On Sustainable Ring-Based Anonymous Systems

Sherman Chow, Christoph Egger, Russell Lai, Viktoria Ronge, Ivy Woo

On Sustainable Ring-based Anonymous Systems

Sherman S. M. Chow[1], Christoph Egger[2], Russell W. F. Lai[3], Viktoria Ronge[4], Ivy K. Y. Woo[3]

[1]Chinese University of Hong Kong
[2]Université Paris Cité, CNRS, IRIF
[3]Aalto University
[4]Friedrich-Alexander University Erlangen-Nuremberg

May 23, 2023

Anonymous systems (e.g. anonymous cryptocurrencies and updatable anonymous credentials) often follow a construction template where an account can only perform a single anonymous action, which in turn potentially spawns new (and still single-use) accounts (e.g. UTXO with a balance to spend or session with a score to claim). Due to the anonymous nature of the action, no party can be sure which account has taken part in an action and, therefore, must maintain an ever-growing list of potentially unused accounts to ensure that the system keeps running correctly. Consequently, anonymous systems constructed based on this common template are seemingly not sustainable.

In this work, we study the sustainability of ring-based anonymous systems, where a user performing an anonymous action is hidden within a set of decoy users, traditionally called a "ring".

On the positive side, we propose a general technique for ring-based anonymous systems to achieve sustainability. Along the way, we define a general model of decentralised anonymous systems (DAS) for arbitrary anonymous actions, and provide a generic construction which provably achieves sustainability. As a special case, we obtain the first construction of anonymous cryptocurrencies achieving sustainability without compromising availability. We also demonstrate the generality of our model by constructing sustainable decentralised anonymous social networks.

On the negative side, we show empirically that Monero, one of the most popular anonymous cryptocurrencies, is unlikely to be sustainable without altering its current ring sampling strategy. The main subroutine is a sub-quadratic-time algorithm for detecting used accounts in a ring-based anonymous system.

1 Introduction

Consider the setting of Hilbert's Hotel thought experiment of running infinitely many rooms, but with a privacy twist: each guest can stay in their room for as long as they want and can leave quietly without notifying Hilbert. As Hilbert respects the privacy of each guest, he never checks whether a previously assigned room has become vacant. The number of potentially occupied rooms thus grows indefinitely as more guests arrive.

The above situation manifests itself as a sustainability issue in real-world anonymous systems. In anonymous cryptocurrencies based on unspent transaction output (UTXO), the number of potentially unused accounts grows indefinitely since transactions are anonymous and confidential. Another example is anonymous credential systems, where the authority assigns to each anonymously authenticated session a score that can be anonymously claimed by the user owning the session. Similarly, the list of scored sessions grows indefinitely as the authority cannot be sure which session has been claimed.

1.1 Sustainability by Partitioning

We focus on the sustainability of ring-based anonymous systems, including popular ring-based anonymous cryptocurrencies such as Monero. Abstractly, users can perform certain actions while remaining anonymous among a set of decoy users, known as a "ring" in ring signatures, chosen ad hoc for each action. A ring-based anonymous action of a guest in Hilbert's hotel is to anonymously notify Hilbert of a set of room numbers upon leaving, one of which is the room that the guest occupied.

To ease the burden of bookkeeping the ever-growing list of potentially occupied rooms, we suggest the following intuitively simple solution. Let the hotel rooms be partitioned into chunks of equal size k. A guest must anonymously inform Hilbert of (a subset of) the chunk their room belongs to upon leaving their room. On one hand, the guest remains anonymous among other guests occupying rooms in the same chunk. On the other hand, once Hilbert is notified $k = |C|$ times about a chunk C, he can be sure that all rooms in chunk C are vacant and thus remove them from his list. A counting argument shows that the number of potentially occupied rooms is at most k times the number of guests currently staying in the hotel.

1.2 Our Contributions

We study the use of the above partitioning strategy in achieving sustainability in ring-based anonymous systems.

Positive Results

We propose a general technique based on partitioning for constructing sustainable ring-based anonymous systems. Despite being intuitively simple, formalising the sustainability notion and the technique for achieving it while maintaining a high generality requires significant effort.

Modelling To discuss the sustainability of anonymous systems in general, we must first rigorously define a general model capturing a wide class of anonymous systems of interest. Our starting point is a formal model [Lai+19] of ring confidential transactions (RingCT) [NM16], which is a ring-based anonymous system allowing users to receive coins in accounts, and transfer these coins to other accounts while preserving both spender and receiver anonymity. Generalising RingCT, Section 3 defines a general model of decentralised anonymous systems (DAS) parametrised by a class of admissible anonymous actions, which capture anonymous transactions as a special case. Note that our techniques are equally applicable to centralised systems.

Different from RingCT [Lai+19], DAS is additionally equipped with a formal definition of sustainability, which roughly requires that the description size of the state of the DAS cannot be much larger than the number of currently unused accounts.

Construction We present in Section 4.1 a construction of DAS for any polynomial-time computable anonymous actions, using the same cryptographic building blocks as those used in the RingCT construction Omniring [Lai+19], which can be efficiently instantiated over prime-order cyclic groups (e.g. elliptic curves). Restricting that rings must be output by a "partitioning sampler" [Ron+21], we can prove that our DAS construction achieves sustainability, formalising the intuitive idea presented in Section 1.1.

Applications Specialising the class of anonymous actions to anonymous transactions, we obtain in Section 5.1 essentially a sustainable version of Omniring – the first construction of anonymous cryptocurrencies that is sustainable without compromising availability. The only prior attempt [Fau+19] achieves sustainability at a detrimental cost of being vulnerable to denial-of-service attacks (see Section 1.4).

To illustrate generality and applicability, Section 5.2 shows a construction of sustainable decentralised anonymous social networks by designing complex classes of anonymous actions.

Negative Results

Our positive results show that partitioning is sufficient for sustainability. We also investigate whether it is necessary. Since this question is too open-ended to answer analytically, we narrow down and focus on

the setting of Monero, one of the popular anonymous cryptocurrencies, and empirically study its ring sampling strategy, which is modelled abstractly as the "mimicking sampler" [Ron+21].

Our contributions here are twofold. In Section 6, we propose an efficient algorithm for detecting surely-used accounts, which applies to general ring samplers (in contrast to the trivial one outlined in Section 1.1, which works only for ring sampling with partitioning samplers). It is based on the graph-theoretic framework [Egg+22] for studying ring samplers, and runs in sub-quadratic-time in the description size of the ring-membership relations. We prove that it is correct and optimal.

Moreover, we evaluate the sustainability of systems employing mimicking samplers [Ron+21]. We simulate the sampling with a mimicking sampler under various parameters, and run our proposed algorithm to detect the number of used accounts. Our empirical results in Section 7 suggest that Monero (with its current ring sampling strategy) is unlikely to be sustainable.

1.3 On Partitioning and Anonymity

Since our DAS construction requires partitioning samplers, a natural question is whether this restriction impacts anonymity, especially when compared to the ring sampling strategy used in Monero. Fortunately, like mimicking samplers, partitioning samplers have been shown [Ron+21] to achieve near-optimal anonymity, i.e. close to the inverse of the chunk size, assuming that accounts belonging to the same chunk have similar signing probabilities. This assumption was justified [Ron+21] by the observation that, in Bitcoin, the signing probability appears to depend mostly on the age (difference between its spawn time and used time) of the account, and by the fact that accounts of similar age can be identified naturally, e.g. all accounts spawned in the same block of a blockchain have the same age. Furthermore, depending on the activity of the DAS, e.g. transaction rate of a cryptocurrency, it is reasonable to have dozens or hundreds of accounts spawned in each block.

