\mathcal{V}$ produce a transcript $tr = \langle \mathcal{P}(\cdot),\mathcal{V}(\cdot) \rangle$ where $\langle \cdot \rangle$ denotes the actual protocol between $\mathcal{P}$ and $\mathcal{V}$.

Definition A.8 (Perfect Completeness)**.** $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ *has* perfect completeness *if for all non-uniform polynomial time adversaries* $\mathcal{A}$*,*

$$\mathsf{Pr}\left[\begin{matrix} (\mathsf{crs},\mathsf{stmt},\mathsf{wit}) \notin \mathcal{R} \ \vee \\ \langle \mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{crs},\mathsf{stmt}) = 1 \rangle \end{matrix} \middle| \begin{matrix} \mathsf{crs} \leftarrow \mathsf{Setup}(1^{\lambda}) \\ (\mathsf{stmt},\mathsf{wit}) \leftarrow \mathcal{A}(\mathsf{crs}) \end{matrix} \right] = 1.$$

Definition A.9 (Computational Witness-Extended Emulation [33]). $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ *has* witness-extended emulation *if for all deterministic polynomial time* $\mathcal{P}^*$ *there exists an expected polynomial time emulator* $\mathcal{E}$ *such that for all pairs of interactive adversaries* $\mathcal{A}_1$, $\mathcal{A}_2$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left|\mathsf{Pr}\left[\mathcal{A}_1(tr)=1 \middle| \begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\(\mathsf{stmt},\mathsf{wit})\leftarrow\mathcal{A}_2(\mathsf{crs}),\\tr\leftarrow\langle\mathcal{P}^*(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\\\mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle\end{array}\right]-\mathsf{Pr}\left[\begin{array}{c}\mathcal{A}_1(tr)=1\wedge\\(tr\text{ is accepting}\\\Rightarrow(\mathsf{crs},\mathsf{stmt},\mathsf{wit})\in\mathcal{R})\end{array}\middle|\begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\(\mathsf{stmt},\mathsf{wit})\leftarrow\mathcal{A}_2(\mathsf{crs}),\\(tr,\mathsf{wit})\leftarrow\mathcal{E}^{\mathcal{O}}(\mathsf{crs},\mathsf{stmt})\end{array}\right]\right|\leq\mathsf{negl}(\lambda).$$

where the oracle is given by $\mathcal{O}=\langle\mathcal{P}^*(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{crs},\mathsf{stmt})\rangle$*, and permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. If the adversaries* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are restricted to run in polynomial time, then we say* $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ *has* computational witness-extended emulation.

Definition A.10 (Public Coin). *An argument of knowledge* $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ *is called* public coin *if all messages sent from the verifier to the prover are chosen uniformly at random and independently of the prover's messages, i.e., the challenges correspond to the verifier's randomness* ρ.

Definition A.11 (Perfect Special Honest-Verifier Zero-Knowledge). *A public coin argument of knowledge* $(\mathsf{Setup},\mathcal{P},\mathcal{V})$ *is a* perfect special-verifier zero knowledge *argument of knowledge for* $\mathcal{R}$ *if there exists a probabilistic polynomial time simulator* $\mathcal{S}$ *such that for all pairs of interactive adversaries* $\mathcal{A}_1$, $\mathcal{A}_2$

$$\begin{aligned}&\mathsf{Pr}\left[\begin{array}{c}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})\in\mathcal{R}\wedge\\\mathcal{A}_1(tr)=1\end{array}\middle|\begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\(\mathsf{stmt},\mathsf{wit},\rho)\leftarrow\mathcal{A}_2(\mathsf{crs}),\\tr\leftarrow\langle\mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{crs},\mathsf{stmt};\rho)\rangle\end{array}\right]\\=&\mathsf{Pr}\left[\begin{array}{c}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})\in\mathcal{R}\wedge\\\mathcal{A}_1(tr)=1\end{array}\middle|\begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\(\mathsf{stmt},\mathsf{wit},\rho)\leftarrow\mathcal{A}_2(\mathsf{crs}),\\tr\leftarrow\mathcal{S}(\mathsf{stmt},\rho)\end{array}\right].\end{aligned}$$

A.3 Signatures of Knowledge

The term signatures of knowledge was widely used in the literature before it was formalized by [14]. We present a simplified definition which captures schemes in the random oracle model.

Definition A.12 (Signatures of Knowledge). *Let* $\mathcal{R}$ *be an NP relation for the language* $\mathcal{L}_R=\{\mathsf{stmt}|\exists\mathsf{wit}:\mathcal{R}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})=1\}$ *for a statement* x *and witness* w*. Let* H *be a random oracle. A signature of knowledge for* $\mathcal{L}$ *and the message* m *in presence of* H *is a tuple of algorithms* $\Sigma=(\mathsf{Setup},\mathsf{SoKSign}^{\mathsf{H}},\mathsf{SoKVerify}^{\mathsf{H}},\mathcal{S})$ *defined below. Note that* SoKSign *and* SoKVerify *have oracle access to the random oracle* H.

$\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda,\mathcal{L})$*: The* setup *algorithm takes as inputs the security parameter* 1^λ *and the description of the language* $\mathcal{L}$*, and outputs the the common reference string* crs.

$\sigma\leftarrow\mathsf{SoKSign}^{\mathsf{H}}(\mathsf{crs},\mathsf{stmt},\mathsf{wit},m)$*: The* signing *algorithm takes as inputs the common reference string* crs*, a statement* $\mathsf{stmt}\in\mathcal{L}$*, the corresponding witness* wit*, and a message* $m\in\mathcal{M}$*, and outputs a signature* σ.

$b\leftarrow\mathsf{SoKVerify}^{\mathsf{H}}(\mathsf{crs},\mathsf{stmt},\sigma,m)$*: The* verification *algorithm takes as inputs the common reference string* crs*, a statement* $\mathsf{stmt}\in\mathcal{L}$*, a signature* σ*, and a message* $m\in\mathcal{M}$*, and outputs a bit* b *deciding whether* σ *is a valid signature on* m.

$\sigma\leftarrow\mathcal{S}(\mathsf{crs},\mathsf{stmt},m)$*: The* simulator *takes as inputs the common reference string* crs*, a statement* $\mathsf{stmt}\in\mathcal{L}$*, and a message* $m\in\mathcal{M}$*, and outputs a signature* σ.

Definition A.13 (Perfect Completeness). *For all* $\lambda\in\mathbb{N}$, $\mathsf{crs}\in\mathsf{Setup}(1^\lambda,\mathcal{L})$, (x,w) *such that* $R(\mathsf{stmt},\mathsf{wit})=1$, $m\in\mathcal{M}$, $\sigma\in\mathsf{SoKSign}^{\mathsf{H}}(\mathsf{crs},\mathsf{stmt},\mathsf{wit},m)$*, we have* $\mathsf{SoKVerify}^{\mathsf{H}}(\mathsf{crs},\mathsf{stmt},\sigma,m)=1$.

Definition A.14 (Perfect Simulatability). *It holds that*

$$\left\{(\mathsf{crs},\sigma)\middle|\begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda,\mathcal{L})\\\sigma\leftarrow\mathsf{SoKSign}^{\mathsf{H}}(\mathsf{crs},\mathsf{stmt},\mathsf{wit},m)\end{array}\right\}=\left\{(\mathsf{crs},\sigma)\middle|\begin{array}{l}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda,\mathcal{L})\\\sigma\leftarrow\mathcal{S}(\mathsf{crs},\mathsf{stmt},m)\end{array}\right\}.$$

Definition A.15 (Extractability). *For all* PPT *adversaries* $\mathcal{A}$*, there exists a* PPT *extractor* $\mathcal{E}_{\mathcal{A}}$ *such that for all* $\mathsf{stmt} \in \{0,1\}^*$*, if*

$$\Pr\left[b=1 \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{L}) \\ (\sigma, m) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{crs}) \\ b \leftarrow \mathsf{SoKVerify}^{\mathsf{H}}(\mathsf{crs}, \mathsf{stmt}, \sigma, m) \end{array}\right] > \mathsf{negl}(\lambda),$$

then

$$\Pr\left[R(x,w)=1 \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(\mathsf{crs}, \mathcal{L}) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{stmt}) \end{array}\right] > 1 - \mathsf{negl}(\lambda).$$

A perfectly complete, perfectly special honest-verifier zero-knowledge, public-coin logarithmic-round argument of knowledge scheme for a language $\mathcal{L}$ with extended-witness emulation can be transformed into a perfectly complete, extractable, perfectly simulatable signature of knowledge scheme for $\mathcal{L}$ and the message space $\mathcal{M} = \{0,1\}^*$ using the Fiat-Shamir heuristic [21].

A.4 Labeled Public-Key Encryption Scheme

The notion of the labeled public-key encryption scheme is formally considered by Shoup [55]. Compared with the standard public-key encryption, the encryption and decryption algorithms of a labeled encryption scheme take an additional labeled as input. A label can be considered as a binary string with a length polynomially bounded by the security parameter.

Definition A.16 (Labeled Public-Key Encryption). *A* labeled public-key encryption scheme *is a tuple of algorithms* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined below.*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: The* setup *algorithm takes as input the security parameter* 1^λ*, and outputs a public parameter* pp*.*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: The* key generation *algorithm takes as input the public parameter* pp*, and outputs a public key* pk *and a secret key* sk*. We assume the existence of an algorithm* $\mathsf{SKVerify}(\mathsf{pk}, \mathsf{sk})$ *which checks if* pk *is a valid public key corresponding to* sk*.*

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, \tau, m)$*: The* encryption *algorithm takes as inputs the public key* pk*, a label* τ*, and a message* m*, and outputs a ciphertext* c*.*

$m \leftarrow \mathsf{Dec}(\mathsf{sk}, \tau, c)$*: The deterministic* decryption *algorithm takes as inputs the secret key* sk*, a label* τ*, and a ciphertext* c*, and outputs a message* m *(or* $\perp$ *upon failure).*

Correctness of PKE requires that for all message-label pairs (m, τ), all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, and all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(\mathsf{pp})$, $\mathsf{Dec}(\mathsf{sk}, \tau, \mathsf{Enc}(\mathsf{pk}, \tau, m))$ always returns m.

We follow the CCA security requirement of the labeled encryption considered in [55]. This notion is stronger than weak CCA in [28, 35] by allowing the adversary to make any query to the decryption oracle provided that the queried label and the queried ciphertext are not the challenging ones simultaneously.

Definition A.17 (IND-CCA). PKE *is* indistinguishable under chosen-ciphertext attack (IND-CCA) *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

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

where the game $\mathsf{IND\text{-}CCA}^b_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ *is defined in Figure 19.*

Key-privacy of the public-key encryption is first introduced by Bellare *et al.* [6], which requires no PPT adversary viewing a chosen message encrypted under one of two public keys can guess which public key is used. It implies that the receiver of the ciphertext (*i.e.*, the owner of the public key) is anonymous from the point of view of the adversary. Key-privacy property is essential for the scenario where identities are needed to be protected, *e.g.*, anonymous communications and cryptocurrencies. We give the definition of key privacy for a labeled public-key encryption scheme as below.

Definition A.18 (IK-CCA). PKE *is* key-private under chosen ciphertext attack (IK-CCA) *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

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

where the game $\mathsf{IK\text{-}CCA}^b_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ *is defined in Figure 20.*

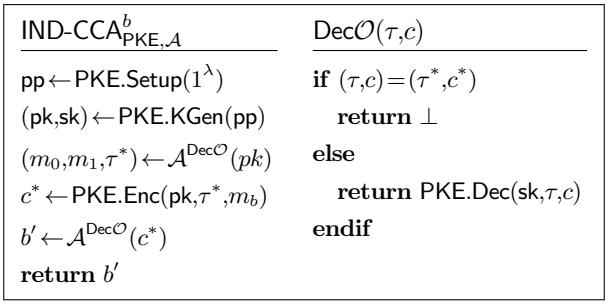

Figure 19: Security game for IND-CCA of PKE.

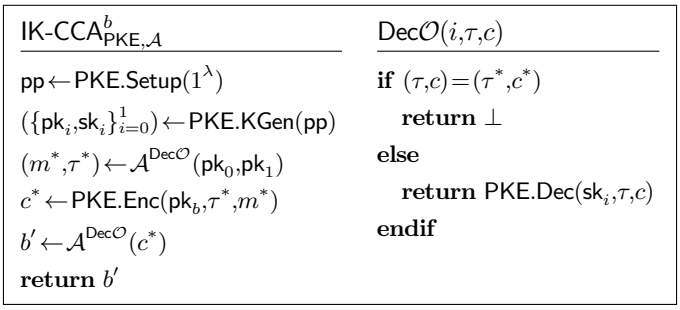

Figure 20: Security game for IK-CCA of PKE.

A.5 Homomorphic Commitment Scheme

A commitment scheme allows the sender to commit to a value and later reveal that value to a receiver by showing the value together with the opening. The receiver is able to verify that this value was indeed contained in the commitment. A commitment scheme should be hiding that the commitment does not tell anything about the committed value, and binding that the commitment can only be opened to the value which it was committed to.

Definition A.19 (Commitment)**.** *A* commitment scheme *is a pair* $\mathsf{HC}=(\mathsf{Setup},\mathsf{Com})$ *of algorithms defined below.*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: The* setup *takes as input the security parameter* 1^λ*, and outputs public parameters* pp*, which specify a message space* $\mathcal{M}$*, a randomness space* χ *and a commitment space* $\mathcal{C}$*.*

$C \leftarrow \mathsf{Com}_{\mathsf{pp}}(m;r)$*: The* commitment *algorithm takes as inputs the common reference string* pp*, a message* $m \in \mathcal{M}$*, and a randomness* $r \in \chi$*, and outputs a commitment* $C \in \mathcal{C}$*.*

Definition A.20 (Perfect Hiding)**.** *A commitment scheme* $\mathsf{HC}=(\mathsf{Setup},\mathsf{Com})$ *is* perfectly hiding *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\mathsf{Hiding}^0_{\mathsf{HC},\mathcal{A}}(1^\lambda)=1\right] = \Pr\left[\mathsf{Hiding}^1_{\mathsf{HC},\mathcal{A}}(1^\lambda)=1\right],$$

where the game $\mathsf{Hiding}^b_{\mathsf{HC},\mathcal{A}}(1^\lambda)$ *is defined in Figure 21.*

Definition A.21 (Computationally Binding)**.** *A commitment scheme* $(\mathsf{Setup},\mathsf{Com})$ *is* computationally binding *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

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

where the game $\mathsf{Binding}_{\mathsf{HC},\mathcal{A}}(1^\lambda)$ *is defined in Figure 22.*

Suppose that $(\mathcal{M},+)$, $(\chi,+)$, and $(\mathcal{C},\cdot)$ are groups. Then HC is said to be homomorphic if the product of two commitments is a commitment to the sum of the two committed values.

Definition A.22 (Homomorphic Commitment Scheme)**.** *A commitment scheme is* homomorphic *if for all* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$*, all* $m,m' \in \mathcal{M}$ *and all* $r,r' \in \chi$

$$\mathsf{Com}_{\mathsf{pp}}(m+m',\, r+r') = \mathsf{Com}_{\mathsf{pp}}(m,r) \cdot \mathsf{Com}_{\mathsf{pp}}(m',r').$$

$\underline{\mathsf{Hiding}^{b}_{\mathsf{HC},\mathcal{A}}(1^\lambda)}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0,m_1) \leftarrow \mathcal{A}(\mathsf{pp})$
$C \leftarrow \mathsf{Com}_{\mathsf{pp}}(m_b)$
$b' \leftarrow \mathcal{A}(C)$
return b'

Figure 21: Security game for hiding of commitments.