1.4 Related Work

Sustainable Anonymous Cryptocurrencies

Sustainability of anonymous cryptocurrencies was studied by Fauzi et al. [Fau+19]. In an anonymous transaction in their ring-based construction, Quisquis, the source and target accounts are replaced by their updated versions, while each decoy in the ring is replaced by a re-randomised representation of itself. As such, all users can safely forget the old accounts present in the ring but remember their updated versions instead. The number of accounts that users need to remember hence remains unchanged after each transaction, meaning that the system is sustainable. This mechanism, however, creates an issue of *availability* [Bün+20], where a malicious user can deny an honest user from using its accounts by repeatedly including them as decoys, thereby re-randomising them, in transactions. As the adversary is first to learn the re-randomised representations of the victim's accounts, it is well-positioned to deny access of the victim indefinitely.

Our construction avoids this problem by keeping decoy accounts unchanged and achieving state clean-up differently: We partition the universe of accounts into chunks and require any set of decoys to stay within a single chunk. A full chunk can be removed once the number of actions (transactions) involving the chunk equals the chunk size, as it can be publicly deduced that all accounts in that chunk have been expended.

Detection of Surely-Used Accounts

To our knowledge, detecting accounts that have surely been used in a ring-based anonymous system has been a largely untouched question. A closely related topic is traceability to the source accounts of rings. Since a traced account is a used account (although the converse is not true), methodologies for tracing purposes can be seen as potential tools for detecting used accounts.

The notion of "closed set" was informally introduced [Yu+19] as a stepping stone for tracing transactions, with an approximated search algorithm for closed sets to replace brute-force searching. Traceability of Monero since the deployment of RingCT was also empirically investigated [Vij21], where the DM decomposition was applied to the graph representation of Monero's ring-membership history, with the result of no account being traced.

Section 6 formalises the prior informal notion [Yu+19] as transaction graphs with two properties: closed and ring-induced. We then give a sub-quadratic-time exact algorithm to search for the largest closed ring-induced subgraphs in a transaction graph. We formally prove that this is equivalent to searching all accounts that are surely used in a DAS. Using our result, we can further deduce from the existing empirical result [Vij21] that there has been no detectable used account in Monero since RingCT was deployed (except those deliberately created [Wij+18]), coinciding with our experimental result in Section 7.

2 Basic Cryptographic Primitives

$\underline{\mathsf{OW}_{\mathsf{TAG},\mathcal{A}}(1^\lambda)}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{msk} \leftarrow \mathsf{SKGen}(\mathsf{pp})$
$\mathsf{mpk} \leftarrow \mathsf{PKGen}(\mathsf{msk})$
$\mathsf{sk}^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{mpk})$
return $(\mathsf{TagEval}(\mathsf{sk}^*) \in \mathsf{Tag})$

$\underline{\mathsf{Binding}_{\mathsf{TAG},\mathcal{A}}(1^\lambda)}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{sk}_0, \mathsf{sk}_1 \leftarrow \mathcal{A}(\mathsf{pp})$
$b_0 := (\mathsf{sk}_0 \neq \mathsf{sk}_1)$
$b_1 := (\mathsf{TagEval}(\mathsf{sk}_0) = \mathsf{TagEval}(\mathsf{sk}_1))$
return $b_0 \wedge b_1$

$\underline{\mathsf{IND\text{-}RKA}^b_{\mathsf{TAG},\mathcal{A}}(1^\lambda)}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda);\ \mathsf{msk} \leftarrow \mathsf{SKGen}(\mathsf{pp})$
$\delta^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{mpk} \leftarrow \mathsf{PKGen}(\mathsf{msk}))$
if $b = 0$ **then** $\mathsf{tag}^* \leftarrow \mathsf{TagEval}(\mathsf{SKDer}(\mathsf{msk}, \delta^*))$
else $\mathsf{tag}^* \leftarrow \mathsf{TagEval}(\mathsf{SKGen}(\mathsf{pp}))$
return $\mathcal{A}^{\mathcal{O}}(\mathsf{tag}^*) \wedge (\delta^* \notin \Delta)$

$\underline{\mathcal{O}(\delta)}$
$\mathsf{sk} \leftarrow \mathsf{SKDer}(\mathsf{msk}, \delta)$
$\mathsf{tag} \leftarrow \mathsf{TagEval}(\mathsf{sk})$
$\Delta := \Delta \cup \{\delta\}$
$\mathsf{Tag} := \mathsf{Tag} \cup \{\mathsf{tag}\}$
return tag

Figure 1: One-wayness, key-binding, and related-key pseudorandomness experiments for tagging schemes. $\mathcal{O}(\delta)$ is an oracle for generating malicious tags (maintained in a set Δ internally) using a secret key derived from the master secret key based on adversarially-chosen auxiliary information δ.

Let $\lambda \in \mathbb{N} = \{1, 2, \ldots\}$ be the security parameter. We write PPT for probabilistic polynomial time. A function $\mu(\lambda)$ is negligible in λ if $\mu(\lambda) = o(1/p(\lambda))$ for any polynomial p. $\mathbb{Z}_q = [q] = \{0, 1, \ldots, q-1\}$ for any $q \in \mathbb{N}$, and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For sets I and X, we write $\langle x_i \rangle_{i \in I}$ for the set $\{(i, x_i)\}_{i \in I}$, and $\langle x_i \rangle_{i \in T} \subseteq \langle y_i \rangle_{i \in U}$ if $T \subseteq U$ and $x_i = y_i\ \forall i \in T$.

Appendix A recalls basic cryptographic primitives. Below is a more general syntax and security properties of tagging schemes [Lai+19]. A tagging scheme provides two functionalities: 1. From a master public key, derive an ephemeral public key along with some auxiliary information. Given the master secret key and the auxiliary information, recover the corresponding ephemeral secret key. 2. From an ephemeral secret key, derive a one-way, binding, and pseudorandom tag. These security properties should hold even if the adversary is given an oracle for generating tags for keys derived from the same master key.

Definition 2.1 (Tagging Scheme)**.** *A tagging scheme* TAG *consists of* (Setup, SKGen, PKGen, PKDer, SKDer, TagEval) *for setup, secret-key generation, pubic-key generation, public-key derivation, secret-key derivation, and tag evaluation.*

$\underline{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)}$ *generates the public parameters* pp *to be input by all other algorithms (often implicitly).*

$\underline{\mathsf{msk} \leftarrow \mathsf{SKGen}(\mathsf{pp})}$ *generates a master secret key* msk.

$\underline{\mathsf{mpk}/\mathsf{pk} \leftarrow \mathsf{PKGen}(\mathsf{msk}/\mathsf{sk})}$ *deterministically generates the master public key* mpk *(or public key* pk*) corresponding to the master secret key* msk *(or secret key* sk*).*

$\underline{(\mathsf{pk}, \delta) \leftarrow \mathsf{PKDer}(\mathsf{mpk})}$ *derives a public key* pk *with some auxiliary information* δ *from a master public key* mpk.

Table 1: DAS Syntax, Part I

Algorithm	Syntax
Setup	$(\mathsf{pp}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda)$
Master Key Generation	$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MKGen}(\mathsf{pp})$
Account Key Derivation	$(\mathsf{ask}, x) \leftarrow \mathsf{AKDer}(\mathsf{msk}, \mathsf{tk})$
Source Account Checking	$b \leftarrow \mathsf{SChk}(\mathsf{acc}, \mathsf{ask}, x)$
Target Account Checking	$b \leftarrow \mathsf{TChk}(\mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y)$

$\underline{\mathsf{sk} \leftarrow \mathsf{SKDer}(\mathsf{msk}, \delta)}$ *deterministically derives the secret key* sk *corresponding to the public key* pk.