$\underline{\mathsf{Binding}_{\mathsf{HC},\mathcal{A}}(1^\lambda)}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0,r_0,m_1,r_1) \leftarrow \mathcal{A}(\mathsf{pp})$
$b_0 := ((m_0,r_0) \neq (m_1,r_1))$
$b_1 := (\mathsf{Com}_{\mathsf{pp}}(m_0;r_0) = \mathsf{Com}_{\mathsf{pp}}(m_1;r_1))$
return $b_0 \wedge b_1$

Figure 22: Security game for binding of commitments.

B Tracking and Viewing

B.1 Definitions

We extend the model to cover transaction tracking and viewing, which are features supported by some previous schemes.

Definition B.1 (Extended RingCT)**.** *An extended RingCT is a RingCT scheme with a slightly extended syntax, where an original account* acc *is split into an extended form, consisting of an account* acc*, some tracking information* $\mathsf{info}_{\mathsf{Track}}$*, and some viewing information* $\mathsf{info}_{\mathsf{View}}$*.*

B.1.1 Trackability

CryptoNote [60], the predecessor of RingCT, introduces a feature called tracking. It allows a user to voluntarily delegate a tracking key to a trusted third party,[2] so that the latter can track incoming transactions on behalf of the user. This is particularly useful for a computationally constrained user as tracking incoming transactions requires monitoring all new messages posted on the public ledger.

Note that the trackability is not meant to work if the delegating user is malicious. Indeed, a user can easily avoid that incoming transactions are tracked by not delegating the tracking key or simply creating a second master public key. Tracking also relies on the well-formedness of the account acc in a transaction. As a spend proof does not necessarily guarantee the well-formedness of the entire account acc while still being considered valid, a "cheating" spender can easily help the receiver to avoid being tracked.

Definition B.2 (Trackability)**.** *An extended RingCT scheme is said to be trackable if the following holds:*

1. *There exists additionally a tuple of* PPT *algorithms* (TKGen,TKVerify,Track) *defined as follows:*

   $\underline{\mathsf{tsk} \leftarrow \mathsf{TKGen}(\mathsf{msk})}$*: The* tracking key generation *algorithm inputs a master secret key* msk*, and outputs a tracking key* tsk*.*

   $\underline{b \leftarrow \mathsf{TKVerify}(\mathsf{mpk},\mathsf{tsk})}$*: The* tracking key verification *algorithm inputs a master public key* mpk *and a tracking key* tsk*. It outputs a bit* b *indicating if* tsk *is a valid tracking key corresponding to the master public key* mpk*.*

[2] Different users can delegate to different third parties.

$b \leftarrow \mathsf{Track}(\mathsf{tsk},\mathsf{acc},\mathsf{info}_{\mathsf{Track}})$*: The* tracking *algorithm inputs a tracking key* tsk*, an account* acc*, and some tracking information. It outputs a bit* b *indicating if* acc *is an account generated from the master public key* mpk *corresponding to* tsk*.*

2. *For all* $\lambda,\alpha,\beta,\in\mathbb{N}$*, all* $\mathsf{pp}\in\mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$*, all* $(\mathsf{mpk},\mathsf{msk})\in\mathsf{SAKGen}(\mathsf{pp})$*, all* $\mathsf{tsk}\in\mathsf{TKGen}(\mathsf{msk})$*, it holds that* $\mathsf{TKVerify}(\mathsf{mpk},\mathsf{tsk})=1$*.*

3. *For all* $\lambda,\alpha,\beta,\in\mathbb{N}$*, all* $\mathsf{pp}\in\mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$*, all* $(\mathsf{mpk},\mathsf{tsk})$ *such that* $\mathsf{TKVerify}(\mathsf{mpk},\mathsf{tsk})=1$*, all* $a\in\{0,...,2^\beta-1\}$*, all* $(\mathsf{ck},\mathsf{acc},\mathsf{info}_{\mathsf{Track}},\mathsf{info}_{\mathsf{View}})\in\mathsf{OTAccGen}(\mathsf{mpk},a)$*, it holds that* $\mathsf{Track}(\mathsf{tsk},\mathsf{acc},\mathsf{info}_{\mathsf{Track}})=1$*.*

B.1.2 Viewability

RingCT extends the tracking capability such that the designated third party can also learn the amount to be received in an incoming transaction. Viewability is useful in scenarios where a user wishes to have incoming transactions to its address audited. For instance, a charity fund may want a third party or even the general public to audit the amount of donations that it receives, given that the donors are willing to disclose it. To allow more fine-grained tracking permissions, we call this new feature viewability. Similar to trackability, viewability is not meant to be a security feature.

Definition B.3 (Viewability)**.** *An extended RingCT scheme is said to be viewable if the following holds:*

1. *There exists additionally a tuple of* PPT *algorithms* $(\mathsf{VKGen},\mathsf{VKVerify},\mathsf{View})$ *defined as follows:*

   $\mathsf{vsk}\leftarrow\mathsf{VKGen}(\mathsf{msk})$*: The* viewing key generation *algorithm inputs a master secret key* msk*, and outputs a viewing key* vsk*.*

   $b\leftarrow\mathsf{VKVerify}(\mathsf{mpk},\mathsf{vsk})$*: The* viewing key verification *algorithm inputs a master public key* mpk *and a viewing key* vsk*. It outputs a bit* b *indicating if* vsk *is a valid viewing key corresponding to the master public key* mpk*.*

   $a\leftarrow\mathsf{View}(\mathsf{vsk},\mathsf{acc},\mathsf{info}_{\mathsf{View}})$*: The* view *algorithm inputs a viewing key* vsk*, an account* acc*, and some viewing information* $\mathsf{info}_{\mathsf{View}}$*. It outputs an amount* a *stored in the account* acc*.*

2. *For all* $\lambda,\alpha,\beta\in\mathbb{N}$*, all* $\mathsf{pp}\in\mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$*, all* $(\mathsf{mpk},\mathsf{msk})\in\mathsf{SAKGen}(\mathsf{pp})$*, all* $\mathsf{vsk}\in\mathsf{VKGen}(\mathsf{msk})$*, it holds that* $\mathsf{VKVerify}(\mathsf{mpk},\mathsf{vsk})=1$*.*

3. *For all* $\lambda,\alpha,\beta\in\mathbb{N}$*, all* $\mathsf{pp}\in\mathsf{Setup}(1^\lambda,1^\alpha,1^\beta)$*, all* $(\mathsf{mpk},\mathsf{vsk})$ *such that* $\mathsf{VKVerify}(\mathsf{mpk},\mathsf{vsk})=1$*, all* $a\in\{0,...,2^\beta-1\}$*, all* $(\mathsf{ck},\mathsf{acc},\mathsf{info}_{\mathsf{Track}},\mathsf{info}_{\mathsf{View}})\in\mathsf{OTAccGen}(\mathsf{mpk},a)$*, it holds that* $\mathsf{View}(\mathsf{vsk},\mathsf{acc},\mathsf{info}_{\mathsf{View}})=a$*.*

B.1.3 On Trackability and Viewability against Malicious Parties

The above definitions for traceability and viewability assume honest spenders and receivers, which is sufficient for the intended purposes as discussed above. For curiosity, we briefly discuss trackability and viewability against malicious spenders and receivers, and even trackers and viewers.

From a definitional point of view, defining such notions is not an issue. Indeed, similar notions are well-known in related primitives, such as the traceability and non-frameability of group signatures [5]. It is straightforward to adopt these definitions to RingCT.

Any schemes satisfying these strengthened notions however are likely much less efficient than those that do not. Intuitively, in a scheme which is trackable against malicious receivers, a user should not be able to spend from an account unless the account can be tracked by the tracker. Consider the following construction template. The user generates a tracking key and proves that the key is bound to its master public key. To spend, the user must prove that the source account is associated to some master public key. This convinces the tracker that the user is only able to spend from accounts that can be tracked. However, notice that the spender has to prove that the master public key associated to the source account is a member of the set of all master public keys. This is much more expensive than proving that the source account is a member of a set of ring accounts, and is against the design philosophy of RingCT.

Lastly, even if the RingCT scheme provides trackability and viewability against malicious parties, the users can still easily avoid being tracked or viewed by simply creating another master public key.

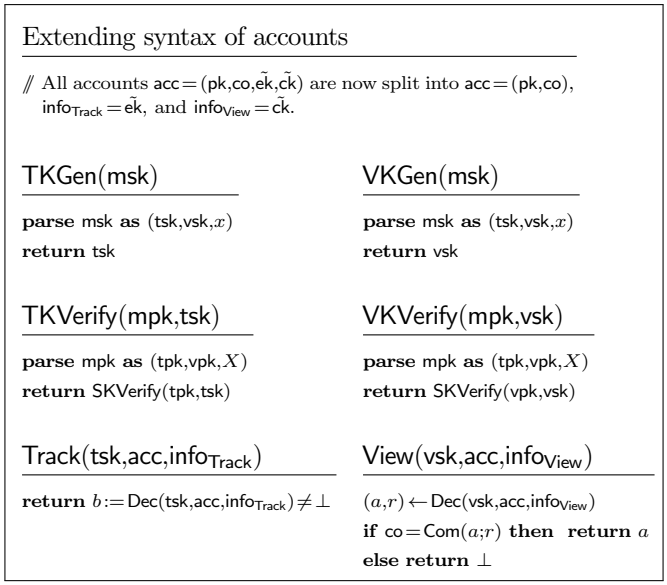

Figure 23: RingCT construction (extensions).

B.2 Extension to Construction

We extend our construction in Section 4 to support tracking and viewing. The descriptions of the algorithms are in Figure 23.

Extending syntax of accounts All accounts (output of $\mathsf{OTAccGen}$, input to Spend, $\mathsf{Receive}$, and Vf) $\mathsf{acc}=(\mathsf{pk},\mathsf{co},\tilde{\mathsf{ek}},\tilde{\mathsf{ck}})$ are now split into $\mathsf{acc}=(\mathsf{pk},\mathsf{co})$, $\mathsf{info}_{\mathsf{Track}}=\tilde{\mathsf{ek}}$, and $\mathsf{info}_{\mathsf{View}}=\tilde{\mathsf{ck}}$.

Tracking key generation Given the master secret key msk the tracking key generation algorithm parses msk as $(\mathsf{tsk},\mathsf{vsk},x)$ and returns tsk.

Track Given a tracking key tsk, an account acc, and some tracking information $\mathsf{info}_{\mathsf{Track}}$, the tracking algorithm checks if $\mathsf{Dec}(\mathsf{tsk},\mathsf{acc},\mathsf{info}_{\mathsf{Track}})\neq\perp$.

Viewing key generation Given the master secret key msk the viewing key generation algorithm parses msk as $(\mathsf{tsk},\mathsf{vsk},x)$ and returns vsk.

View Given a viewing key vsk, an account acc, and some viewing information $\mathsf{info}_v iew$, the viewing algorithm checks if $\mathsf{Dec}(\mathsf{vsk},\mathsf{acc},\mathsf{info}_{\mathsf{View}})=(a,r)\neq\perp$ and returns a if successful.

To prove security in the presence of these additional algorithms we define the oracles Figure 24.

C Security Proofs for Ω

We present here the full security analysis of our RingCT construction Ω in Section 4. In addition to the oracles given in Figure 1 we extend the security games for privacy and non-slanderability by providing the adversary with the additional oracles described in Figure 24 to model the implications of Tracking and Viewing on security.

C.1 Proof of Theorem 4.2 (Balance)

Proof. First we show that $\mathsf{CheckTag}$ is computationally binding. Suppose not, let $\mathcal{A}$ be a PPT adversary who outputs an account $\mathsf{acc}=(\mathsf{pk},\mathsf{co},\tilde{\mathsf{ek}},\tilde{\mathsf{ck}})$ and two distinct inputs $(\mathsf{sk},\mathsf{tag})$ and $(\mathsf{sk}',\mathsf{tag}')$ such that both satisfy the predicate $\mathsf{CheckTag}$. That is, $\mathsf{tag}=\mathsf{TagEval}(\mathsf{sk})$, $\mathsf{tag}'=\mathsf{TagEval}(\mathsf{sk}')$, and $\mathsf{pk}=\mathsf{TagKGen}(\mathsf{sk})=\mathsf{TagKGen}(\mathsf{sk}')$.

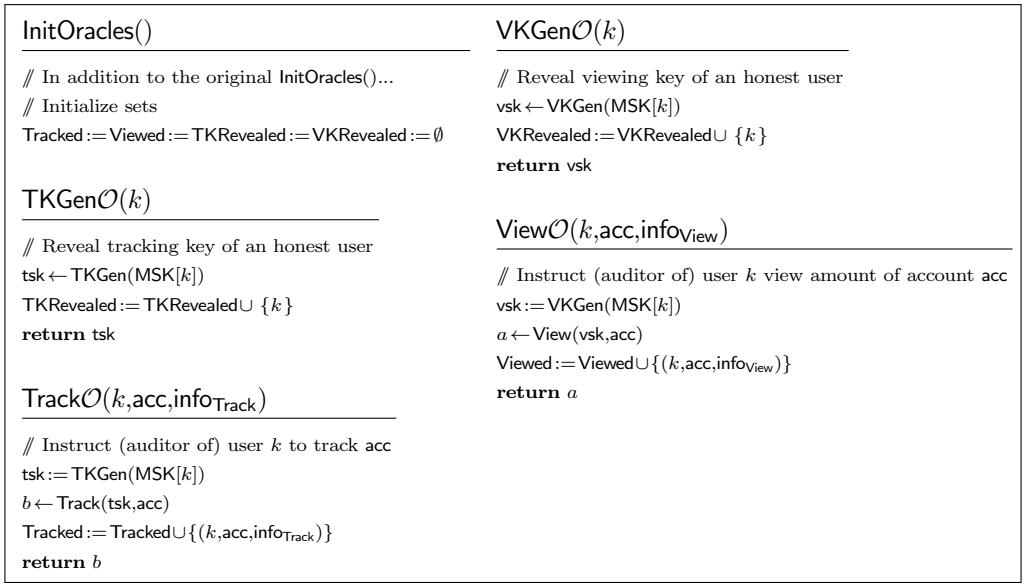

Figure 24: Oracles for track and view.

From the last relation, we must have $\mathsf{sk}=\mathsf{sk}'$ since $\mathsf{TagKGen}$ is bijective. It then follows that $\mathsf{tag}=\mathsf{tag}'$ since $\mathsf{TagEval}$ is deterministic. We therefore have $(\mathsf{sk},\mathsf{tag})=(\mathsf{sk}',\mathsf{tag}')$ which is a contradiction.

Next we show that $\mathsf{CheckAmount}$ is computationally binding. Suppose not, let $\mathcal{A}$ be a PPT adversary who outputs an account $\mathsf{acc}=(\mathsf{pk},\mathsf{co},\tilde{\mathsf{ek}},\tilde{\mathsf{ck}})$ and two distinct inputs (ck,a) and (ck',a') such that both satisfy the predicate $\mathsf{CheckAmount}$. That is, $\mathsf{co}=\mathsf{Com}(a;\mathsf{ck})=\mathsf{Com}(a';\mathsf{ck}')$. This directly contradicts with the binding property of HC.