$\underline{\mathsf{tag} \leftarrow \mathsf{TagEval}(\mathsf{sk})}$ *deterministically generates a tag corresponding to the secret key* sk *(or* msk*).*

Definition 2.2 (Correctness)**.** TAG *is correct if, for all* $\lambda \in \mathbb{N}$, $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, $\mathsf{msk} \in \mathsf{SKGen}(\mathsf{pp})$, $\mathsf{mpk} \in \mathsf{PKGen}(\mathsf{msk})$, *and* $(\mathsf{pk}, \delta) \in \mathsf{PKDer}(\mathsf{mpk})$, $\mathsf{pk} = \mathsf{PKGen}(\mathsf{SKDer}(\mathsf{msk}, \delta))$ *holds.*

Definition 2.3 (One-Way)**.** TAG *is one-way if, for any PPT adversary* $\mathcal{A}$*, it holds that*

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

is negligible in λ*, where* $\mathsf{OW}_{\mathsf{TAG},\mathcal{A}}$ *is defined in Figure 1.*

Definition 2.4 (Key-Binding)**.** TAG *is key-binding if, for any PPT adversary* $\mathcal{A}$*, it holds that*

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

is negligible in λ*, where* $\mathsf{Binding}_{\mathsf{TAG},\mathcal{A}}$ *is defined in Figure 1.*

Definition 2.5 (Related-Key Pseudorandomness)**.** *A tagging scheme* TAG *is related-key pseudorandom if*

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

is negligible in λ *for any PPT adversary* $\mathcal{A}$*, where* $\mathsf{IND\text{-}RKA}^b_{\mathsf{TAG},\mathcal{A}}$ *is defined in Figure 1.*

3 Decentralised Anonymous Systems

We formalise a general notion of decentralised anonymous systems (DAS), which captures a wide class of systems parametrised by their supported anonymous actions. For a more intuitive understanding, in a series of footnotes, we show how DAS captures RingCT [Lai+19] and anonymous credentials (AC) as special cases and highlight additional differences.

3.1 Overview

Account Account acc is the basic unit of a DAS, which is associated with an account secret key ask and a (hidden) attribute x belonging to an attribute space $\mathcal{X}$.[1] A "ring" is a set of accounts. A current list of potentially unused accounts is maintained by all users, along with the state st[2] of the system.

Keys Each DAS user is identified by a master public key mpk who knows the corresponding master secret key msk. A user owns an account if they know the corresponding account secret key ask. Each user may own arbitrarily many accounts.

[1] In RingCT, an attribute x would be an integer representing the amount of coins stored in the account acc. In AC, it could be a reputation score.

[2] In RingCT, the state includes tags corresponding to spent accounts. The existing model [Lai+19] implicitly assumes that all users have access to the (growing) list of spent tags. In AC, it could contain a list of scores of unclaimed sessions.

Table 2: DAS Syntax, Part II

Algorithm	Syntax
Action	$\left(\mathsf{tx}, \langle \overline{\mathsf{acc}}_i, \mathsf{tk}_i \rangle_{i\in[n]}\right) \leftarrow \mathsf{Act}\left(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid}\in U}, p_{m,n}, \langle R_i, \mathsf{aid}_i, \mathsf{ask}_i, x_i \rangle_{i\in[m]}, \langle \mathsf{mpk}_i, y_i \rangle_{i\in[n]}, \mathsf{aux}\right)$
Verification	$\left(b, \mathsf{st}', \langle \mathsf{acc}'_{\mathsf{aid}} \rangle_{\mathsf{aid}\in U'}\right) \leftarrow \mathsf{Vf}(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid}\in U}, p_{m,n}, q_m, \mathsf{tx}, \langle \overline{\mathsf{acc}}_i \rangle_{i\in[n]})$

Action An anonymous action is modelled as a mapping from the attributes $(x_i)_{i\in[m]}$ of some source accounts to the attributes $(y_j)_{j\in[n]}$ of some target accounts the action creates.[3] Each account can be used in only a single anonymous action. More concretely, a user who knows:

- the account secret keys $(\mathsf{ask}_i)_{i\in[m]}$ of the source accounts $(\mathsf{acc}_i)_{i\in[m]}$ (identified by the identifiers $(\mathsf{aid}_i)_{i\in[m]}$ in our formal definition) with attributes $(x_i)_{i\in[m]}$, and
- the master public keys $(\mathsf{mpk}_j)_{j\in[n]}$ and attributes $(y_j)_{j\in[n]}$

could perform an anonymous action which results in

- an action transcript tx to be verified by other users,
- the creation of the target accounts $(\overline{\mathsf{acc}}_j)_{j\in[n]}$ with attributes $(y_j)_{j\in[n]}$, and
- a token $\mathsf{tk}_{j\in[n]}$ for each target account $\overline{\mathsf{acc}}_j$, which can be combined with the master secret key msk_j to recover the account secret key for $\overline{\mathsf{acc}}_j$.

Predicates A DAS is parametrised by a family of attribute predicates $\mathcal{P}$ and a family of ring predicates $\mathcal{Q}$. These predicates define "legal" relations of input and output attributes and choices of rings, respectively.

The class of admissible anonymous actions[4] is captured by $\mathcal{P}$. Formally, $\mathcal{P}$ is a family of polynomial-time computable predicates where each predicate $p_{m,n} \in \mathcal{P}$ is parametrised by $(m, n) \in \mathbb{N}^2$ and takes the following inputs:

- input attributes $x_i \in \mathcal{X}$ for $i \in [m]$,
- output attributes $y_j \in \mathcal{X}$ for $j \in [n]$, and
- auxiliary input aux justifying the action (e.g. the witness of an NP statement) required in more complex policies.

Each source account acc_i used in an anonymous action is hidden within a ring R_i chosen by the action-performing user, subject to some public constraints $\mathcal{Q}$. Formally, $\mathcal{Q}$ is a family of polynomial-time computable predicates where each predicate $q_m \in \mathcal{Q}$ is parametrised by $m \in \mathbb{N}$ and takes as input:

- a universe $U \subseteq \mathbb{N}_0$, and
- a sequence of rings $(R_i)_{i\in[m]} \subseteq (\mathbb{N}_0)^m$.

Looking ahead, without the enforcement of predicate q_m in our sustainable DAS construction, one can have complete flexibility in claiming the ring of each source account; our partitioning strategy (Section 1.1) would run into availability issues since the adversary could elicit premature account deletion.

Verification and Garbage Collection Upon receiving predicates $p_{m,n}$ and q_m, an action transcript tx, and a corresponding list of target accounts $(\overline{\mathsf{acc}}_i)_{i\in[n]}$, a user could verify them against the current state st of the system and the current list of potentially unused accounts. Should the action be considered valid, the user suitably updates its view of the state of the system to st' and appends the list of target accounts $(\overline{\mathsf{acc}}_i)_{i\in[n]}$ to the list of potentially unused accounts. Finally, a "garbage collection" step may be performed to detect accounts which surely have been used.

[3] In RingCT, actions are transactions. A user spends by creating a transaction that transfers amounts, i.e. creating target receiving accounts which store the transferred amounts as attributes. In AC, an action would be to authenticate.