We then construct an extractor $\mathcal{E}$ given any PPT adversary $\mathcal{A}$ which outputs (tx,σ) such that $\mathsf{Vf}(\mathsf{tx},\sigma)=1$ with non-negligible probability. By extractability of the SoK, there exists an efficient extraction algorithm $\mathsf{SoK}.\mathcal{E}_{\mathcal{A}}$ which extracts a witness wit for the statement $\mathsf{stmt}=\mathsf{stmt}(\mathsf{tx})$ with overwhelming probability. Parse stmt and wit as

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

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

and let $\mathcal{R}=\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}$, $\mathcal{S}=\left\{(j_i,\mathsf{ck}_i^{\mathcal{S}},a_i^{\mathcal{S}},\mathsf{sk}_i,\mathsf{tag}_i)\right\}_{i=1}^{|\mathcal{S}|}$, and $\mathcal{T}=\left\{(\mathsf{ck}_i^{\mathcal{T}},a_i^{\mathcal{T}},\mathsf{acc}_i^{\mathcal{T}})\right\}_{i=1}^{|\mathcal{T}|}$. Clearly $\mathsf{tx}=\mathsf{tx}(\mathcal{R},\mathcal{S},\mathcal{T},\mu)$. Furthermore, we have the following relations:

$$\forall i\in[|\mathcal{S}|],\begin{cases}\mathsf{pk}_{j_i}^{\mathcal{R}}=\mathsf{TagKGen}(x_i)\\ \mathsf{co}_{j_i}^{\mathcal{R}}=\mathsf{Com}(a_i^{\mathcal{S}};r_i^{\mathcal{S}})\\ \mathsf{tag}_i=\mathsf{TagEval}(x_i)\end{cases}$$

$$\forall i\in[|\mathcal{T}|],\begin{cases}\mathsf{co}_i^{\mathcal{T}}=\mathsf{Com}(a_i^{\mathcal{T}};r_i^{\mathcal{T}})\\ a_i^{\mathcal{T}}\in\left\{0,\ldots,2^{\beta}-1\right\}\end{cases}$$

$$\sum_{i\in[|\mathcal{S}|]}a_i^{\mathcal{S}}=\sum_{i\in[|\mathcal{T}|]}a_i^{\mathcal{T}}$$

This implies the following:

- $\forall i\in[|\mathcal{S}|]$, $\mathsf{CheckTag}(\mathsf{acc}_{j_i}^{\mathcal{R}},\mathsf{sk}_i,\mathsf{tag}_i)=1$
- $\forall i\in[|\mathcal{S}|]$, $\mathsf{CheckAmount}(\mathsf{acc}_{j_i}^{\mathcal{R}},\mathsf{ck}_i^{\mathcal{S}},a_i^{\mathcal{S}})=1$
- $\forall i\in[|\mathcal{T}|]$, $\mathsf{CheckAmount}(\mathsf{acc}_i^{\mathcal{T}},\mathsf{ck}_i^{\mathcal{T}},a_i^{\mathcal{T}})=1$

$\mathsf{Privacy}^{b}_{\Omega,\mathcal{A}}(1^{\lambda},1^{\alpha},1^{\beta})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda},1^{\alpha},1^{\beta})$, $\mathsf{InitOracles}()$

$\mathbb{O} := \{\mathsf{SAKGen}\mathcal{O},\mathsf{TKGen}\mathcal{O},\mathsf{VKGen}\mathcal{O},\mathsf{Spend}\mathcal{O},\mathsf{Track}\mathcal{O},\mathsf{View}\mathcal{O}\}$

$(I,J,\mathcal{R},\mathcal{S},\mathcal{T},\mu) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$

$\mathcal{S}_0 := \mathcal{S}_1 := \mathcal{S}$, $\mathcal{T}_0 := \mathcal{T}_1 := \mathcal{T}$

// Preparing honest spenders as instructed by adversary

parse I **as** $\left\{(s_i,\{j_{t,i},\mathsf{acc}^{\mathcal{S}}_{t,i}\}^{1}_{t=0})\right\}^{|I|}_{i=1}$

for $i \in [|I|]$ **do**

  for $t \in \{0,1\}$ **do**

    $(\mathsf{ck}^{\mathcal{S}}_{t,i},\mathsf{sk}^{\mathcal{S}}_{t,i},a^{\mathcal{S}}_{t,i},\mathsf{tag}_{t,i},\mathsf{info}^{\mathcal{S}}_{\mathsf{Track},t,i},\mathsf{info}^{\mathcal{S}}_{\mathsf{View},t,i}) := \mathsf{TryReceive}(\mathsf{acc}^{\mathcal{S}}_{t,i})$

    $\mathcal{R}[j_{t,i}] := \mathsf{acc}^{\mathcal{S}}_{t,i}$

    $\mathcal{S}_t[s_i] := (j_{t,i},\mathsf{ck}^{\mathcal{S}}_{t,i},\mathsf{sk}^{\mathcal{S}}_{t,i},a^{\mathcal{S}}_{t,i},\mathsf{tag}_{t,i})$

  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** $\left\{(d_j,\{k^{\mathcal{T}}_{t,j},a^{\mathcal{T}}_{t,j}\}^{1}_{t=0})\right\}^{|J|}_{j=1}$

for $j \in [|J|]$ **do**

  for $t \in \{0,1\}$ **do**

    $(\mathsf{ck}^{\mathcal{T}}_{t,j},\mathsf{acc}^{\mathcal{T}}_{t,j},\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},t,j},\mathsf{info}^{\mathcal{T}}_{\mathsf{View},t,j}) := \mathsf{OTAccGen}(\mathsf{MPK}[k^{\mathcal{T}}_{t,j}],a^{\mathcal{T}}_{t,j})$

    $\mathcal{T}_t[d_j] := (\mathsf{ck}^{\mathcal{T}}_{t,j},a^{\mathcal{T}}_{t,j},\mathsf{acc}^{\mathcal{T}}_{t,j},\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},t,j},\mathsf{info}^{\mathcal{T}}_{\mathsf{View},t,j})$

  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

$b_0 \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{tx}_b,\sigma_b)$

$b_1 := \left((\mathsf{TKRevealed} \cup \mathsf{VKRevealed}) \cap \left\{k^{\mathcal{T}}_{t,j} : t \in \{0,1\}, j \in [|J|]\right\} = \emptyset\right)$

$b_2 := \left(\mathsf{Tracked} \cap \left\{(k^{\mathcal{T}}_{t,j},\mathsf{acc}^{\mathcal{T}}_{b,j},\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},b,j}) : t \in \{0,1\}, j \in [|J|]\right\} = \emptyset\right)$

$b_3 := \left(\mathsf{Viewed} \cap \left\{(k^{\mathcal{T}}_{t,j},\mathsf{acc}^{\mathcal{T}}_{b,j},\mathsf{info}^{\mathcal{T}}_{\mathsf{View},b,j}) : t \in \{0,1\}, j \in [|J|]\right\} = \emptyset\right)$

return $b_0 \wedge b_1 \wedge b_2 \wedge b_3$

Figure 25: Privacy experiment (with tracking and viewing)

Furthermore, since $a^{\mathcal{T}}_i \in \{0,\ldots,2^{\beta}-1\}$ for all $i \in [|\mathcal{T}|]$, $|\mathcal{T}| \leq 2^{\alpha}$, and $\{0,\ldots,2^{\alpha+\beta}-1\} \subseteq \mathcal{M}$, $\sum_{i\in[|\mathcal{T}|]} a^{\mathcal{T}}_i$ has the same value when interpreted as an element in $\mathcal{M}$ and as an element in $\mathbb{Z}$. On the other hand, for each $i \in [|\mathcal{S}|]$, the value of $a^{\mathcal{S}}_i$ when interpreted as an element in $\mathbb{Z}$ must be greater than that when interpreted as an element in $\mathcal{M}$. Therefore $\sum_{i\in[|\mathcal{S}|]} a^{\mathcal{S}}_i = \sum_{i\in[|\mathcal{T}|]} a^{\mathcal{T}}_i$ (in $\mathcal{M}$) implies $\sum_{i\in[|\mathcal{S}|]} a^{\mathcal{S}}_i \geq \sum_{i\in[|\mathcal{T}|]} a^{\mathcal{T}}_i$ (in $\mathbb{Z}$).

Now, for any PPT adversary $\mathcal{A}$, let $\mathcal{E} = \mathsf{SoK}.\mathcal{E}_{\mathcal{A}}$. By the analysis above, it must hold that

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

□

C.2 Proof of Theorem 4.3 (Privacy)

We extend the privacy game from Figure 3 to include viewing and tracing.

Proof. We give a brief intuition of the proof.

Intuition We prove by hybrid arguments. To give some intuition as to how we progress through the hybrids: We start off in the real experiment for $\mathsf{Privacy}^0_{\Omega,\mathcal{A}}$ where the challenge bit b is set to 0. This means that the challenge spend proof given to the adversary is associated with the keys from the sets $\mathcal{S}_0$ and $\mathcal{T}_0$ (remember, two sets of source accounts $\mathcal{S}_0$ and $\mathcal{S}_1$ with $|\mathcal{S}_0| = |\mathcal{S}_1|$ and target accounts $\mathcal{T}_0$ and $\mathcal{T}_1$ with $|\mathcal{T}_0| = |\mathcal{T}_1|$ were specified by the adversary). We then transition through four hybrids where in the end the spend proof is independent of the sets $\mathcal{S}_0$ and $\mathcal{T}_0$, while the tags are generated corresponding to the keys in $\mathcal{S}_1$.

We now begin switching the target accounts from being associated with the set $\mathcal{T}_0$ to the set $\mathcal{T}_1$. We can do this without worrying about the spending keys because the spending keys are already delinked from the tags and are independent from the target accounts. After a sequence of hybrids we will completely switch to having the target accounts as being associated with set $\mathcal{T}_1$. Now we switch the spend proof from being simulated to being honestly generated from $(\mathcal{S}_1, \mathcal{T}_1)$. With this we finally end up with a hybrid that is identical to the real experiment for $\mathsf{Privacy}^1_{\Omega,\mathcal{A}}$ where the challenge bit b is set to 1. Proving the indistinguishability of the successive hybrids results in proving indistinguishability in the theorem. We define the hybrid experiments as follows:

Definition of main hybrid experiments We define experiments as follows:

$\underline{\mathsf{Hyb}_1}$ is identical to the privacy experiment $\mathsf{Privacy}^0_{\Omega,\mathcal{A}}$.

$\underline{\mathsf{Hyb}_2}$ differs from Hyb_1 in the way the signature σ is generated whenever the Spend algorithm is executed (both in the spend oracle and the challenge selection). Instead of using SoKSig, the challenger computes σ using the simulator $\mathcal{S}$ that is guaranteed to exist by the simulatability of SoK.

Note that in Hyb_2 the only information about b available to the adversary is the transaction

$$\mathsf{tx}_0 = \Big(\{\mathsf{acc}^{\mathcal{R}}_i\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{0,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}, \Big\{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},0,i}\Big\}_{i=1}^{|\mathcal{T}|}, \mu \Big).$$

The proof σ_0 does not give any extra information as it is computed from $\mathsf{stmt}(\mathsf{tx}_0)$. In the following hybrids, we gradually switch

$$\{\mathsf{tag}_{0,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|} \text{ and } \Big\{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},0,i}\Big\}_{i=1}^{|\mathcal{T}|}$$

to

$$\{\mathsf{tag}_{1,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|} \text{ and } \Big\{\mathsf{acc}^{\mathcal{T}}_{1,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},1,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},1,i}\Big\}_{i=1}^{|\mathcal{T}|}$$

respectively.
$\underline{\mathsf{Hyb}_3}$ differs from Hyb_2 in the way the proof is simulated. The proof is now simulated from

$$\Big(\{\mathsf{acc}^{\mathcal{R}}_i\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{1,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}, \Big\{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},0,i}\Big\}_{i=1}^{|\mathcal{T}|}, \mu \Big)$$

instead of from

$$\Big(\{\mathsf{acc}^{\mathcal{R}}_i\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{0,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}, \Big\{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},0,i}\Big\}_{i=1}^{|\mathcal{T}|}, \mu \Big).$$

$\underline{\mathsf{Hyb}_4}$ differs from Hyb_3 in the way the proof is simulated. The proof is now simulated from

$$\Big(\{\mathsf{acc}^{\mathcal{R}}_i\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{1,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}, \Big\{\mathsf{acc}^{\mathcal{T}}_{1,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},1,i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},1,i}\Big\}_{i=1}^{|\mathcal{T}|}, \mu \Big)$$

instead of from

$$\left(\left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{1,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}, \left\{\mathsf{acc}_{0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},0,i}^{\mathcal{T}}\right\}_{i=1}^{|\mathcal{T}|},\mu \right).$$

$\underline{\mathsf{Hyb}_5}$ differs from Hyb_4 in the way the signature σ is generated whenever the Spend algorithm is executed (both in the spend oracle and the challenge selection). Instead of using the simulator $\mathcal{S}$ of SoK, the challenger computes σ using SoKSig. Note that this hybrid is identical to the privacy experiment $\mathsf{Privacy}_{\Omega,\mathcal{A}}^{b}$ with $b=1$.

Proving indistinguishability of hybrids The proof proceeds as follows.

$\underline{\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2}$ The equivalence of Hyb_2 and Hyb_1 follows directly from the perfect simulatability of the signature of knowledge SoK.

$\underline{\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3}$ To show the indistinguishability between Hyb_2 and Hyb_3 we build a series of $|\mathcal{S}|+1$ sub-hybrids with $\mathsf{Hyb}_2 = \mathsf{Hyb}_{2,0}$ and $\mathsf{Hyb}_3 = \mathsf{Hyb}_{2,|\mathcal{S}|}$. In $\mathsf{Hyb}_{2,\ell}$, The proof is now simulated from

$$\left(\begin{array}{c} \left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{1,\mathsf{i}}\}_{i=1}^{\ell} \cup \{\mathsf{tag}_{0,\mathsf{i}}\}_{i=\ell+1}^{|\mathcal{S}|}, \\ \left\{\mathsf{acc}_{0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},0,i}^{\mathcal{T}}\right\}_{i=1}^{|\mathcal{T}|},\mu \end{array} \right).$$

It remains to show that $\mathsf{Hyb}_{2,\ell-1} \approx_c \mathsf{Hyb}_{2,\ell}$. Note that at every point in the experiment $\mathsf{TagEval}$ is called with a randomly chosen input of the form $x+s$, where s is an output of the random oracle. Therefore we can use the related-input pseudorandomness of the tag function as defined in Definition 4.1 to argue that $\mathsf{tag}_{0,\ell}$ and $\mathsf{tag}_{1,\ell}$ are both indistinguishable from a random tag in ψ, and hence indistinguishable from each other.

$\underline{\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4}$ To show the indistinguishability between Hyb_3 and Hyb_4 we build a series of $|\mathcal{T}|+1$ sub-hybrids with $\mathsf{Hyb}_3 = \mathsf{Hyb}_{3,0}$ and $\mathsf{Hyb}_4 = \mathsf{Hyb}_{3,|\mathcal{T}|}$. In $\mathsf{Hyb}_{3,\ell}$, The proof is now simulated from

$$\left(\begin{array}{c} \left\{\mathsf{acc}_i^{\mathcal{R}}\right\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_{1,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}, \\ \left\{\mathsf{acc}_{1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},1,i}^{\mathcal{T}}\right\}_{i=1}^{\ell} \cup \\ \left\{\mathsf{acc}_{0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},0,i}^{\mathcal{T}}\right\}_{i=\ell+1}^{|\mathcal{T}|},\mu \end{array} \right).$$

It remains to show that $\mathsf{Hyb}_{3,\ell-1} \approx_c \mathsf{Hyb}_{3,\ell}$. For this we define 4 sub-hybrids with $\mathsf{Hyb}_{3,\ell-1} = \mathsf{Hyb}_{3,\ell-1,0}$ and $\mathsf{Hyb}_{3,\ell} = \mathsf{Hyb}_{3,\ell-1,3}$. In $\mathsf{Hyb}_{3,\ell-1,1}$, the proof is simulated from

$$(\mathsf{acc}_{1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},0,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},0,i}^{\mathcal{T}}).$$

In $\mathsf{Hyb}_{3,\ell-1,2}$, the proof is simulated from

$$(\mathsf{acc}_{1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},0,i}^{\mathcal{T}}).$$

In $\mathsf{Hyb}_{3,\ell-1,3}$, the proof is simulated from

$$(\mathsf{acc}_{1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},1,i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},1,i}^{\mathcal{T}}).$$

Recall that $\mathsf{acc}_{0,\ell}^{\mathcal{T}}$ and $\mathsf{acc}_{1,\ell}^{\mathcal{T}}$ are of the form

$$\mathsf{acc}_{0,\ell}^{\mathcal{T}} = (\mathsf{pk}_{0,\ell}^{\mathcal{T}},\mathsf{co}_{0,\ell}^{\mathcal{T}}),$$
$$\mathsf{acc}_{1,\ell}^{\mathcal{T}} = (\mathsf{pk}_{1,\ell}^{\mathcal{T}},\mathsf{co}_{1,\ell}^{\mathcal{T}}).$$

Clearly, $\mathsf{pk}_{0,\ell}^{\mathcal{T}}$ and $\mathsf{pk}_{1,\ell}^{\mathcal{T}}$ are identically distributed. Moreover, since HC is perfectly hiding, $\mathsf{co}_{0,\ell}^{\mathcal{T}}$ and $\mathsf{co}_{1,\ell}^{\mathcal{T}}$ are also identically distributed. Therefore $\mathsf{Hyb}_{3,\ell-1,0} \equiv \mathsf{Hyb}_{3,\ell-1,1}$.