[4] In RingCT, admissible anonymous actions are given by transactions with balanced, non-negative, and bounded (to prevent overflow) input and output amounts and tags honestly derived from the account secret key (so double-spending can be detected upon duplication of tags). In AC, we may want to forbid users with negative reputation to authenticate, or claim a reputation score beyond the claimable value.

3.2 Syntax

Definition 3.1 (DAS)**.** *A decentralised anonymous system (DAS) for* $(\mathcal{P}, \mathcal{Q})$ *consists of the following seven PPT algorithms with the syntax defined in Tables 1 and 2.*

Setup. Setup *generates the public parameters* pp *and an initial state* st *of the system.*

Master Key Generation. MKGen *generates a pair of master public and secret keys* $(\mathsf{mpk}, \mathsf{msk})$*.*

Account Key Derivation. AKDer *inputs* msk *and a token* tk*. It derives an account secret key* ask *and an attribute* x*.*

Action. Act *inputs a state* st*, a list of potentially unused accounts* $\langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}$*, a predicate* $p_{m,n} \in \mathcal{P}$*, a sequence of* m *tuples* $(R_i, \mathsf{aid}_i, \mathsf{ask}_i, x_i)$ *consisting of a ring* $R_i \subseteq U$*, a source account identifier* $\mathsf{aid}_i \in R_i$*, an account secret key* ask_i*, and an attribute* $x_i \in \mathcal{X}$*, a sequence of* n *tuples* (mpk_i, y_i) *consisting of a master public key* mpk_i *and an attribute* $y_i \in \mathcal{X}$*, and some auxiliary information* aux*. It outputs an action transcript* tx *and a sequence of* n *tuples* $(\overline{\mathsf{acc}}_i, \mathsf{tk}_i)$ *consisting of a target account and its token.*

Verification. Vf *inputs a state* st*, a list of potentially unused accounts* $\langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}$*, predicates* $p_{m,n}$ *and* q_m*, an action transaction* tx*, and a sequence of* n *target accounts* $\langle \overline{\mathsf{acc}}_i \rangle_{i \in [n]}$*. It outputs a bit* b *deciding whether to accept the action and, if so, an updated state* st' *and an updated list* $\langle \mathsf{acc}'_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U'}$ *of potentially unused accounts.*

Source Account Checking. SChk *is an auxiliary algorithm deciding if a tuple* $(\mathsf{acc}, \mathsf{ask}, x)$ *is valid, i.e. the account secret key and attribute of account* acc *is* ask *and* $x \in \mathcal{X}$*, respectively.*

Target Account Checking. TChk *is an auxiliary algorithm checking the validity of a tuple* $(\mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y)$*, i.e. account* acc *with token* tk *is supposed to be created for master public key* mpk *and attribute* $y \in \mathcal{X}$*.*

The syntax of DAS is very similar to that of RingCT [Lai+19]. The setup and master key generation MKGen algorithms share identical syntax. The DAS account key derivation AKDer algorithm generalises the RingCT receive algorithm. The DAS action algorithm Act combines and generalises the RingCT one-time account generation and spend algorithms. The verification algorithms are identical up to statefulness and definition style. The DAS source and target account checking algorithms SChk and TChk generalise the amount and tag checking algorithms.

We highlight the following differences in the definitions: 1. Our DAS model is explicitly stateful (recall Footnote 2). 2. A DAS is parametrised by a ring predicate family $\mathcal{Q}$ which is crucial for achieving sustainability. RingCT [Lai+19] does not impose any restrictions on rings. 3. A DAS captures arbitrary actions beyond transactions. 4. The RingCT model [Lai+19] explicitly involves linkable tags for enforcing the single-use restriction. Other mechanisms are possible (e.g. [Fau+19], see Section 1.4), which our DAS model captures. Our DAS model enforces the single-use restriction through the authenticity property to be defined shortly, which generalises the RingCT balance property [Lai+19], but does not explicitly specify mechanisms for achieving it. 5) RingCT does not have a garbage collection step. Their (implicit) system state can only grow indefinitely.

3.3 Properties

We define the integrity, authenticity, privacy, and availability properties of DAS, generalising the correctness, balance, privacy, and non-slanderability of RingCT [Lai+19], respectively, as well as a new sustainability property. To reduce clutter, uninitialised sets which are referred to are implicitly initialised as empty sets.

Integrity

A DAS is said to have integrity if it satisfies both derivation integrity and action integrity.

Derivation integrity. If a tuple $(\mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y)$ is considered valid by TChk, then (ask, y) can be recovered given $(\mathsf{msk}, \mathsf{tk})$ such that $(\mathsf{acc}, \mathsf{ask}, y)$ is considered valid by SChk.

Action integrity. If actions are performed over valid inputs, which in particular include $(\mathsf{acc}, \mathsf{ask}, x)$ tuples considered valid by SChk, and each account is only used once, then Vf always accepts and resulting target accounts $\langle \overline{\mathsf{acc}}_{i,j} \rangle$ are always considered valid by TChk.

DAS integrity strengthens RingCT correctness in that correctness is required to hold even if some or all inputs, in particular the target master public keys, for generating actions are adversarially chosen. Since integrity subsumes correctness, we do not formally define the latter.

Definition 3.2 (Integrity)**.** *A decentralised anonymous system Ω for $(\mathcal{P}, \mathcal{Q})$ has integrity if for any PPT algorithm $\mathcal{A}$, $\Pr\left[\mathsf{DerivationIntegrity}_{\Omega,\mathcal{A}}(1^\lambda) = 1\right]$ and $\Pr\left[\mathsf{ActionIntegrity}_{\Omega,\mathcal{A}}(1^\lambda) = 1\right]$ are both negligible in λ. Figure 2 define the two experiments.*

Authenticity

A DAS is said to have authenticity if a user can authenticate that its action satisfies the system predicates. This is captured by the properties of source binding, target binding and knowledge-sound, explained in the following.

Definition 3.3 (Authenticity)**.** *A decentralised anonymous system Ω for $(\mathcal{P}, \mathcal{Q})$ has authenticity if it satisfies source binding, target binding, and knowledge soundness as below.*

Source Binding *Source binding states that an account* acc *is computationally bound by* SChk *to a tuple* (ask, x) *of account secret key and attribute. The probability*

$$\Pr\left[\begin{array}{l} \mathsf{SChk}(\mathsf{acc}, \mathsf{ask}, x) = 1 \\ \wedge\ \mathsf{SChk}(\mathsf{acc}, \mathsf{ask}', x') = 1 \\ \wedge\ (\mathsf{ask}, x) \neq (\mathsf{ask}', x') \end{array} \middle| \begin{array}{l} (\mathsf{pp}, \mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{acc}, \mathsf{ask}, x, \mathsf{ask}', x') \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}) \end{array} \right]$$

should be negligible in λ for any PPT adversary $\mathcal{A}$.

Target Binding *Target binding states that an account* acc *is computationally bound by* TChk *to an attribute y. The probability below should be negligible in λ for any PPT $\mathcal{A}$:*

$$\Pr\left[\begin{array}{l} \mathsf{TChk}(\mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y) = 1 \\ \wedge\ \mathsf{TChk}\left(\begin{array}{c} \mathsf{acc}, \mathsf{mpk}', \\ \mathsf{tk}', y' \end{array}\right) = 1 \\ \wedge\ y \neq y' \end{array} \middle| \begin{array}{l} (\mathsf{pp}, \mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \left(\begin{array}{c} \mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y, \\ \mathsf{mpk}', \mathsf{tk}', y' \end{array}\right) \leftarrow \mathcal{A}\left(\begin{array}{c} \mathsf{pp}, \mathsf{st}, \\ \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U} \end{array}\right) \end{array} \right].$$

Knowledge Soundness *Knowledge soundness states that a (collection of) user(s) producing valid action transcripts must "know" the valid inputs to the* Act *algorithm which result in said transcripts, and that each source account used to produce said transcripts is only used once. Formally, for any PPT adversary $\mathcal{A}$, there exists an expected polynomial-time extractor $\mathcal{E}_\mathcal{A}$ such that $\Pr\left[\mathsf{Authenticity}_{\Omega,\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda) = 1\right]$ is negligible in λ, where $\mathsf{Authenticity}_{\Omega,\mathcal{A},\mathcal{E}_\mathcal{A}}$ is defined in Figure 3.*

RingCT provides a mechanism (i.e. linkable tags) for enforcing the single-use restriction, but such a restriction is not captured in any security property, in particular, the balance property. The DAS authenticity property explicitly captures the single-use restriction.

Privacy

A DAS is private if, given an action transcript tx, each source account acc_i involved in the action is computationally hidden within the ring R_i (i.e. source-private) and each target account computationally hides its associated master public key and attribute (i.e. target-private).

Definition 3.4 (Privacy)**.** *A decentralised anonymous system Ω for $(\mathcal{P}, \mathcal{Q})$ is private if for any PPT adversary $\mathcal{A}$,*

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

is negligible in λ, where $\mathsf{Privacy}^b_{\Omega,\mathcal{A}}$ is defined in Figure 5.

$\mathsf{DerivationIntegrity}_{\Omega,\mathcal{A}}(1^\lambda)$

$(\mathsf{pp}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MKGen}(\mathsf{pp})$

$(\mathsf{acc}, \mathsf{tk}, y) \leftarrow \mathcal{A}\,(\mathsf{pp}, \mathsf{st}, \mathsf{mpk}, \mathsf{msk})$

$(\mathsf{ask}, x) \leftarrow \mathsf{AKDer}(\mathsf{msk}, \mathsf{tk})$

$\mathsf{valid\_target} := (\mathsf{TChk}(\mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y) = 1)$

$\mathsf{attribute\_recovered} := (x = y)$

$\mathsf{valid\_source} := (\mathsf{SChk}(\mathsf{acc}, \mathsf{ask}, x) = 1)$

return $\mathsf{valid\_target} \wedge \neg(\mathsf{attribute\_recovered} \wedge \mathsf{valid\_source})$

$\mathsf{ActionIntegrity}_{\Omega,\mathcal{A}}(1^\lambda)$

$(\mathsf{pp}, \mathsf{st}_0) \leftarrow \mathsf{Setup}(1^\lambda)$

$$\left(\begin{array}{c} p_{i,m_i,n_i}, q_{i,m_i}, \\ \langle R_{i,j}, \mathsf{aid}_{i,j}, \mathsf{ask}_{i,j}, x_{i,j} \rangle_{j\in[m_i]}, \\ \langle \mathsf{mpk}_{i,j}, y_{i,j} \rangle_{j\in[n_i]}, \mathsf{aux}_i \end{array} \right)_{i\in[t]} \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{st}_0)$$

for $i \in [t]$ **do**

$$\left(\mathsf{tx}_i, \langle \overline{\mathsf{acc}}_{i,j}, \mathsf{tk}_{i,j} \rangle_{j\in[n_i]}\right) \leftarrow \mathsf{Act}\left(\begin{array}{c} \mathsf{st}_i, \langle \mathsf{acc}_{i,j} \rangle_{j\in U_i}, p_{i,m_i,n_i}, \\ \langle R_{i,j}, \mathsf{aid}_{i,j}, \mathsf{ask}_{i,j}, x_{i,j} \rangle_{j\in[m_i]}, \\ \langle \mathsf{mpk}_{i,j}, y_{i,j} \rangle_{j\in[n_i]}, \mathsf{aux}_i \end{array} \right)$$

$$\left(b_i, \mathsf{st}_{i+1}, \langle \mathsf{acc}_{i+1,j} \rangle_{j\in U_{i+1}}\right) \leftarrow \mathsf{Vf}\left(\begin{array}{c} \mathsf{st}_i, \langle \mathsf{acc}_{i,j} \rangle_{j\in U_i}, p_{i,m_i,n_i}, \\ q_{i,m_i}, \mathsf{tx}_i, \langle \overline{\mathsf{acc}}_{i,j} \rangle_{j\in[n_i]} \end{array} \right)$$

$\mathsf{valid\_predicates} := (\forall i \in [t],\ p_{i,m_i,n_i} \in \mathcal{P} \wedge q_{i,m_i} \in \mathcal{Q})$

$\mathsf{valid\_rings} := (\forall i \in [t],\ \forall j \in [m_i],\ \mathsf{aid}_{i,j} \in R_{i,j} \subseteq U_i)$

$\mathsf{valid\_sources} := \left(\forall i \in [t],\ \forall j \in [m_i],\ \mathsf{SChk}(\mathsf{acc}_{\mathsf{aid}_{i,j}}, \mathsf{ask}_{i,j}, x_{i,j}) = 1\right)$

$\mathsf{valid\_attributes} := (\forall i \in [t],\ \forall j \in [m_i],\ \forall k \in [n_i],\ x_{i,j}, y_{i,k} \in \mathcal{X})$

$$\mathsf{satisfied} := \left(\forall i \in [t], \quad \begin{array}{l} p_{i,m_i,n_i}((x_{i,j})_{j\in[m_i]}, (y_{i,j})_{j\in[n_i]}, \mathsf{aux}_i) = 1 \\ \wedge\, q_{i,m_i}(U_i, (R_{i,j})_{j\in[m_i]}) = 1 \end{array} \right)$$

$$\mathsf{single\_use} := \left(\forall \text{ distinct } i, i' \in [t],\ \{\mathsf{aid}_{i,j}\}_{j\in[m_i]} \cap \{\mathsf{aid}_{i',j}\}_{j\in[m_{i'}]} = \emptyset\right)$$

$\mathsf{valid\_txs} := (\forall i \in [t],\ b_i = 1)$

$\mathsf{valid\_targets} := \left(\forall i \in [t],\ \forall j \in [m_i],\ \mathsf{TChk}(\overline{\mathsf{acc}}_{i,j}, \mathsf{mpk}_{i,j}, \mathsf{tk}_{i,j}, y_i) = 1\right)$

$$\mathsf{targets\_pushed} := \left(\forall i \in [t],\ \forall j \in [n_i],\ \overline{\mathsf{acc}}_{i,j} \in \{\mathsf{acc}_{i+1,j'}\}_{j'\in U_{i+1}}\right)$$

return $\mathsf{valid\_predicates} \wedge \mathsf{valid\_rings} \wedge \mathsf{valid\_sources} \wedge \mathsf{valid\_attributes}$
$\wedge\, \mathsf{satisfied} \wedge \mathsf{single\_use} \wedge \neg(\mathsf{valid\_txs} \wedge \mathsf{valid\_targets} \wedge \mathsf{targets\_pushed})$

Figure 2: Derivation and Action integrity experiments.