Next we argue that $\mathsf{Hyb}_{3,\ell-1,1} \approx_c \mathsf{Hyb}_{3,\ell-1,2}$. Note that in $\mathsf{Hyb}_{3,\ell-1,1}$, $\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i}$ is of the form

$$\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i} \leftarrow \mathsf{PKE.Enc}(\mathsf{tpk}_0, (\mathsf{pk}^{\mathcal{T}}_{1,\ell}, \mathsf{co}^{\mathcal{T}}_{1,\ell}), \mathsf{ek}_0)$$

for some tpk_0 and ek_0. In $\mathsf{Hyb}_{3,\ell-1,2}$, $\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},1,i}$ is of the form

$$\mathsf{info}^{\mathcal{T}}_{\mathsf{Track},1,i} \leftarrow \mathsf{PKE.Enc}(\mathsf{tpk}_1, (\mathsf{pk}^{\mathcal{T}}_{1,\ell}, \mathsf{co}^{\mathcal{T}}_{1,\ell}), \mathsf{ek}_1)$$

for some tpk_1 and ek_1. Note that by the definition of the privacy experiment, the adversary cannot succeed if it requests for the keys tsk_0 or tsk_1, or queries the tracking oracle on $((\mathsf{pk}^{\mathcal{T}}_{1,\ell}, \mathsf{co}^{\mathcal{T}}_{1,\ell}), \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},0,i})$ or $((\mathsf{pk}^{\mathcal{T}}_{1,\ell}, \mathsf{co}^{\mathcal{T}}_{1,\ell}), \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},1,i})$. We can thus use the IK-CCA and IND-CCA security of PKE to show that $\mathsf{Hyb}_{3,\ell-1,1} \approx_c \mathsf{Hyb}_{3,\ell-1,2}$.

The argument for $\mathsf{Hyb}_{3,\ell-1,2} \approx_c \mathsf{Hyb}_{3,\ell-1,3}$ is similar to that for $\mathsf{Hyb}_{3,\ell-1,1} \approx_c \mathsf{Hyb}_{3,\ell-1,2}$, except that now the adversary might be able to learn the amount hidden in $\mathsf{co}^{\mathcal{T}}_{1,\ell}$ using the $\mathsf{View}\mathcal{O}$ oracle. To do so, the adversary could query the $\mathsf{View}\mathcal{O}$ oracle on $((\mathsf{pk}, \mathsf{co}^{\mathcal{T}}_{1,\ell}), c)$ for some pk and c. If the oracle outputs some $a \neq \perp$, then the adversary could learn the amount hidden in $\mathsf{co}^{\mathcal{T}}_{1,\ell}$. We argue that this would only happen with negligible probability.

Note that since HC is perfectly hiding, $\mathsf{co}^{\mathcal{T}}_{1,\ell}$ contains no information about the amount that is committed. Therefore the probability that the $\mathsf{View}\mathcal{O}$ oracle outputs the "correct" amount is negligible. Suppose the oracle outputs a different amount than what is committed, then the oracle would have obtained a different opening to $\mathsf{co}^{\mathcal{T}}_{1,\ell}$ which breaks the binding property of HC. We can therefore conclude that the $\mathsf{View}\mathcal{O}$ oracle outputs $\perp$ on such inputs with overwhelming probability.

$\underline{\mathsf{Hyb}_4 \equiv \mathsf{Hyb}_5}$ The equivalence of Hyb_4 and Hyb_5 follows directly from the perfect simulatability of the signature of knowledge SoK. □

C.3 Proof of Theorem 4.4 (Non-slanderability)

We extend the non-slanderability game in Figure 4 to the one in Figure 26 which includes viewing and tracing.

$\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$, $\mathsf{InitOracles}()$

$(\mathsf{tx}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SAKGen}\mathcal{O}, \mathsf{Spend}\mathcal{O}, \mathsf{TKGen}\mathcal{O}, \mathsf{VKGen}\mathcal{O}}(\mathsf{pp})$

$\mathbf{parse}\ \mathsf{tx}^*\ \mathbf{as}\ \Big(\big\{\mathsf{acc}^{\mathcal{R}}_i\big\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \big\{\mathsf{acc}^{\mathcal{T}}_i, \mathsf{info}^{\mathcal{T}}_{\mathsf{Track},i}, \mathsf{info}^{\mathcal{T}}_{\mathsf{View},i}\big\}_{i=1}^{|\mathcal{T}|}, \mu \Big)$

$b_0 := \mathsf{Vf}(\mathsf{tx}^*, \sigma^*)$

$b_1 := ((\mathsf{tx}^*, \sigma^*) \notin \Sigma)$

$b_2 := \Big(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|} \cap \mathsf{Spent} \neq \emptyset \Big)$

$\mathbf{return}\ b_0 \wedge b_1 \wedge b_2$

Figure 26: Non-slanderability experiment (with tracking and viewing).

Proof. We prove by hybrid arguments. Consider a PPT adversary $\mathcal{A}$ who participates in the experiment $\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$. Without loss of generality, we assume that $\mathcal{A}$ makes at most q_{H}, q_A, and q_S queries to the H oracle, $\mathsf{SAKGen}\mathcal{O}$ oracle, and $\mathsf{Spend}\mathcal{O}$ oracle respectively. We define a hybrid experiment $\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ as follows:

$\underline{\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)}$ differs from $\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ in the way the $\mathsf{Spend}\mathcal{O}$ behaves. In $\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$, after running the line $\mathsf{tx} := \mathsf{tx}(\mathcal{R}, \mathcal{S}, \mathcal{T}, \mu)$ and $\mathsf{stmt} = \mathsf{stmt}(\mathsf{tx})$, the challenger runs the simulator $\mathcal{S}$ (which is guaranteed to exist by the simulatability property of SoK) to obtain σ.

We argue that the two hybrid experiments are functionally identical, *i.e.*, $\mathsf{NSland}_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) \equiv \mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$, which follows directly from the simulatability property of SoK.

Now, we show that if there exists a PPT adversary $\mathcal{A}$ such that

$$\mathsf{Pr}\Big[\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1\Big] > \frac{1}{p(\lambda)}$$

for some polynomial p, then we can construct another PPT adversary $\mathcal{B}$ that can find a pre-impage of Tag with non-negligible probability thus breaking the OneWay property of Tag.

The algorithm $\mathcal{B}$ simulates the $\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda,1^\alpha,1^\beta)$ experiment for $\mathcal{A}$ by generating $\mathsf{pp}\leftarrow\Omega.\mathsf{Setup}(1^\lambda)$ and simulating the oracles in the following way: $\mathcal{B}$ guesses an index $k^*\in[q_A]$ such that user k^* will be slandered and answers oracle queries as follows.

H oracle When the adversary queries with a master public key mpk and an ephemeral key ek as inputs for the j-th time, $\mathcal{B}$ queries $\mathsf{Tag}\mathcal{O}_x()$ and receives $(s_j,\mathsf{TagEval}(x+s_j))$. $\mathcal{B}$ programs $\mathsf{H}(\mathsf{mpk},\mathsf{ek}):=s_j$ and returns s_j. Since we assume that $\mathcal{A}$ only queries H at most q_H times, we have that $j\in[q_\mathsf{H}]$.

$\mathsf{SAKGen}\mathcal{O}$ oracle When the adversary queries the $\mathsf{SAKGen}\mathcal{O}$ oracle for the k-th time, if $k\neq k^*$, $\mathcal{B}$ generates the keys for the k-th user by running SAKGen algorithm honestly. If $k=k^*$, it generates $\mathsf{vpk},\mathsf{vsk},\mathsf{tsk},\mathsf{tpk}$ honestly, and set $\bar{\mathsf{pk}}:=X$ which is received from the OneWay challenger.

Spending oracle To simulate a spend proof, $\mathcal{B}$ needs to get hold of the transaction description of the form

$$\mathsf{tx}=\Big(\{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|},\{\mathsf{tag}_{0,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|},\Big\{\mathsf{acc}_i^{\mathcal{T}},\mathsf{info}_{\mathsf{Track},i}^{\mathcal{T}},\mathsf{info}_{\mathsf{View},i}^{\mathcal{T}}\Big\}_{i=1}^{|\mathcal{T}|},\mu\Big).$$

While most information of the transaction are provided by the adversary $\mathcal{A}$ explicitly, $\mathcal{B}$ has to compute the tags $\{\mathsf{tag}_{0,\mathsf{i}}\}_{i=1}^{|\mathcal{S}|}$ and the corresponding source accounts based on the instruction I provided by $\mathcal{A}$. Towards this end, for each tuple $(s_i,j_i,\mathsf{acc}_i)\in I$, $\mathcal{B}$ runs Receive on acc_i with each master secret key, except for that of user k^*. For the user k^*, it uses the viewing and tracking key to check the validity of acc_i, and if it is valid with respect to those keys, it generates the tag by querying the tagging oracle provided by the OneWay challenger.

With the description of the transaction, $\mathcal{B}$ can simulate the responses to the $\mathsf{Spend}\mathcal{O}$ queries honestly as in $\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda,1^\alpha,1^\beta)$, since SoK is simulatable.

$\mathcal{A}$ eventually outputs a slander (tx^*,σ^*). $\mathcal{B}$ parses tx^* as

$$\Big(\{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|},\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|},\{(\mathsf{acc}_i^{\mathcal{T}},\mathsf{info}_i)\}_{i=1}^{|\mathcal{T}|},\mu\Big).$$

Since $\mathsf{NSland}'_{\Omega,\mathcal{A}}(1^\lambda,1^\alpha,1^\beta)=1$, we know that $\mathsf{SoKVf}(\mathsf{stmt}^*,\sigma^*,\mathsf{tx}^*)=1$ and

$$\exists\mathsf{tag}_{i^*}\in\Big(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}\cap\mathsf{Spent}\neq\emptyset\Big),$$

where $\mathsf{stmt}^*=\mathsf{stmt}(\mathsf{tx}^*)$.

$\mathcal{B}$ exploits the PPT extractor $\mathcal{E}$ of SoK to extract a witness wit^* for the statement stmt^*. Parse wit^* as $\Big(\{(j_i,x_i,a_i^{\mathcal{S}},r_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|},\{(a_i^{\mathcal{T}},r_i^{\mathcal{T}})\}_{i=1}^{|\mathcal{T}|}\Big)$. Since $\mathsf{tag}_{i^*}\in\mathsf{Spent}$, it must be the case that in a previous received oracle query on some account acc' and help information info', the algorithm computes the tag tag' where $\mathsf{tag}_{i^*}=\mathsf{tag}'$. If $\mathcal{B}$ has guessed correctly that tag' was generated by user k^*, then tag' was set as $\mathsf{tag}'=\mathsf{TagEval}(x+s^*)$ where $s^*=\mathsf{H}(\mathsf{mpk},\mathsf{ek}')$ for some ek'. $\mathcal{B}$ then simply outputs x_{i^*} as a pre-image of Tag.

Note that $\mathcal{B}$ has at least $1/q_A$ probability of guessing k^* correctly, and the extractability property of SoK guarantees that $R(\mathsf{stmt}^*,\mathsf{wit}^*)=1$ with non-negligible probability. So x_{i^*} is a pre-image of Tag with non-negligible probability, which violates the security of Tag from Definition 4.1. □

D Security Proofs for Argument of Knowledge Construction

We present the formal security proofs of the argument of knowledge construction in Section 5.

D.1 Proof of Theorem 5.1 (Zero-Knowledge)

Proof. By inspection, $\mathcal{V}$ is public-coin and Π consists of 8 rounds. Π is also trivially perfectly complete. Next, we show that Π is perfect special honest-verifier zero-knowledge by constructing an efficient simulator $\mathcal{S}$.

Let $\mathsf{stmt}=(\vec{\mathbf{R}},\vec{\mathbf{C}}_{\mathcal{R}},\vec{\mathbf{T}},\vec{\mathbf{C}}_{\mathcal{T}})$ and the verifier public coin $(F,\vec{\mathbf{P}},\vec{\mathbf{G}}',\vec{\mathbf{H}},u,v,w,x,y,z)$ be provided by $\mathcal{A}$. Recall that $\mathcal{S}$ computes $\hat{\mathbf{Y}}$, $\hat{T}$, and $\vec{\mathbf{G}}_w$ as in Π. That is,

$$\hat{\mathbf{Y}}=\vec{\mathbf{R}}\circ\vec{\mathbf{C}}_{\mathcal{R}}^{\circ u}$$
$$\hat{T}=\vec{\mathbf{T}}^{u^2\vec{v}^{|\mathcal{S}|}}$$
$$\vec{\mathbf{G}}_w=((G\|H\|\hat{T}\|\hat{\mathbf{Y}})^{\circ w}\circ\vec{\mathbf{P}}\|\vec{\mathbf{G}}')$$

$\mathcal{S}$ then samples $A,T_2\leftarrow_{\$}\mathbb{G}$, $\tau,r\leftarrow_{\$}\mathbb{Z}_q$, and $\vec{\ell},\vec{r}\leftarrow_{\$}\mathbb{Z}_q^m$. It computes $t=\langle\vec{l},\vec{r}\rangle$. It then computes S and T_1 as follows:

$$S=(F^{-r}A\vec{\mathbf{G}}_w^{\vec{\alpha}-\vec{l}}\vec{\mathbf{H}}^{\vec{\beta}-\vec{\theta}^{\circ-1}\circ\vec{r}})^{-1/x}$$
$$T_1=(G^{\delta(y,z)-t}H^{-\tau}\vec{\mathbf{C}}_{\mathcal{T}}^{z^2\cdot\vec{y}^{|\mathcal{T}|}}T_2^{x^2})^{-1/x}$$

Finally, $\mathcal{S}$ outputs $(A,S,T_1,T_2,\tau,r,\vec{l},\vec{r},t)$.

Since all elements produced by $\mathcal{S}$ and those produced by Π are either independently randomly distributed or fully determined by the verification equations, they are identically distributed. □

D.2 Proof of Theorem 5.2 (Soundness)

To prove that Π is sound, or more precisely has extended-witness emulation, we first state some useful lemmas (Lemma 1 and Lemma 2). We establish the following notation.

For $i\in\{L,R\}$, let $\vec{\gamma}_i=(\gamma_{i,1},\gamma_{i,2},\gamma_{i,3},\vec{\gamma}_{i,4},\ldots,\vec{\gamma}_{i,9})\in\mathbb{Z}_q^m$ be variable vectors of the same format as $\vec{\mathbf{c}}_i$, *i.e.*, $\vec{\gamma}_{i,1},\vec{\gamma}_{i,2},\vec{\gamma}_{i,3}\in\mathbb{Z}$, $\vec{\gamma}_{i,4}\in\mathbb{Z}_q^{|\mathcal{R}|}$, $\vec{\gamma}_{i,5}\in\mathbb{Z}_q^{|\mathcal{R}||\mathcal{S}|}$, $\vec{\gamma}_{i,6}\in\mathbb{Z}_q^{\beta|\mathcal{T}|}$, $\vec{\gamma}_{i,7},\vec{\gamma}_{i,8}\vec{\gamma}_{i,9}\in\mathbb{Z}_q^{|\mathcal{S}|}$, and $\vec{\gamma}_{i,5}=\mathsf{vec}(\Gamma_i)$ and $\vec{\gamma}_{i,6}=\mathsf{vec}(\Delta_i)$ for some matrices $\Gamma_i\in\mathbb{Z}_q^{|\mathcal{S}|\times|\mathcal{R}|}$ and $\Delta_i\in\mathbb{Z}_q^{|\mathcal{T}|\times\beta}$.