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

$(\mathsf{pp}, \mathsf{st}_0) \leftarrow \mathsf{Setup}(1^\lambda)$

$$\left(\begin{array}{l} \left(p_{i,m_i,n_i}, q_{i,m_i}, \mathsf{tx}_i, \langle \overline{\mathsf{acc}}_{i,j} \rangle_{j\in[n_i]}\right)_{i\in[t]}, \\ \left(\begin{array}{l} \langle R_{i,j}, \mathsf{aid}_{i,j}, \mathsf{ask}_{i,j}, x_{i,j} \rangle_{j\in[m_i]}, \\ \langle \mathsf{mpk}_{i,j}, \mathsf{tk}_{i,j}, y_{i,j} \rangle_{j\in[n_i]}, \mathsf{aux}_i \end{array} \right)_{i\in[t]} \end{array} \right) \leftarrow (\mathcal{A}, \mathcal{E}_{\mathcal{A}})(\mathsf{pp}, \mathsf{st}_0)$$

for $i \in [t]$ **do**

$$\left(b_i, \mathsf{st}_{i+1}, \langle \mathsf{acc}_{i+1,j} \rangle_{j\in U_{i+1}}\right) \leftarrow \mathsf{Vf}\left(\mathsf{st}_i, \langle \mathsf{acc}_{i,j} \rangle_{j\in U_i}, p_{i,m_i,n_i}, q_{i,m_i}, \mathsf{tx}_i, \langle \overline{\mathsf{acc}}_{i,j} \rangle_{j\in[n_i]}\right)$$

$$b'_i := \begin{cases} \forall j \in [n_i],\ \exists j' \in U_{i+1},\ \overline{\mathsf{acc}}_{i,j} = \mathsf{acc}_{i+1,j'} \\ (p_{i,m_i,n_i} \in \mathcal{P}) \wedge (q_{i,m_i} \in \mathcal{Q}) \\ p_{i,m_i,n_i}\left((x_{i,j})_{j\in[m_i]}, (y_{i,j})_{j\in[n_i]}, \mathsf{aux}_i\right) = 1 \\ q_{i,m_i}\left(U_i, (R_{i,j})_{j\in[m_i]}\right) = 1 \\ \forall j \in [m_i],\ \mathsf{aid}_{i,j} \in R_{i,j} \subseteq U_i \\ \forall j \in [m_i],\ \mathsf{SChk}(\mathsf{acc}_{i,j}, \mathsf{ask}_{i,j}, x_{i,j}) = 1 \\ \forall j \in [n_i],\ \mathsf{TChk}(\overline{\mathsf{acc}}_{i,j}, \mathsf{mpk}_{i,j}, \mathsf{tk}_{i,j}, y_{i,j}) = 1 \end{cases}$$

$\mathsf{valid\_txs} := (\forall i \in [t],\ b_i = 1)$; $\mathsf{valid\_actions} := \left(\forall i \in [t],\ b'_i = 1\right)$

$\mathsf{single\_use} := \left(\forall \text{ distinct } i, i' \in [t],\ \{\mathsf{aid}_{i,j}\}_{j\in[m_i]} \cap \{\mathsf{aid}_{i',j}\}_{j\in[m_{i'}]} = \emptyset\right)$

return $\mathsf{valid\_txs} \wedge \neg(\mathsf{valid\_actions} \wedge \mathsf{single\_use})$

Figure 3: Authenticity experiment.

The definitions of DAS and RingCT privacy are essentially the same, modulo the differences inherited from syntactical changes. In particular, we emphasise that both definitions capture both source and target accounts anonymity: source anonymity by hiding source accounts within a ring and target anonymity by the one-time account derivation mechanism.

Availability

A DAS is said to be available if an action which is considered valid against a state st will still be considered valid against a newer state st'.

Definition 3.5 (Availability)**.** *A decentralised anonymous system Ω for $(\mathcal{P}, \mathcal{Q})$ is available if for any PPT adversary $\mathcal{A}$, $\Pr\left[\mathsf{Availability}_{\Omega,\mathcal{A}}(1^\lambda) = 1\right]$ is negligible in λ, where $\mathsf{Availability}_{\Omega,\mathcal{A}}$ is defined in Figure 5.*

Our availability property generalises the RingCT non-slanderability property [Lai+19]. In RingCT, which has the linkable tag mechanism baked into the syntax, the only way to invalidate an (intercepted) honest user's transaction is to forge a tag linking to the honest user's account. Since DAS allows any single-use mechanism, we generalise the non-slanderability to availability, capturing attacks trying to invalidate honest action transcripts by any means, e.g. the attack against Quisquis mentioned in Section 1.4. To elaborate, consider that a user generates a Quisquis transaction tx at time t, which is valid w.r.t. a state st_t. Before being able to publish the transaction, an adversary intercepts and publishes another transaction with overlapping source accounts, i.e. input ring members, thus advancing the state to st_{t+1}. Now, the transaction tx will no longer be valid w.r.t. st_{t+1}. This violates availability.

Formally, these attacks are captured in the availability experiment by running the verification algorithm on an honestly generated transcript, followed by an arbitrary sequence of adversarial actions (e.g. intercepting the honest transcript), followed by another run of the verification algorithm. The adversary wins if they manage to let the verification pass the first time but fail the second time.

Sustainability

Finally, we define the notion of sustainability for DAS. A DAS is said to be sustainable if, at any point in time, the list $\langle \mathsf{acc}_i \rangle_{i\in U}$ of potentially unused accounts and the system state st are not much larger than

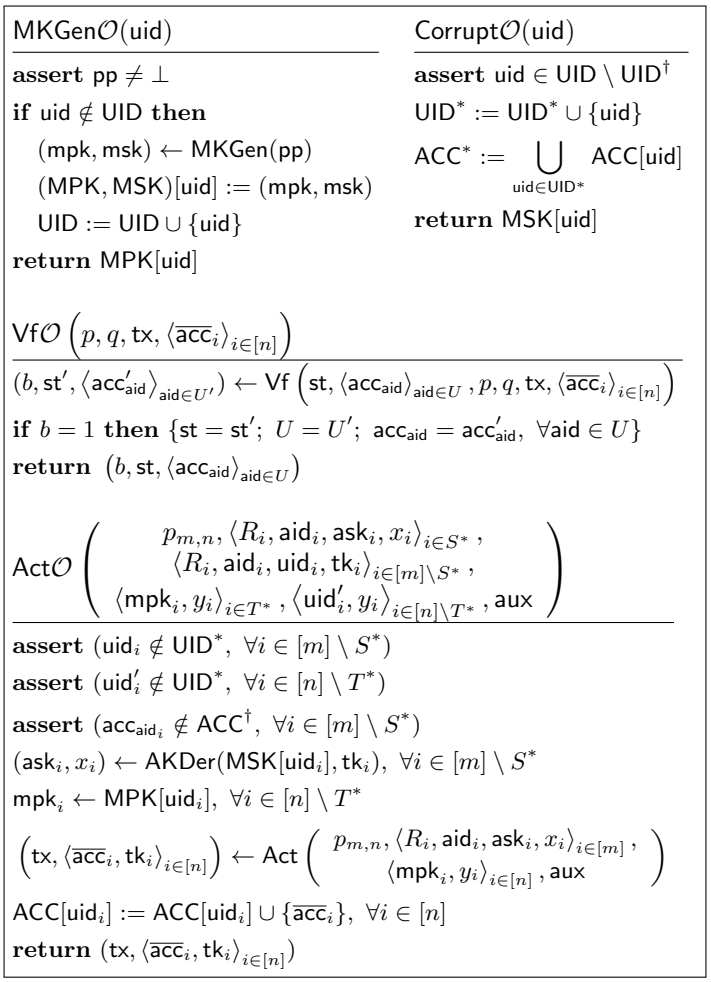

Figure 4: Oracles for Privacy and Availability experiments.