We define a system of constraints $\mathsf{CS}=\mathsf{CS}[\vec{\mathbf{a}}^{\mathcal{T}},u,v]$ parameterized by $\vec{\mathbf{a}}_{\mathcal{T}},u,v$ as follows:

$\mathsf{CS}(\vec{\gamma}_L,\vec{\gamma}_R)=0\iff$

$$\begin{cases}
(\vec{\gamma}_{L,5},\vec{\gamma}_{L,6})\circ(\vec{\gamma}_{R,5},\vec{\gamma}_{R,6}) = \vec{0}^{|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|} & (14)\\
\vec{\gamma}_{R,9} = \vec{\gamma}_{L,9}^{\circ-1} & (15)\\
\Delta_L\vec{2}^{\beta} = \vec{\mathbf{a}}^{\mathcal{T}} & (16)\\
\Gamma_L\vec{1}^{|\mathcal{R}|} = \vec{1}^{|\mathcal{S}|} & (17)\\
\gamma_{L,1} = -\langle\vec{v}^{|\mathcal{S}|},u\cdot\vec{\gamma}_{L,7}+u^2\cdot\vec{\gamma}_{R,9}\rangle & (18)\\
\gamma_{L,2} = -\langle\vec{v}^{|\mathcal{S}|},\vec{\gamma}_{L,9}+u\cdot\vec{\gamma}_{L,8}\rangle & (19)\\
\vec{\gamma}_{L,4} = \vec{v}^{|\mathcal{S}|}\Gamma_L & (20)\\
\langle\vec{1}^{|\mathcal{S}|},\vec{\gamma}_{L,7}\rangle = \vec{1}^{|\mathcal{T}|}\Delta_L\vec{2}^{\beta} & (21)\\
(\vec{\gamma}_{R,5},\vec{\gamma}_{R,6}) = (\vec{\gamma}_{L,5},\vec{\gamma}_{L,6})-\vec{1}^{|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|} & (22)\\
\gamma_{L,3} = 1 & (23)
\end{cases}$$

Lemma 1. *Fix $q>2^\lambda$, $\vec{\mathbf{a}}^{\mathcal{T}}\in\mathbb{Z}_q^{|\mathcal{T}|}$, $u,v\in\mathbb{Z}_q$. Suppose there exists $\vec{\gamma}_L,\vec{\gamma}_R\in\mathbb{Z}_q^m$ such that, for $|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|$ different values of y we have $\mathsf{EQ}(\vec{\gamma}_L,\vec{\gamma}_R)=0$, then $\mathsf{CS}(\vec{\gamma}_L,\vec{\gamma}_R)=0$.*

Proof. Since $\mathsf{EQ}(\vec{\gamma}_L,\vec{\gamma}_R)=0$ for $|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|$ different values of y, then the following polynomials (in y) of degree at most $|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|-1$ each has $|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|$ different roots and therefore must be all equal to the

zero polynomial. These polynomials are:

$$\langle(\vec{\gamma}_{L,5},\vec{\gamma}_{L,6})\circ(\vec{\gamma}_{R,5},\vec{\gamma}_{R,6}),\vec{y}^{|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|}\rangle=0 \qquad \text{by } Eq\ (5)$$

$$\langle(\vec{\gamma}_{L,9}\circ\vec{\gamma}_{R,9})-\vec{1}^{|\mathcal{S}|},\vec{y}^{|\mathcal{S}|}\rangle=0 \qquad \text{by } Eq\ (6)$$

$$\langle\vec{y}^{|\mathcal{T}|},(\Delta_L\vec{2}^{\beta}-\vec{\mathbf{a}}^{\mathcal{T}})\rangle=0 \qquad \text{by } Eq\ (7)$$

$$(\gamma_{L,3}-1)y^{|\mathcal{S}|}+\langle\vec{\Gamma}_L\vec{1}^{|\mathcal{R}|}-\vec{1}^{|\mathcal{S}|},\vec{y}^{|\mathcal{S}|}\rangle=0 \qquad \text{by } Eq\ (8)$$

$$\gamma_{L,1}+\langle\vec{\gamma}_{L,7},u\cdot\vec{v}^{|\mathcal{S}|}\rangle+\langle\vec{\gamma}_{R,9},u^2\cdot\vec{v}^{|\mathcal{S}|}\rangle=0 \qquad \text{by } Eq\ (9)$$

$$\gamma_{L,2}+\langle\vec{\gamma}_{L,8},u\cdot\vec{v}^{|\mathcal{S}|}\rangle+\langle\vec{\gamma}_{L,9},\vec{v}^{|\mathcal{S}|}\rangle=0 \qquad \text{by } Eq\ (10)$$

$$\langle\vec{v}^{|\mathcal{S}|}\Gamma_L-\vec{\gamma}_{L,4},\vec{y}^{|\mathcal{R}|}\rangle=0 \qquad \text{by } Eq\ (11)$$

$$\vec{1}^{|\mathcal{T}|}\Delta_L\vec{2}^{\beta}-\langle\vec{1}^{|\mathcal{S}|},\vec{\gamma}_{L,7}\rangle=0 \qquad \text{by } Eq\ (12)$$

$$\langle(\vec{\gamma}_{L,5},\vec{\gamma}_{L,6})-(\vec{\gamma}_{R,5},\vec{\gamma}_{R,6})-\vec{1}^{|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|},\vec{y}^{|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|}\rangle=0 \qquad \text{by } Eq\ (13)$$

By comparing coefficients, we conclude that $\mathsf{CS}(\vec{\gamma}_L,\vec{\gamma}_R)=0$. □

The following is obtained by observing the system CS.

Lemma 2. *If* $\mathsf{CS}(\vec{\gamma}_L,\vec{\gamma}_R)=0$*, then:*

- *Each row of* Γ_L *is a unit vector of length* $|\mathcal{R}|$.
- *The* i*-row of* Δ_L *is the length-*β *binary representation of* $a_i^{\mathcal{T}}$.
- $\langle\vec{1}^{|\mathcal{S}|},\vec{\gamma}_{L,7}\rangle=\sum_{i\in[|\mathcal{T}|]}a_i^{\mathcal{T}}$.

Using the above lemmas, we now prove that Π has witness-extended emulation.

Proof. (Theorem 5.2) Assuming the discrete logarithm assumption holds over $\mathcal{G}$, by Theorem A.6, the interactive discrete logarithm assumption (as defined in Definition A.5) also holds over $\mathcal{G}$. The following is a straightforward corollary of Theorem A.6.

Corollary 1. *If there exists a* PPT *adversary* $\mathcal{A}$ *which does the following:*

1. *On input* $(\mathbb{G},q,G,H)$*, choose vector of group elements* $(\hat{T}\|\hat{\mathbf{Y}})$ *and an integer* w.
2. *Receive a uniformly random vector of group elements* $(F\|\vec{\mathbf{P}}\|\vec{\mathbf{G}}'\|\vec{\mathbf{H}})$ *of appropriate dimensions.*
3. *Produce a non-zero integer vector* $(r\|\vec{\mathbf{a}}\|\vec{\mathbf{b}})$ *such that* $I=F^r\vec{\mathbf{G}}_w^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$*, where* $\vec{\mathbf{G}}_w=((G\|H\|\hat{T}\|\hat{\mathbf{Y}})^w\circ\vec{\mathbf{P}}\|\vec{\mathbf{G}}')$.

then there exists a PPT *algorithm solving the discrete logarithm problem over* $\mathcal{G}$.

Proof. Suppose $\mathcal{A}$ exists, we construct an adversary $\mathcal{B}$ against the interactive discrete logarithm assumption. $\mathcal{B}$ receives $(\mathbb{G},q,G)$ and samples a random group element H. It passes $(\mathbb{G},q,G,H)$ to $\mathcal{A}$, and receives from the latter a vector of group elements $(\hat{T}\|\hat{\mathbf{Y}})$ and an integer w. $\mathcal{B}$ then sends to its challenger the vector $(G\|H\|\hat{T}\|\hat{\mathbf{Y}})$, and receives from the latter a uniformly random vector of group elements $(h\|\vec{\mathbf{P}}\|\vec{\mathbf{G}}'\|\vec{\mathbf{H}})$. $\mathcal{B}$ simply forwards $(h\|\vec{\mathbf{P}}\|\vec{\mathbf{G}}'\|\vec{\mathbf{H}})$ to $\mathcal{A}$, and receives a non-zero integer vector $(r\|\vec{\mathbf{a}}\|\vec{\mathbf{b}})$ such that $I=F^r\vec{\mathbf{G}}_w^{\vec{\mathbf{a}}}\vec{\mathbf{H}}^{\vec{\mathbf{b}}}$.

Let $\vec{\mathbf{a}}=(\vec{\mathbf{a}}_1,\vec{\mathbf{a}}_2)$ be of the appropriate dimensions. We can thus write

$$I=(G\|H\|\hat{T}\|\hat{\mathbf{Y}})^{w\cdot\vec{\mathbf{a}}_1}(h\|\vec{\mathbf{P}}\|\vec{\mathbf{G}}'\|\vec{\mathbf{H}})^{(r\|\vec{\mathbf{a}}\|\vec{\mathbf{b}})}$$

Since $(r\|\vec{\mathbf{a}}\|\vec{\mathbf{b}})$ is non-zero, $(w\cdot\vec{\mathbf{a}}_1\|r\|\vec{\mathbf{a}}\|\vec{\mathbf{b}})$ is a valid solution to the interactive discrete logarithm problem instance. $\mathcal{B}$ therefore outputs $(w\cdot\vec{\mathbf{a}}_1\|r\|\vec{\mathbf{a}}\|\vec{\mathbf{b}})$. □

With the above corollary, we can proceed to construct an extractor. Let $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,\mathcal{L})$ and $(\mathsf{stmt},\mathsf{wit})\leftarrow\mathcal{A}_2(\mathsf{pp})$. We construct an extractor $\mathcal{E}$ which, on input pp and stmt, outputs a transcript and, if the transcript is accepting, a witness wit' to the statement stmt. It is trivial for $\mathcal{E}$ to produce a transcript which is indistinguishable to that produced by $\langle\mathcal{P}^*(\mathsf{pp},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{pp},\mathsf{stmt})\rangle$ for any prover $\mathcal{P}^*$ as $\mathcal{E}$ is given

an oracle $\mathcal{O}=\langle\mathcal{P}^*(\mathsf{pp},\mathsf{stmt},\mathsf{wit}),\mathcal{V}(\mathsf{pp},\mathsf{stmt})\rangle$. We thus focus on describing how $\mathcal{E}$ can extract a witness wit' in the case where the transcript is accepting.

$\mathcal{E}$ runs $\mathcal{P}^*$ on 1 uniformly random chosen (u,v), 2 different values of w, $|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|$ different values of y, 9 different values of z, 3 different values of x. This results in $54(|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|)$ transcripts. Fix a particular choice of (w,y,z), $\mathcal{E}$ obtains three transcripts of the form $(A,S,T_1,T_2,\tau_{x_i},r_{x_i},\vec{l}_{x_i},\vec{r}_{x_i},t_{x_i})$ for $i=1,2,3$. Below, we show how the extractor can extract the discrete logarithm representations of A, S, T_1, and T_2.

Extracting A Choose $\kappa_{i=1,2}\in\mathbb{Z}_q$ such that $\sum_{i=1,2}\kappa_i=1$ and $\sum_{i=1,2}\kappa_i x_i=0$. This leads to the extraction of A by Equation (3) as

$$\begin{aligned}A&=F^{\sum_{i=1}^{2}\kappa_i r_{x_i}}\vec{\mathbf{G}}_w^{\sum_{i=1}^{2}\kappa_i\cdot\vec{l}_{x_i}-\vec{\alpha}}\vec{\mathbf{H}}^{\sum_{i=1}^{2}\kappa_i\cdot\vec{\theta}^{\circ-1}\circ\vec{r}_{x_i}-\vec{\beta}}\\&:=F^{r'_A}\vec{\mathbf{G}}_w^{\vec{\mathbf{c}}'_L}\vec{\mathbf{H}}^{\vec{\mathbf{c}}'_R}\end{aligned}$$

Note that r'_A, $\vec{\mathbf{c}}'_L$, and $\vec{\mathbf{c}}'_R$ depend on w. To obtain a discrete logarithm representation which is independent of w, $\mathcal{E}$ repeats the above for the other choice of w, which we denote by w'. With this additional transcript the extractor can extract A as $F^{r''_A}\vec{\mathbf{G}}_{w'}^{\vec{\mathbf{c}}''_L}\vec{\mathbf{H}}^{\vec{\mathbf{c}}''_R}$. Note that we now have two (possibly different) representations of A. Write $\vec{\mathbf{c}}'_L=(\vec{\mathbf{c}}'_{L,1}\|\vec{\mathbf{c}}'_{L,2})$ and $\vec{\mathbf{c}}''_L=(\vec{\mathbf{c}}''_{L,1}\|\vec{\mathbf{c}}''_{L,2})$ with appropriate dimensions. We have

$$F^{r'_A}\vec{\mathbf{G}}_w^{\vec{\mathbf{c}}'_L}\vec{\mathbf{H}}^{\vec{\mathbf{c}}'_R}=F^{r''_A}\vec{\mathbf{G}}_{w'}^{\vec{\mathbf{c}}''_L}\vec{\mathbf{H}}^{\vec{\mathbf{c}}''_R}$$

$$I=F^{r'_A-r''_A}(G\|H\|\hat{T}\|\hat{\mathbf{Y}})^{w\cdot\vec{\mathbf{c}}'_{L,1}-w'\cdot\vec{\mathbf{c}}''_{L,1}}\cdot(\vec{\mathbf{P}}\|\vec{\mathbf{G}}')^{\vec{\mathbf{c}}'_L-\vec{\mathbf{c}}''_L}\vec{\mathbf{H}}^{\vec{\mathbf{c}}'_R-\vec{\mathbf{c}}''_R}$$

We can assume $r'_A=r''_A$, $\vec{\mathbf{c}}'_L=\vec{\mathbf{c}}''_L$ and $\vec{\mathbf{c}}'_R=\vec{\mathbf{c}}''_R$. Suppose not, since $(F\|\vec{\mathbf{P}}\|\vec{\mathbf{G}}'\|\vec{\mathbf{H}})$ is chosen uniformly after fixing $(G\|H\|\hat{T}\|\hat{\mathbf{Y}})$, we would have an efficient algorithm against interactive discrete logarithm assumption. Note that $w\neq w'$. Let $\vec{\mathbf{c}}'_{L,1}=(\xi'\|\eta'\|\psi'\|\hat{\mathbf{e}}')$. We can obtain the following relations:

$$\begin{aligned}I&=(G\|H\|\hat{T}\|\hat{\mathbf{Y}})^{(w-w')\cdot\vec{\mathbf{c}}'_{L,1}}\\I&=(G\|H\|\hat{T}\|\hat{\mathbf{Y}})^{\vec{\mathbf{c}}'_{L,1}}\\I&=G^{\xi'}H^{\eta'}\hat{T}^{\psi'}\hat{\mathbf{Y}}^{\hat{\mathbf{e}}'}\end{aligned}\tag{24}$$

Extracting S Similar to the extraction of A, $\mathcal{E}$ can extract S from Equation (3) by sampling some $\kappa_{i=1,2}$ with $\sum_{i=1,2}\kappa_i=0$ and $\sum_{i=1,2}\kappa_i x_i=1$. This leads to

$$S=F^{\sum_{i=1}^{2}\kappa_i r_{x_i}}\vec{\mathbf{G}}_w^{\sum_{i=1}^{2}\kappa_i\vec{l}_{x_i}}\vec{\mathbf{H}}^{\sum_{i=1}^{2}\kappa_i\vec{\theta}^{\circ-1}\circ\vec{r}_{x_i}}=:F^{r'_S}\vec{\mathbf{G}}_w^{\vec{\mathbf{s}}'_L}\vec{\mathbf{H}}^{\vec{\mathbf{s}}'_R}.$$

Note that for fixed w, the above expressions of A and S hold for all choices of (x,y,z), for otherwise we would obtain a non-trivial discrete logarithm representation base $(h\|\vec{\mathbf{G}}_w\|\vec{\mathbf{H}})$, which violates the discrete logarithm assumption due to Corollary 1.

Putting the expressions of A and S back into Equation (3), it follows that for all challenges (x,y,z)

$$\begin{aligned}\vec{l}'_x&=\vec{\mathbf{c}}'_L+\vec{\theta}^{\circ-1}\circ(\vec{\omega}-\vec{\nu})+\vec{\mathbf{s}}'_L\cdot x\\\vec{r}'_x&=\vec{\theta}\circ(\vec{\mathbf{c}}'_R+\vec{\mathbf{s}}'_R\cdot x)+\vec{\mu}\end{aligned}$$

or otherwise we would have obtained a non-trivial representation of the identity element base $(h\|\vec{\mathbf{G}}_w\|\vec{\mathbf{H}})$, and thus violated the discrete logarithm assumption due to Corollary 1.

Extracting T_1 **and** T_2 To extract T_1, the extractor chooses $\kappa_{i=1,2,3}\leftarrow\mathbb{Z}_q$ with $\sum_{i=1}^{3}\kappa_i=0$, $\sum_{i=1}^{3}\kappa_i x_i=1$ and $\sum_{i=1}^{3}\kappa_i x_i^2=0$. Together with Equation (4) we have

$$T_1=G^{\sum_{i=1}^{3}\kappa_i t_{x_i}}H^{\sum_{i=1}^{3}\kappa_i\tau_{x_i}}=:G^{t'_1}H^{\tau'_1}.$$

Similarly, by choosing $\kappa'_{i=1,2,3} \leftarrow \mathbb{Z}_q$ with $\sum_{i=1}^{3}\kappa'_i = 0$, $\sum_{i=1}^{3}\kappa'_i x_i = 0$ and $\sum_{i=1}^{3}\kappa'_i x_i^2 = 1$ and Equation (4) we have

$$T_2 = G^{\sum_{i=1}^{3}\kappa'_i t_{x_i}} H^{\sum_{i=1}^{3}\kappa'_i \tau_{x_i}} =: G^{t'_2} H^{\tau'_2}.$$

Note that the above expressions of T_1 and T_2 hold for all challenge x, or otherwise we would have obtained a non-trivial discrete logarithm representation of the identity element base $(G\|H)$ which directly violates the discrete logarithm assumption.

Extracting $\vec{\mathbf{C}}_{\mathcal{T}}$ Fix a certain choice of (y,z). By putting the representations of T_1 and T_2 back to Equation (4), the extractor can find some (a,r) with $G^a H^r = \vec{\mathbf{C}}_{\mathcal{T}}^{\vec{y}^{|\mathcal{T}|}}$. Repeating this for $|\mathcal{T}|$ different y, and using the technique similar to that for extracting A, S, T_1, and T_2, $\mathcal{E}$ can extract $(a_i^{\mathcal{T}'}, r_i^{\mathcal{T}'})$ such that $\mathsf{co}_i^{\mathcal{T}} = G^{a_i^{\mathcal{T}'}} H^{r_i^{\mathcal{T}'}}$ for all $i \in [|\mathcal{T}|]$. In the following, we write $\vec{\mathbf{a}}^{\mathcal{T}'} := (a_1^{\mathcal{T}'} \| ... \| a_{|\mathcal{T}|}^{\mathcal{T}}{}')$ and $\vec{\mathbf{r}}^{\mathcal{T}'} := (r_1^{\mathcal{T}'} \| ... \| r_{|\mathcal{T}|}^{\mathcal{T}}{}')$.

Note that the above expressions of $\mathsf{co}_i^{\mathcal{T}}$ hold for all challenges (x,y,z), for otherwise we would obtain a non-trivial discrete logarithm representation base $(G\|H)$, which directly violates the discrete logarithm assumption.

Outputting Witness Write $\vec{\mathbf{c}}'_L$ as

$$\vec{\mathbf{c}}'_L = (\xi' \| \eta' \| \psi' \| \hat{\mathbf{e}}' \| \mathsf{vec}(\mathbf{E}') \| \mathsf{vec}(\mathbf{B}') \| \vec{\mathbf{a}}^{\mathcal{S}'} \| \vec{\mathbf{r}}^{\mathcal{S}'} \| \vec{\mathbf{x}}').$$

Together with the vectors $\vec{\mathbf{a}}^{\mathcal{T}'}$ and $\vec{\mathbf{r}}^{\mathcal{T}'}$ extracted above, $\mathcal{E}$ outputs the witness

$$\mathsf{wit}' = \left(\mathbf{E}', \vec{\mathbf{x}}', \vec{\mathbf{a}}^{\mathcal{S}'}, \vec{\mathbf{r}}^{\mathcal{S}'}, \mathbf{B}', \vec{\mathbf{a}}^{\mathcal{T}'}, \vec{\mathbf{r}}^{\mathcal{T}'}\right).$$

Showing Well-Formedness It remains to show that $\vec{\mathbf{c}}'_L$ (and $\vec{\mathbf{c}}'_R$) are well-formed, and hence wit' is a valid witness to stmt. Putting the expression of $\vec{\mathbf{C}}_{\mathcal{T}}$ back to Equation (4), we have

$$t'_x = z^2 \cdot \langle \vec{\mathbf{a}}^{\mathcal{T}'}, \vec{y}^{|\mathcal{T}|} \rangle + \delta + t'_1 x + t'_2 x^2$$

for all challenges (x,y,z), or we would have a discrete logarithm relation between G and H. Assume the following equations hold:

$$\begin{aligned} t'_0 &:= z^2 \cdot \langle \vec{\mathbf{a}}^{\mathcal{T}'}, \vec{y}^{|\mathcal{T}|} \rangle + \delta \\ l'(X) &:= \vec{\mathbf{c}}'_L + \vec{\alpha} + \vec{\mathbf{s}}'_L \cdot X \\ r'(X) &:= \vec{\theta} \circ (\vec{\mathbf{c}}'_R + \vec{\mathbf{s}}'_R \cdot X) + \vec{\mu} \\ t'(X) &:= \langle l'(X), r'(X) \rangle \end{aligned}$$

We have that for each choice of (y,z) the quadratic polynomial

$$\sum_{i=0}^{2} t'_i X^i - t'(X)$$

has at least three distinct roots and therefore is the zero polynomial. In particular, for all (y,z) it holds that $t'_0 = t'(0)$. Examining both sides of the equation, we have

$$\begin{aligned} t'_0 &= z^2 \cdot \langle \vec{\mathbf{a}}^{\mathcal{T}'}, \vec{y}^{|\mathcal{T}|} \rangle + \delta \\ &= z \cdot \langle \vec{1}^{|\mathcal{S}|}, \vec{y}^{|\mathcal{S}|} \rangle + z^2 \cdot \langle \vec{\mathbf{a}}^{\mathcal{T}'}, \vec{y}^{|\mathcal{T}|} \rangle + z^3 \cdot \langle \vec{1}^{|\mathcal{S}|+1}, \vec{y}^{|\mathcal{S}|+1} \rangle + \langle \vec{\alpha}, \vec{\mu} \rangle + \langle \vec{1}^m, \vec{\nu} \rangle, \end{aligned}$$

and

$$\begin{aligned} t'(0) &= \langle \vec{\mathbf{c}}'_L, \vec{\theta} \circ \vec{\mathbf{c}}'_R \rangle + \langle \vec{\mathbf{c}}'_L, \vec{\mu} \rangle + \langle \vec{\mathbf{c}}'_R, \vec{\omega} - \vec{\nu} \rangle + \langle \vec{\alpha}, \vec{\mu} \rangle \\ &= \langle \vec{\mathbf{c}}'_L, \vec{\theta} \circ \vec{\mathbf{c}}'_R \rangle + (\langle \vec{\mathbf{c}}'_L, \vec{\zeta} \rangle + \langle \vec{\mathbf{c}}'_R, \vec{\omega} \rangle) + \langle \vec{\mathbf{c}}'_L - \vec{\mathbf{c}}'_R, \vec{\nu} \rangle + \langle \vec{\alpha}, \vec{\mu} \rangle. \end{aligned}$$

The above implies that

$$\begin{aligned}
&z\cdot\langle\vec{1}^{|\mathcal{S}|},\vec{y}^{|\mathcal{S}|}\rangle+z^2\cdot\langle\vec{\mathbf{a}}^{\mathcal{T}'},\vec{y}^{|\mathcal{T}|}\rangle\ +z^3\cdot\langle\vec{1}^{|\mathcal{S}|+1},\vec{y}^{|\mathcal{S}|+1}\rangle\\
&\quad=\langle\vec{\mathbf{c}}'_L,\vec{\theta}\circ\vec{\mathbf{c}}'_R\rangle+(\langle\vec{\mathbf{c}}'_L,\vec{\zeta}\rangle+\langle\vec{\mathbf{c}}'_R,\vec{\omega}\rangle)\ +\langle\vec{\mathbf{c}}'_L-\vec{\mathbf{c}}'_R-\vec{1}^m,\vec{\nu}\rangle\\
&\quad=\sum_{i=0}^{1}z^i\langle\vec{\mathbf{c}}'_L,\vec{\mathbf{c}}'_R\circ\vec{\mathbf{v}}_i\rangle\ +\sum_{i=2}^{7}z^i\langle\vec{\mathbf{c}}'_L,\vec{\mathbf{v}}_i\rangle+z^4\langle\vec{\mathbf{c}}'_R,\vec{\mathbf{u}}_4\rangle\ +z^8\langle\vec{\mathbf{c}}'_L-\vec{\mathbf{c}}'_R-\vec{1}^m,\vec{\mathbf{v}}_8\rangle.
\end{aligned}$$

Since the above holds for 9 different values of z, the system of equations $\mathsf{EQ}[\vec{\mathbf{a}}^{\mathcal{T}'},u,v,y](\vec{\mathbf{c}}'_L,\vec{\mathbf{c}}'_R)$ as defined in Figure 11 is satisfied for $|\mathcal{R}||\mathcal{S}|+\beta|\mathcal{T}|$ different values of y. By Lemma 1, we have $\mathsf{CS}[\vec{\mathbf{a}}^{\mathcal{T}'},u,v](\vec{\mathbf{c}}'_L,\vec{\mathbf{c}}'_R)=0$. Then from the definition of $\mathsf{CS}[\vec{\mathbf{a}}^{\mathcal{T}'},u,v]$, and by Lemma 2, the following conditions must hold:

- Each row of $\mathbf{E}'$ is a unit vector of length $|\mathcal{R}|$.
- The ℓ-th row of $\mathbf{B}'$ is the length-β binary representation of $a_\ell^{\mathcal{T}'}$.
- $\sum_{\ell\in[|\mathcal{S}|]}a_\ell^{\mathcal{S}'}=\sum_{\ell\in[|\mathcal{T}|]}a_\ell^{\mathcal{T}'}$.

Furthermore, let $\mathsf{vec}(\mathbf{E}')=(\vec{\mathbf{e}}'_1,...,\vec{\mathbf{e}}'_{|\mathcal{S}|})$, we can write

$$\begin{aligned}
\xi'&=-\langle\vec{v}^{|\mathcal{S}|},u\cdot\vec{\mathbf{a}}^{\mathcal{S}'}+u^2\cdot\vec{\mathbf{x}}'^{\circ-1}\rangle\\
\eta'&=-\langle\vec{v}^{|\mathcal{S}|},\vec{\mathbf{x}}'+u\cdot\vec{\mathbf{r}}^{\mathcal{S}'}\rangle\\
\psi'&=1\\
\hat{\mathbf{e}}'&=\vec{v}^{|\mathcal{S}|}\mathbf{E}'=\sum_{\ell\in[|\mathcal{S}|]}v^{\ell-1}\cdot\vec{\mathbf{e}}'_\ell.
\end{aligned}$$

By Equation (24), we have

$$\begin{aligned}
I&=G^{\xi'}H^{\eta'}\hat{T}^{\psi'}\hat{\mathbf{Y}}^{\hat{\mathbf{e}}'}\\
&=G^{-\langle\vec{v}^{|\mathcal{S}|},u\cdot\vec{\mathbf{a}}^{\mathcal{S}'}+u^2\cdot\vec{\mathbf{x}}'^{\circ-1}\rangle}H^{-\langle\vec{v}^{|\mathcal{S}|},\vec{\mathbf{x}}'+u\cdot\vec{\mathbf{r}}^{\mathcal{S}'}\rangle}\prod_{\ell\in[|\mathcal{S}|]}\mathsf{tag}_\ell^{v^{\ell-1}u^2}\\
&\qquad(\vec{\mathbf{R}}\circ\vec{\mathbf{C}}_{\mathcal{R}}^{\circ u})^{\sum_{\ell\in[|\mathcal{S}|]}v^{\ell-1}\cdot\vec{\mathbf{e}}'_\ell}\\
&=\prod_{\ell\in[|\mathcal{S}|]}G^{-(a_\ell^{\mathcal{S}'})uv^{\ell-1}}\prod_{\ell\in[|\mathcal{S}|]}G^{\left(\frac{-1}{x'_\ell}\right)u^2v^{\ell-1}}\\
&\qquad\prod_{\ell\in[|\mathcal{S}|]}H^{-(x'_\ell)v^{\ell-1}}\prod_{\ell\in[|\mathcal{S}|]}H^{-(r_\ell^{\mathcal{S}'})uv^{\ell-1}}\prod_{\ell\in[|\mathcal{S}|]}\mathsf{tag}_\ell^{u^2v^{\ell-1}}\\
&\qquad\prod_{\ell\in[|\mathcal{S}|]}(\vec{\mathbf{R}}^{\vec{\mathbf{e}}_\ell})^{v^{\ell-1}}\prod_{\ell\in[|\mathcal{S}|]}(\vec{\mathbf{C}}_{\mathcal{R}}^{\vec{\mathbf{e}}_\ell})^{uv^{\ell-1}}\\
&=\prod_{\ell\in[|\mathcal{S}|]}(H^{-x_{\ell'}}\vec{\mathbf{R}}^{\vec{\mathbf{e}}_\ell})^{v^{\ell-1}}\\
&\qquad\prod_{\ell\in[|\mathcal{S}|]}(G^{-a_\ell^{\mathcal{S}'}}H^{-r_\ell^{\mathcal{S}'}}\vec{\mathbf{C}}_{\mathcal{R}}^{\vec{\mathbf{e}}_\ell})^{uv^{\ell-1}}\\
&\qquad\prod_{\ell\in[|\mathcal{S}|]}(G^{\frac{-1}{x'_\ell}}\mathsf{tag}_\ell)^{u^2v^{\ell-1}}.
\end{aligned}$$

The last equality can be viewed as an evaluation of a degree-$(|\mathcal{S}|+1)$ polynomial (in the exponent) at a random point (u,v) to zero. By the Schwartz–Zippel lemma, the probability that this happens when the polynomial is non-zero is bounded by $\frac{|\mathcal{S}|+1}{q}$ which is negligible given $q>2^\lambda$. We can therefore assume that the polynomial

is always zero. That is, for all $\ell \in [|\mathcal{S}|]$, the following equations hold:

$$\begin{aligned} \vec{\mathbf{R}}^{\vec{\mathbf{e}}_\ell} &= H^{x'_\ell} \\ \vec{\mathbf{C}}_{\mathcal{R}}^{\vec{\mathbf{e}}_\ell} &= G^{a_\ell^{\mathcal{S}'}} H^{r_\ell^{\mathcal{S}'}} \\ \mathsf{tag}_\ell &= G^{1/x'_\ell} \end{aligned}$$

To conclude, wit' extracted by $\mathcal{E}$ is indeed a valid witness corresponding to stmt. □

E Detailed Instantiation

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

E.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 E.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.*

Proof. We prove related-input one-wayness and related-input pseudorandomness separately.