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

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

$\mathbb{O} := \{\mathsf{MKGen}\mathcal{O}, \mathsf{Corrupt}\mathcal{O}, \mathsf{Act}\mathcal{O}, \mathsf{Vf}\mathcal{O}\}$

$$\left(\begin{array}{c} p_{m,n}, q_m, \langle R_i, \mathsf{aid}_i, \mathsf{ask}_i, x_i\rangle_{i\in S^*}, \\ \left(\langle R_{i,j}, \mathsf{aid}_{i,j}, \mathsf{uid}_{i,j}, \mathsf{tk}_{i,j}\rangle_{i\in[m]\setminus S^*}\right)_{j=0}^{1}, \langle \mathsf{mpk}_i, y_i\rangle_{i\in T^*}, \\ \left(\langle \mathsf{uid}'_{i,j}, y_{i,j}\rangle_{i\in[n]\setminus T^*}\right)_{j=0}^{1}, (\mathsf{aux}_j)_{j=0}^{1} \end{array}\right) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp},\mathsf{st})$$

assert $\langle R_{i,0}\rangle_{i\in[m]\setminus S^*} = \langle R_{i,1}\rangle_{i\in[m]\setminus S^*}$

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

$$\left(\begin{array}{c} \mathsf{tx}_j, \\ \langle \overline{\mathsf{acc}}_{i,j}, \mathsf{tk}_{i,j}\rangle_{i\in[n]} \end{array}\right) \leftarrow \mathsf{Act}\mathcal{O}\left(\begin{array}{c} p_{m,n}, \langle R_i, \mathsf{ask}_i, x_i\rangle_{i\in S^*}, \\ \langle R_{i,j}, \mathsf{uid}_{i,j}, \mathsf{tk}_{i,j}\rangle_{i\in[m]\setminus S^*}, \\ \langle \mathsf{mpk}_i, y_i\rangle_{i\in T^*}, \langle \mathsf{uid}'_{i,j}, y_{i,j}\rangle_{i\in[n]\setminus T^*}, \\ \mathsf{aux}_j \end{array}\right)$$

$(b_j, *, *) \leftarrow \mathsf{Vf}(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}}\rangle_{\mathsf{aid}\in U}, p_{m,n}, q_m, \mathsf{tx}_j, \langle \overline{\mathsf{acc}}_{i,j}\rangle_{i\in[n]})$

assert $b_j = 1$

$\mathsf{UID}^\dagger := \mathsf{UID}^\dagger \cup \{\mathsf{uid}_{i,j} : i \in [m] \setminus S^*\} \cup \{\mathsf{uid}'_{i,j} : i \in [n] \setminus T^*\}$

$\mathsf{ACC}^\dagger := \mathsf{ACC}^\dagger \cup \{\mathsf{acc}_{\mathsf{aid}_i,j}\}_{i\in[m]\setminus S^*} \cup \{\overline{\mathsf{acc}}_{i,j}\}_{i\in[n]\setminus T^*}$

return $b' \leftarrow \mathcal{A}^{\mathbb{O}}\left(\mathsf{tx}_b, \langle \overline{\mathsf{acc}}_{i,b}\rangle_{i\in[n]}\right)$

$\mathsf{Availability}_{\Omega,\mathcal{A}}(1^\lambda)$

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

$\mathbb{O} := \{\mathsf{MKGen}\mathcal{O}, \mathsf{Corrupt}\mathcal{O}, \mathsf{Act}\mathcal{O}, \mathsf{Vf}\mathcal{O}\}$

$$\left(\begin{array}{c} p_{m,n}, q_m, \langle R_i, \mathsf{aid}_i, \mathsf{uid}_i, \mathsf{tk}_i\rangle_{i\in[m]}, \\ \langle \mathsf{mpk}_i, y_i\rangle_{i\in T^*}, \\ \langle \mathsf{uid}'_i, y_i\rangle_{i\in[n]\setminus T^*}, \mathsf{aux} \end{array}\right) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp},\mathsf{st})$$

$$(\mathsf{tx}, \langle \overline{\mathsf{acc}}_i, \mathsf{tk}_i\rangle_{i\in[n]}) \leftarrow \mathsf{Act}\mathcal{O}\left(\begin{array}{c} p_{m,n}, q_m, \emptyset, \langle R_i, \mathsf{aid}_i, \mathsf{uid}_i, \mathsf{tk}_i\rangle_{i\in[m]}, \\ \langle \mathsf{mpk}_i, y_i\rangle_{i\in T^*}, \\ \langle \mathsf{uid}'_i, y_i\rangle_{i\in[n]\setminus T^*}, \mathsf{aux} \end{array}\right)$$

$\mathsf{UID}^\dagger := \{\mathsf{uid}_i : i \in [m]\}$

$\mathsf{ACC}^\dagger := \{\mathsf{acc}_{\mathsf{aid}_i} : i \in [m]\}$

$(b, *, *) \leftarrow \mathsf{Vf}\left(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}}\rangle_{\mathsf{aid}\in U}, p_{m,n}, q_m, \mathsf{tx}, \langle \overline{\mathsf{acc}}_i\rangle_{i\in[n]}\right)$

$* \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{tx}, \langle \mathsf{tk}_i\rangle_{i\in T})$

$(b', *, *) \leftarrow \mathsf{Vf}\left(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}}\rangle_{\mathsf{aid}\in U}, p_{m,n}, q_m, \mathsf{tx}, \langle \overline{\mathsf{acc}}_i\rangle_{i\in[n]}\right)$

return $b \wedge \neg b'$

Figure 5: Privacy and Availability experiments.

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

$(\mathsf{pp}, \mathsf{st}_0) \leftarrow \mathsf{Setup}(1^\lambda)$

$\left(p_{i,m_i,n_i}, q_{i,m_i}, \mathsf{tx}_i, \langle \overline{\mathsf{acc}}_{i,j} \rangle_{j\in[n_i]}\right)_{i\in[t]} \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{st}_0)$

for $i \in [t]$ **do**

$$\left(b_i, \mathsf{st}_{i+1}, \langle \mathsf{acc}_{i+1,j} \rangle_{j\in U_{i+1}}\right) \leftarrow \mathsf{Vf}\left(\begin{array}{c} \mathsf{st}_i, \langle \mathsf{acc}_{i,j} \rangle_{j\in U_i}, p_{i,m_i,n_i}, q_{i,m_i}, \\ \mathsf{tx}_i, \langle \overline{\mathsf{acc}}_{i,j} \rangle_{j\in[n_i]} \end{array}\right)$$

$\mathsf{unused} := \sum_{i\in[t]} n_i - \sum_{i\in[t]} m_i$; $\mathsf{valid\_txs} := (\forall i \in [t],\ b_i = 1)$

$\mathsf{sustainable\_state} := (|\mathsf{st}_t| \le k \cdot \beta(\lambda) \cdot \mathsf{unused} \wedge |U_t| \le k \cdot \mathsf{unused})$

return $\mathsf{valid\_txs} \wedge \neg \mathsf{sustainable\_state}$

Figure 6: Sustainability experiment.

the true number of unused accounts.