Related-input one-wayness Suppose Tag is not related-input one-way, let $\mathcal{A}$ be a PPT adversary for which

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

i.e. $\mathcal{A}$ can guess a preimage for $\mathsf{TagEval}(x + s^*)$. Without loss of generality, we assume $\mathcal{A}$ makes at most $\ell - 1$ queries to the oracle $\mathsf{Tag}\mathcal{O}_x$. We construct an adversary $\mathcal{B}$ against ℓ-DL from Definition A.1 with input $(\mathbb{G}, q, G, ..., G^{x^\ell})$ as follows:

- Sample $\alpha, s^*, s_1, ..., s_{\ell-1} \leftarrow \mathbb{Z}_q$.
- Denote symbolically the polynomial

$$p(X) := (X + s^*) \prod_{i=1}^{\ell-1} (X + s_i) = \sum_{i=1}^{\ell} a_i X^i.$$

- Set $T := G^{p(X)} = G^{(x+s^*)\prod_{i=1}^{\ell-1}(x+s_i)}$, $H := G^\alpha$.
- Set $\mathsf{pp} = (\mathbb{G}, q, T, H)$, $\mathsf{pk}^* := (G^x)^\alpha$, and $\mathsf{tag}^* = T^{\frac{1}{x+s^*}}$. Note that

$$T^{\frac{1}{x+s^*}} = G^{\frac{(x+s^*)\prod_{i=1}^{\ell-1}(x+s_i)}{(x+s^*)}}.$$

- Run $\mathcal{A}$ on $(\mathsf{pp}, \mathsf{pk}^*, s^*, \mathsf{tag}^*)$. Note that since $G, G^x, ..., G^{x^\ell}$ are not given to $\mathcal{A}$, pp generated this way has the same distribution as those generated using Setup.
- On the j-th query to $\mathsf{Tag}\mathcal{O}_x$ return $T^{\frac{1}{x+s_j}}$. Note that

$$T^{\frac{1}{x+s_j}} = G^{\frac{(x+s^*)\prod_{i=1}^{\ell-1}(x+s_i)}{(x+s_j)}}.$$

- Eventually, $\mathcal{A}$ returns x'. $\mathcal{B}$ outputs $x = x' - s^*$.

With non-negligible probability, $\mathcal{A}$ is successful in breaking the related-input one-wayness of Tag, *i.e.*, $x' = x + s^*$. Then, with the same non-negligible probability, $\mathcal{B}$ recovers x correctly and solve the ℓ-DL instance, violating the parametrized discrete logarithm assumption.

Related-input pseudorandomness Suppose Tag is not related-input pseudorandom, let $\mathcal{A}$ be a PPT adversary for which

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

holds, *i.e.* it can distinguish $\mathsf{Tag}(x+s^*)$ from a random value $y_1\leftarrow_{\$}\mathbb{G}$ given s^*.

We construct an adversary $\mathcal{B}$ against SDDHI from Definition A.3, which uses $\mathcal{A}$ as follows:

- On input $(\mathbb{G},q,G,G^x)$ sample $s^*\leftarrow\mathbb{Z}_q$ and output s^* to get a challenge y_b.
- Sample $\alpha\leftarrow\mathbb{Z}_q$, set $\mathsf{pp}=(\mathbb{G},q,G,G^\alpha)$, and run $\mathcal{A}$ on $(\mathsf{pp},(G^x)^\alpha,s^*,y_b)$.
- Upon receiving a query to $\mathsf{Tag}\mathcal{O}_x$ from $\mathcal{A}$, sample $s\leftarrow\mathbb{Z}_q$ and return $(s,\mathcal{O}_x(s))$.
- Eventually, $\mathcal{A}$ outputs b' which is also output by $\mathcal{B}$.

Note that if $\mathcal{B}$ is participating in $\mathsf{SDDHI}^0_{\mathcal{B}}$, then it simulates the environment of $\mathsf{PR}^0_{\mathcal{A}}$ perfectly. Likewise, if $\mathcal{B}$ is participating in $\mathsf{SDDHI}^1_{\mathcal{B}}$, then it simulates the environment of $\mathsf{PR}^1_{\mathcal{A}}$ perfectly. Furthermore, with overwhelming probability, we have $s\neq s^*$. Therefore, for each $b\in\{0,1\}$, the probability that $\mathcal{B}$ outputs 1 in $\mathsf{SDDHI}^b_{\mathcal{B}}$ is negligibly close to that of $\mathcal{A}$ outputting 1 in $\mathsf{PR}^1_{\mathcal{A}}$. Hence,

$$\left|\mathsf{Pr}\left[\mathsf{SDDHI}^0_{\mathcal{B}}(\mathbb{G},q,G)=1\right]-\mathsf{Pr}\left[\mathsf{SDDHI}^1_{\mathcal{B}}(\mathbb{G},q,G)=1\right]\right|>\mathsf{negl}\,(\lambda)$$

which violates the SDDHI assumption. □

E.2 Homomorphic Commitments

The Pedersen commitment [47] can commit to a vector of messages $\mathbf{m}=(m_1,...,m_n)\in\mathbb{Z}_q^n$ by picking group elements $G_1,...,G_n\leftarrow\mathbb{G}$ and computing $\mathsf{Com}_{\mathsf{pp}}(\mathbf{m};r):=H^r\prod_{i=1}^n G_i^{m_i}$. The Pedersen commitment is naturally homomorphic. This commitment scheme is perfectly hiding, and it is computationally binding under the discrete logarithm assumption.

E.3 Labeled Public-Key Encryption

The Elliptic Curve Integrated Encryption Scheme (ECIES) [54] is a practical hybrid encryption scheme on elliptic curves. We provide below an abstract description of a labeled variant over generic groups. Below, let $H:\{0,1\}^*\to\{0,1\}^{2\lambda}$ 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 $\{0,1\}^\lambda$.

$\underline{\mathsf{Setup}(1^\lambda)}$: The setup algorithm takes as input the security parameter $1^\lambda$and outputs public parameters pp consisting of a description of a cyclic group $\mathbb{G}$ of order q together with a group elements $G\in\mathbb{G}$.

$\underline{\mathsf{KGen}(\mathsf{pp})}$: The key generation algorithm takes as input the public parameters pp. It 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.

$\underline{\mathsf{Enc}(\mathsf{pk},\tau,m)}$: The encryption algorithm takes as inputs the public key pk, a label τ, and a message $m\in\mathbb{G}$. It 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 $c:=(R,e,\sigma)$.

$\underline{\mathsf{Dec}(\mathsf{sk},\tau,c)}$: The decryption algorithm takes as input the secret key sk, a label τ, and a ciphertext c. It computes $P:=R^x$ and $(\mathsf{sk}_{\mathsf{SKE}},\mathsf{sk}_{\mathsf{MAC}}):=H(P,\tau)$, and 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 (Definition A.2) 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 [18, 54]; this can easily be extended to key-privacy (IK-CCA [6]).

F Adaption to Monero

Our RingCT scheme Omniring presented in the body of the paper is incompatible with Monero due to the difference in the formats of tags. In Omniring, the tag for an account with public key $\mathsf{pk}=H^x$ is given by $\mathsf{tag}=G^{x^{-1}}$. On the other hand, the corresponding tag in Monero would be $\mathsf{tag}=\mathsf{H}(\mathsf{pk})^x$ for some hash function H. If Monero were to adopt Omniring while keeping the current set of unspent transaction outputs, a spender who has already spent from an account with $\mathsf{tag}=\mathsf{H}(\mathsf{pk})^x$ would be able to spend from that account again using a transaction with $\mathsf{tag}=G^{x^{-1}}$, which would wrongly not be rejected as a double-spend.

To resolve this issue and make Omniring usable in Monero, we describe slight changes to the instantiations of the tagging scheme and the argument system, which ensure that tags have the same format as those currently used in Monero.

F.1 Tagging Scheme

The tagging scheme $\mathsf{Tag}_{\mathrm{mj}}=(\mathsf{TagSetup}_{\mathrm{mj}},\mathsf{TagKGen}_{\mathrm{mj}},\mathsf{TagEval}_{\mathrm{mj}})$ in Monero is as follows. $\mathsf{TagSetup}_{\mathrm{mj}}$ chooses a hash function $\mathsf{H}:\mathbb{G}\to\mathbb{G}$ (modeled as a random oracle) which maps group elements to group elements, and a group element H which is shared with the commitment scheme. On input $x\in\mathbb{Z}_q$, $\mathsf{TagKGen}_{\mathrm{mj}}$ outputs H^x. On input $x\in\mathbb{Z}_q$, $\mathsf{TagEval}_{\mathrm{mj}}$ outputs $\mathsf{H}(H^x)^x$.

We show that Tag satisfies (related-input) one-waynesss and pseudorandomness.

Theorem F.1. *If the discrete logarithm assumption holds over $\mathcal{G}=(\mathbb{G},q,G)$, then the tagging scheme $\mathsf{Tag}_{\mathrm{mj}}$ instantiated as in Monero is a secure tagging scheme according to Definition 4.1 in the random oracle model.*

Proof. We prove related-input one-wayness and related-input pseudorandomness separately.

Related-input one-wayness Suppose $\mathsf{Tag}_{\mathrm{mj}}$ is not related-input one-way, let $\mathcal{A}$ be a PPT adversary for which

$$\mathsf{Pr}\big[\mathsf{OneWay}_{\mathcal{A}}(1^\lambda)=1\big]>\mathsf{negl}\,(\lambda),$$

i.e. $\mathcal{A}$ can guess a preimage for $\mathsf{TagEval}_{\mathrm{mj}}(x+s^*)$. Without loss of generality, we assume $\mathcal{A}$ makes at most q_t queries to the oracle $\mathsf{Tag}\mathcal{O}_x$ and q_h queries to the random oracle. We construct an adversary $\mathcal{B}$ against the discrete logarithm assumption (1-DL) with input $(\mathbb{G},q,G,G^x)$ as follows:

- Sample $s^*,a^*\leftarrow\mathbb{Z}_q$.
- Compute $G^{x+s^*}=G^x\cdot G^{s^*}$ and set $H(G^{x+s^*})=G^{a^*}$.
- Compute the challenge tag $\mathsf{tag}^*=G^{(x+s^*)\cdot a^*}=(G^x)^{a^*}\cdot G^{s^*\cdot a^*}$.
- Set $\mathsf{pp}=(\mathbb{G},q,G)$ and run $\mathcal{A}$ on $(\mathsf{pp},G^x,s^*,\mathsf{tag}^*)$.
- On the j-th query to $\mathsf{Tag}\mathcal{O}_x$, sample $s_j,a_j\leftarrow\mathbb{Z}_q$ and set $H(G^{x+s_j})=G^{a_j}$. Then return $(s_j,G^{(x+s_j)\cdot a_j})$.
- On querying the random oracle with input X, check if $X=G^{x+s_j}$ for some s_j where $H(G^{x+s_j})$ was set during the j-th query to the $\mathsf{Tag}\mathcal{O}_x$. If so, return the set value. If not, sample $a\leftarrow\mathbb{Z}_q$ and return G^a as the reply.
- Eventually, $\mathcal{A}$ returns x'. $\mathcal{B}$ outputs $x=x'-s^*$.

With non-negligible probability, $\mathcal{A}$ is successful in breaking the related-input one-wayness of $\mathsf{Tag}_{\mathrm{mj}}$, *i.e.*, $x'=x+s^*$. Then, with the same non-negligible probability, $\mathcal{B}$ recovers x correctly and solve the 1-DL instance, violating the discrete logarithm assumption.

Related-input pseudorandomness Suppose $\mathsf{Tag}_{\mathrm{ŋ}}$ is not related-input pseudorandom, let $\mathcal{A}$ be a PPT adversary for which

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

holds, *i.e.* it can distinguish $\mathsf{Tag}_{\mathrm{ŋ}}(x+s^*)$ from a random value $y_1 \leftarrow_{\$}\mathbb{G}$ given s^*.

We construct an adversary $\mathcal{B}$ against DDH from Definition A.4, which uses $\mathcal{A}$ as follows:

- On input $(\mathbb{G},q,G,G^x,G^y,Z)$, sample $s^*\leftarrow\mathbb{G}$ and set $H(G^{x+s^*})=H(G^x\cdot G^{s^*})=G^y$ and $\mathsf{tag}^*=Z\cdot(G^y)^{s^*}$.
- Set $\mathsf{pp}=(\mathbb{G},q,G)$ and run $\mathcal{A}$ on $(\mathsf{pp},G^x,s^*,\mathsf{tag}^*)$.
- On the j-th query to $\mathsf{Tag}\mathcal{O}_x$, sample $s_j,a_j\leftarrow\mathbb{Z}_q$ and set $H(G^{x+s_j})=G^{a_j}$. Then return $(s_j,G^{(x+s_j)\cdot a_j})$.
- On querying the random oracle with input X, check if $X=G^{x+s_j}$ for some s_j where $H(G^{x+s_j})$ was set during the j-th query to the $\mathsf{Tag}\mathcal{O}_x$. If so, return the set value. If not, sample $a\leftarrow\mathbb{Z}_q$ and return G^a as the reply.
- Eventually, $\mathcal{A}$ outputs b' which is also output by $\mathcal{B}$.

Note that if $\mathcal{B}$ is participating in $\mathsf{DDH}^0_{\mathcal{B}}$, then it simulates the environment of $\mathsf{PR}^0_{\mathcal{A}}$ perfectly. Likewise, if $\mathcal{B}$ is participating in $\mathsf{DDH}^1_{\mathcal{B}}$, then it simulates the environment of $\mathsf{PR}^1_{\mathcal{A}}$ perfectly. Therefore, for each $b\in\{0,1\}$, the probability that $\mathcal{B}$ outputs 1 in $\mathsf{DDH}^b_{\mathcal{B}}$ equals the probability of $\mathcal{A}$ outputting 1 in $\mathsf{PR}^b_{\mathcal{A}}$. Hence,

$$\left|\mathsf{Pr}\left[\mathsf{DDH}^0_{\mathcal{B}}(\mathbb{G},q,G)=1\right]-\mathsf{Pr}\left[\mathsf{DDH}^1_{\mathcal{B}}(\mathbb{G},q,G)=1\right]\right|>\mathsf{negl}\,(\lambda),$$

which violates the DDH assumption. □

F.2 Language for Spend Proofs

With the new instantiation of the tagging scheme, the language that needs to be proven by spenders changes slightly. Given a vector of public keys $\vec{\mathbf{R}}$, we define a new vector $\vec{\mathbf{H}}$ of hashes of public keys as

$$\vec{\mathbf{H}}:=\left(\mathsf{H}(\mathsf{pk}^{\mathcal{R}}_1),\ldots,\mathsf{H}(\mathsf{pk}^{\mathcal{R}}_{|\mathcal{R}|})\right).$$

The corresponding language is changed to the following:

$$\mathcal{L}_{\mathrm{ŋ}}[\mathbb{G},q,G,H]$$

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

F.3 Argument System

As in Section 5 we describe a protocol $\Pi_{\mathrm{ŋ}}$ with linear communication for the updated language $\mathcal{L}_{\mathrm{ŋ}}$. A protocol with logarithmic communication can be obtained using the same squashing technique presented in Section 5. The notation used in the protocol below is defined above (for $\vec{\mathbf{H}}$), and in Section 4.4, Figures 27 to 29 and Table 2.

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

$\underline{\langle \mathcal{P}_{\mathfrak{y}}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}),\mathcal{V}_{\mathfrak{y}}(\mathsf{crs},\mathsf{stmt})\rangle}$:
$\mathcal{V}_{\mathfrak{y}}$:

1. $u,v \leftarrow_{\$} \mathbb{Z}_q$
2. $F \leftarrow_{\$} \mathbb{G}$, $\vec{\mathbf{P}} \leftarrow_{\$} \mathbb{G}^{2+|\mathcal{R}|+|\mathcal{S}|}$, $\vec{\mathbf{G}}' \leftarrow_{\$} \mathbb{G}^{m-|\mathcal{R}|-|\mathcal{S}|-2}$, $\vec{\mathbf{H}} \leftarrow_{\$} \mathbb{G}^m$

$\mathcal{P}_{\mathfrak{y}} \leftarrow \mathcal{V}_{\mathfrak{y}} : u,v,F,\vec{\mathbf{P}},\vec{\mathbf{G}}',\vec{\mathbf{H}}$
$\mathcal{P}_{\mathfrak{y}},\mathcal{V}_{\mathfrak{y}}$:

1. $\hat{\mathbf{Y}} := \vec{\mathbf{R}} \circ \vec{\mathbf{C}}_{\mathcal{R}}^{\circ u} \circ \vec{\mathbf{H}}^{\circ u^2}$
2. $\hat{\mathbf{T}} := \vec{\mathbf{T}}^{\circ -u^2 \vec{v}^{|\mathcal{S}|}}$
3. For $w \in \mathbb{Z}_q$, denote

$$\vec{\mathbf{G}}_w := ((G \| H \| \hat{\mathbf{Y}} \| \hat{\mathbf{T}})^{\circ w} \circ \vec{\mathbf{P}} \| \vec{\mathbf{G}}') \tag{25}$$

$\mathcal{P}_{\mathfrak{y}}$:

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 $I = G^{\xi} H^{\eta} \hat{\mathbf{Y}}^{\hat{\mathbf{e}}} \hat{\mathbf{T}}^{\vec{\mathbf{x}}^{\circ -1}}$. Thus $A = F^{r_A} \vec{\mathbf{G}}_w^{\vec{\mathbf{c}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{c}}_R}$ for all $w \in \mathbb{Z}_q$.

$\mathcal{P}_{\mathfrak{y}} \rightarrow \mathcal{V}_{\mathfrak{y}} : A$
$\mathcal{V}_{\mathfrak{y}} : w \leftarrow_{\$} \mathbb{Z}_q$
$\mathcal{P}_{\mathfrak{y}} \leftarrow \mathcal{V}_{\mathfrak{y}} : w$
$\mathcal{P}_{\mathfrak{y}}$:

1. $r_S \leftarrow_{\$} \mathbb{Z}_q$, $\vec{\mathbf{s}}_L \leftarrow_{\$} \mathbb{Z}_q^m$, $\vec{\mathbf{s}}_R \leftarrow_{\$} \{\vec{\mathbf{s}} = (s_1,\ldots,s_m) \in \mathbb{Z}_q^m : \forall i \in [m],\ \vec{\mathbf{c}}_R[i] = 0 \implies s_i = 0\}$.
2. $S := F^{r_S} \vec{\mathbf{G}}_w^{\vec{\mathbf{s}}_L} \vec{\mathbf{H}}^{\vec{\mathbf{s}}_R}$.

$\mathcal{P}_{\mathfrak{y}} \rightarrow \mathcal{V}_{\mathfrak{y}} : S$
$\mathcal{V}_{\mathfrak{y}} : y,z \leftarrow_{\$} \mathbb{Z}_q$
$\mathcal{P}_{\mathfrak{y}} \leftarrow \mathcal{V}_{\mathfrak{y}} : y,z$
$\mathcal{P}_{\mathfrak{y}}$:

1. Define the following polynomials (in X):

$$\begin{aligned} l(X) &:= \vec{\mathbf{c}}_L + \vec{\alpha} + \vec{\mathbf{s}}_L \cdot X \\ r(X) &:= \vec{\theta} \circ (\vec{\mathbf{c}}_R + \vec{\mathbf{s}}_R \cdot X) + \vec{\mu} \\ t(X) &:= \langle l(X), r(X) \rangle = t_2 X^2 + t_1 X + t_0 \end{aligned}$$

   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. $T_1 := G^{t_1} F^{\tau_1}$, $T_2 := G^{t_2} F^{\tau_2}$

$\mathcal{P}_{\mathfrak{y}} \rightarrow \mathcal{V}_{\mathfrak{y}} : T_1,T_2$
$\mathcal{V}_{\mathfrak{y}} : x \leftarrow_{\$} \mathbb{Z}_q$
$\mathcal{P}_{\mathfrak{y}} \leftarrow \mathcal{V}_{\mathfrak{y}} : x$
$\mathcal{P}_{\mathfrak{y}}$:

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. $(\vec{l}, \vec{r}, t) := (l(x), r(x), t(x))$

$\mathcal{P}_{\text{ŋ}} \to \mathcal{V}_{\text{ŋ}} : \tau, r, \vec{l}, \vec{r}, t$
$\mathcal{V}_{\text{ŋ}}$: Check if the following relations hold:

$$t = \langle \vec{l}, \vec{r} \rangle \tag{26}$$

$$F^r \vec{\mathbf{G}}_w^{\vec{l}} \vec{\mathbf{H}}^{\vec{\theta}^{\circ -1} \circ \vec{r}} = A S^x \vec{\mathbf{G}}_w^{\vec{\alpha}} \vec{\mathbf{H}}^{\vec{\beta}} \tag{27}$$

$$G^t H^\tau = G^\delta \vec{\mathbf{C}}_{\mathcal{T}}^{z^2 \cdot \vec{y}^{|\mathcal{T}|}} T_1^x T_2^{x^2} \tag{28}$$

Theorem F.2. *The verifier $\mathcal{V}_{\text{ŋ}}$ is public-coin. $\Pi_{\text{ŋ}}$ is constant-round, perfectly complete, and perfect special honest-verifier zero-knowledge.*

Theorem F.3. *Assuming the discrete logarithm assumption holds over $\mathcal{G}$, $\Pi_{\text{ŋ}}$ has computational witness-extended emulation.*

The proofs of the above theorems are almost identical to those of Theorems 5.1 and 5.2 and are omitted.

Notation	*Description*
$\hat{\mathbf{Y}}=\hat{\mathbf{Y}}(u):=\vec{\mathbf{R}}\circ\vec{\mathbf{C}}_{\mathcal{R}}^{\circ u}\circ\vec{\mathbf{H}}^{\circ u^2}$	Vector of compressed public keys and coins with randomness $u\in\mathbb{Z}_q$
$\hat{\mathbf{T}}=\hat{\mathbf{T}}(u,v):=\vec{\mathbf{T}}^{\circ -u^2\vec{v}^{\lvert\mathcal{S}\rvert}}$	Scaled tag with randomness $u,v\in\mathbb{Z}_q$
$\xi=\xi(u,v):=-\langle\vec{v}^{\lvert\mathcal{S}\rvert},u\cdot\vec{\mathbf{a}}^{\mathcal{S}}\rangle$	Compressed secrets with randomness $u,v\in\mathbb{Z}_q$. Note that $(\xi,\eta,\hat{\mathbf{e}},\vec{\mathbf{x}}^{\circ -1})$ satisfies $I=G^{\xi}H^{\eta}\hat{\mathbf{Y}}^{\hat{\mathbf{e}}}\hat{\mathbf{T}}^{\vec{\mathbf{x}}^{\circ -1}}$.
$\eta=\eta(u,v):=-\langle\vec{v}^{\lvert\mathcal{S}\rvert},\vec{\mathbf{x}}+u\cdot\vec{\mathbf{r}}^{\mathcal{S}}\rangle$	
$\hat{\mathbf{e}}=\hat{\mathbf{e}}(v):=\vec{v}^{\lvert\mathcal{S}\rvert}\mathbf{E}$	
$\vec{\mathbf{c}}_L,\vec{\mathbf{c}}_R$	Encoding of witness by honest prover dependent on u and v, see Figure 27.
$m=2+\lvert\mathcal{R}\rvert+\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert+3\lvert\mathcal{S}\rvert$	Length of $\vec{\mathbf{c}}_L$ and $\vec{\mathbf{c}}_R$
$(\vec{\mathbf{v}}_0,...,\vec{\mathbf{v}}_8,\vec{\mathbf{u}}_5)=(\vec{\mathbf{v}}_0,...,\vec{\mathbf{v}}_8,\vec{\mathbf{u}}_5)(u,v,y)$	Constraint vectors parameterized by the randomness $u,v,y\in\mathbb{Z}_q$, see Figure 28.
$(\vec{\alpha},\vec{\beta},\vec{\delta},\vec{\theta},\vec{\zeta},\vec{\mu},\vec{\nu},\vec{\omega})$ $=(\vec{\alpha},\vec{\beta},\vec{\delta},\vec{\theta},\vec{\zeta},\vec{\mu},\vec{\nu},\vec{\omega})(u,v,y,z)$	Compressed constraint vectors parameterized by the randomness $u,v,y,z\in\mathbb{Z}_q$, see Figure 28 and Figure 29.
$\mathsf{EQ}=\mathsf{EQ}[\vec{\mathbf{a}}^{\mathcal{T}},u,v,y]$	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 2: Notation for signatures of knowledge construction (for Monero).

$$\begin{aligned}\vec{\mathbf{c}}_L &:= (\ \xi \parallel \eta \parallel \hat{\mathbf{e}} \parallel \vec{\mathbf{x}}^{\circ -1} \parallel \quad \mathsf{vec}(\mathbf{E}) \quad \parallel \quad \mathsf{vec}(\mathbf{B}) \quad \parallel \vec{\mathbf{a}}^{\mathcal{S}} \parallel \vec{\mathbf{r}}^{\mathcal{S}}\ )\\ \vec{\mathbf{c}}_R &:= (\ \vec{0}^{2+\lvert\mathcal{R}\rvert} \parallel \vec{\mathbf{x}} \parallel \mathsf{vec}(\mathbf{E})-\vec{1}^{\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert} \parallel \mathsf{vec}(\mathbf{B})-\vec{1}^{\beta\lvert\mathcal{T}\rvert} \parallel \vec{0}^{2\lvert\mathcal{S}\rvert}\ )\end{aligned}$$

Figure 27: Honest encoding of witness (for Monero).

$$\begin{bmatrix}\vec{\mathbf{v}}_0\\ \vec{\mathbf{v}}_1\\ \vec{\mathbf{v}}_2\\ \vec{\mathbf{v}}_3\\ \vec{\mathbf{v}}_4\\ \vec{\mathbf{v}}_5\\ \vec{\mathbf{v}}_6\\ \vec{\mathbf{v}}_7\\ \vec{\mathbf{v}}_8\\ \vec{\mathbf{u}}_5\end{bmatrix} := \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert} & \cdot & \cdot\\ \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{S}\rvert} & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{T}\rvert}\otimes\vec{2}^{\beta} & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{S}\rvert}\otimes\vec{1}^{\lvert\mathcal{R}\rvert} & \cdot & \cdot\\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\cdot\vec{v}^{\lvert\mathcal{S}\rvert} & \cdot\\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\cdot\vec{v}^{\lvert\mathcal{S}\rvert}\\ \cdot & \cdot & -\vec{y}^{\lvert\mathcal{R}\rvert} & \cdot & \vec{v}^{\lvert\mathcal{S}\rvert}\otimes\vec{y}^{\lvert\mathcal{R}\rvert} & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot & \cdot & \vec{1}^{\lvert\mathcal{T}\rvert}\otimes\vec{2}^{\beta} & -\vec{1}^{\lvert\mathcal{S}\rvert} & \cdot\\ \cdot & \cdot & \cdot & \cdot & \vec{y}^{\lvert\mathcal{R}\rvert\lvert\mathcal{S}\rvert+\beta\lvert\mathcal{T}\rvert} & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \vec{v}^{\lvert\mathcal{S}\rvert} & \cdot & \cdot & \cdot & \cdot\end{bmatrix}$$

Figure 28: Definitions of constraint vectors (for Monero). (Dots mean zeros.)

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

$$\vec{\nu}:=z^8\cdot\vec{\mathbf{v}}_8 \qquad\qquad \vec{\omega}:=z^5\cdot\vec{\mathbf{u}}_5$$

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

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

Figure 29: Definitions of constraint vectors (for Monero) (cont.).

$$\mathsf{EQ}(\vec{\gamma}_L,\vec{\gamma}_R)=0 \iff$$

$$\begin{cases}\langle\vec{\gamma}_L,\vec{\gamma}_R\circ\vec{\mathbf{v}}_0\rangle & =0 & (29)\\ \langle\vec{\gamma}_L,\vec{\gamma}_R\circ\vec{\mathbf{v}}_1\rangle & =\langle\vec{1}^{\lvert\mathcal{S}\rvert},\vec{y}^{\lvert\mathcal{S}\rvert}\rangle & (30)\\ \langle\vec{\gamma}_L,\vec{\mathbf{v}}_2\rangle & =\langle\vec{\mathbf{a}}^{\mathcal{T}},\vec{y}^{\lvert\mathcal{T}\rvert}\rangle & (31)\\ \langle\vec{\gamma}_L,\vec{\mathbf{v}}_3\rangle & =\langle\vec{1}^{\lvert\mathcal{S}\rvert},\vec{y}^{\lvert\mathcal{S}\rvert}\rangle & (32)\\ \langle\vec{\gamma}_L,\vec{\mathbf{v}}_4\rangle & =0 & (33)\\ \langle\vec{\gamma}_L,\vec{\mathbf{v}}_5\rangle+\langle\vec{\gamma}_R,\vec{\mathbf{u}}_5\rangle & =0 & (34)\\ \langle\vec{\gamma}_L,\vec{\mathbf{v}}_6\rangle & =0 & (35)\\ \langle\vec{\gamma}_L,\vec{\mathbf{v}}_7\rangle & =0 & (36)\\ \langle\vec{\gamma}_L-\vec{\gamma}_R-\vec{1}^{m},\vec{\mathbf{v}}_8\rangle & =0 & (37)\end{cases}$$

Figure 30: A system of equations guaranteeing the integrity of the encoding of witness (for Monero).

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, Haixia Shi, and Chong Zhang. 2005. Foundations of group signatures: the case of dynamic groups. In *CT-RSA 2005*.

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

[7] 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*.

[8] Dan Boneh and Matthew K. Franklin. 2001. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*.

[9] 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*.

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

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

[12] 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*.

[13] 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*.

[14] Melissa Chase and Anna Lysyanskaya. 2006. On signatures of knowledge. In *CRYPTO 2006*.

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

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

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

[18] 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`.

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

[20] Nico Döttling and Sanjam Garg. 2017. Identity-based encryption from the Diffie-Hellman assumption. In *CRYPTO 2017, Part I*.

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

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

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

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

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

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

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

[28] Eike Kiltz. 2006. Chosen-ciphertext security from tag-based encryption. In *TCC 2006*.

[29] 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*.

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

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

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

[33] Yehuda Lindell. 2003. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 3.

[34] 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*.

[35] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. 2004. Alternatives to non-malleability: definitions, constructions, and applications (extended abstract). In *TCC 2004*.

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

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

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

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

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

[41] 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*.

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

[43] Monero. 2014. `https://getmonero.org/`.

[44] 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.

[45] 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`.

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

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

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

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

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

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

[52] 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*.

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

[54] 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.

[55] Victor Shoup. 2001. OAEP reconsidered. In *CRYPTO 2001*.

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

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

[58] 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*.

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

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

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

[62] 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.

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

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

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