Definition 3.6 (Sustainability)**.** *Let $k \in \mathbb{N}$. A decentralised anonymous system Ω for $(\mathcal{P}, \mathcal{Q})$ is said to be k-sustainable if there exists $\beta(\lambda) \in \mathsf{poly}(\lambda)$ such that for any PPT adversary $\mathcal{A}$, $\Pr\left[\mathsf{Sustainability}_{\Omega,\mathcal{A},k,\beta}(1^\lambda) = 1\right]$ is negligible in λ, where $\mathsf{Sustainability}_{\Omega,\mathcal{A},k,\beta}$ is defined in Figure 6. Ω is said to be sustainable if it is k-sustainable for some k.*

4 Sustainable Anonymous System

We present a generic construction of DAS with all desired properties. A main ingredient is an account partitioning technique which leads to sustainability while retaining other properties. We also discuss how to deploy the partitioning technique and instantiate the cryptographic building blocks.

4.1 Construction

We present a construction of DAS for an arbitrary attribute predicate family $\mathcal{P}$ and a specific ring predicate family $\mathcal{Q}$.

Fix any chunk size $k \in \mathbb{N}$. To define $\mathcal{Q}$ (parametrised by k), let $\mathsf{Partition}$ be a function that maps $U \subseteq \mathbb{N}_0$ to

$$\mathsf{Partition}(U) := \{U \cap (i \cdot k + \mathbb{Z}_k) : i \in \mathbb{N}_0\}.^5$$

Let $\mathcal{Q}$ be a singleton family consisting of a single predicate q which accepts $(U, (R_i)_{i\in[m]})$ if and only if for all $i \in [m]$ there exists a chunk $C_i \in \mathsf{Partition}(U)$ such that $R_i \subseteq C_i$.

Let COM be a commitment scheme for message space $\mathcal{X}$, TAG a tagging scheme, and ARG a non-interactive argument system. Consider statement-witness tuples $(\mathsf{stmt}, \mathsf{wit})$[6]:

$$\mathsf{stmt} = (p, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid}\in R}, \{(R_i, \mathsf{tag}_i)\}_{i\in[m]}, \langle \overline{\mathsf{acc}}_i \rangle_{i\in[n]}),$$

where $\mathsf{acc}_{\mathsf{aid}} = (\mathsf{pk}_{\mathsf{aid}}, \mathsf{com}_{\mathsf{aid}})$ and $\overline{\mathsf{acc}}_i = (\overline{\mathsf{pk}}_i, \overline{\mathsf{com}}_i)$, and

$$\mathsf{wit} = \left((\mathsf{aid}_i, \mathsf{sk}_i, r_i, x_i)_{i\in[m]}, (\mathsf{mpk}_i, \delta_i, y_i, r_i)_{i\in[n]}, \mathsf{aux}\right).$$

Let $\mathcal{R}$ be a relation which accepts statement-witness tuples $(\mathsf{stmt}, \mathsf{wit})$ of the above form satisfying the following:

- $p((x_i)_{i\in[m]}, (y_i)_{i\in[n]}, \mathsf{aux}) = 1$.

[5] For example, if $k = 3$ (so that $i \cdot k + \mathbb{Z}_k = \{3i, 3i+1, 3i+2\}$ for each $i \in \mathbb{N}_0$) and $U = \{0, 1, 4, 5, 6, 7, 8\}$, then $\mathsf{Partition}(U) = \{\{0, 1\}, \{4, 5\}, \{6, 7, 8\}\}$.

[6] Public parameters for COM and TAG are implicitly included in stmt.

- For all $i \in [m]$, $\begin{cases} \mathsf{aid}_i \in R_i \\ \mathsf{pk}_{\mathsf{aid}_i} = \mathsf{TAG.PKGen}(\mathsf{sk}_i) \\ \mathsf{com}_{\mathsf{aid}_i} = \mathsf{COM.Com}(x_i; r_i) \\ \mathsf{tag}_i = \mathsf{TAG.TagEval}(\mathsf{sk}_i). \end{cases}$
- For all $i \in [n]$, $\begin{cases} (\overline{\mathsf{pk}}_i, \delta_i) = \mathsf{TAG.PKDer}(\mathsf{mpk}_i) \\ \overline{\mathsf{com}}_i = \mathsf{COM.Com}(y_i; r_i). \end{cases}$

Our DAS construction Ω for $(\mathcal{P}, \mathcal{Q})$ is presented in Figure 7. The core of the construction is the Act algorithm, which does the following. For each source account, it derives the tag corresponding to the account secret key. For each output attribute, it generates a new target account.[7] Then, it generates a zero-knowledge argument that the above are computed correctly and that the input and output attributes satisfy the predicate.

The following theorems summarise the properties of our DAS construction. The proofs of Theorems 4.1 to 4.4 are standard, and we defer them to Appendix C. We highlight Theorem 4.5 for the sustainability of our DAS.

Theorem 4.1. *If* TAG *is correct and key-binding and* ARG *is complete for* $\mathcal{R}$*, then* Ω *has integrity.*

Theorem 4.2. *If* COM *is binding and* ARG *is knowledge-sound for* $\mathcal{R}$*, then* Ω *has authenticity.*

Theorem 4.3. *If* TAG *is related-key pseudorandom,* COM *is hiding, and* ARG *is zero-knowledge for* $\mathcal{R}$*, then* Ω *is private.*

Theorem 4.4. *If* TAG *is one-way and related-key pseudorandom, and* ARG *is knowledge-sound and zero-knowledge for* $\mathcal{R}$*, then* Ω *has availability.*

Theorem 4.5. *If* Ω *has authenticity, then* Ω *is* k*-sustainable.*

Proof. Recall that k is the chunk size. Let $\beta(\lambda)$ be the size of a tag produced by the tagging scheme TAG. Since TAG.TagEval is PPT, $\beta(\lambda)$ is a polynomial. Let $\mathcal{A}$ be a PPT adversary in the experiment $\mathsf{Sustainability}_{\Omega,\mathcal{A},\alpha,\beta}(1^\lambda)$. Write $m := \sum_{i\in[t]} m_i$ and $n := \sum_{i\in[t]} n_i$; therefore, $\mathsf{unused} = n - m$. Since Ω has authenticity, $\mathsf{unused} \geq 0$. In the following, suppose that all transcripts output by $\mathcal{A}$ are valid so that $\mathsf{valid\_txs} = 1$. By the definition of $\mathcal{Q}$, each ring $R_{i,j}$ included in each transcript tx_i satisfies $R_{i,j} \subseteq C_{i,j}$ for some chunk $C_{i,j} \in \mathsf{Partition}([n])$, for all $i \in [t]$ and $j \in [m_i]$. Since Ω has authenticity, a ring R can only appear at most k times across all transcripts. In other words, each chunk $C \in \mathsf{Partition}([n])$ covers at most k rings appearing in transactions. The number of rings that can be covered by all chunks combined is thus $\lceil n/k \rceil \cdot k$. The number of chunks that are not covering k rings is at most $\min\{\lceil n/k \rceil, \lceil n/k \rceil \cdot k - m\} \leq n - m + k - 1 = \mathsf{unused} + k - 1 \leq k \cdot \mathsf{unused}$. By the construction of GC, the tags and accounts corresponding to chunks covering k rings are removed. Therefore, we have $|\mathsf{st}_t| \leq k \cdot \beta(\lambda) \cdot \mathsf{unused}$ and $|U_t| \leq k \cdot \mathsf{unused}$. In other words, $\mathsf{Sustainability}_{\Omega,\mathcal{A},k,\beta}(1^\lambda) = 0$. □

4.2 On Sustainability versus Privacy

For a DAS scheme to be s-sustainable, we show that s must be as large as the minimum ring size n ever used in the sequence of anonymous actions. In the following, consider a private DAS scheme Ω. As Ω is private, only the ring-membership relations can possibly reveal whether an account has been used or not.

Suppose time t has elapsed, i.e. there have been t actions, and let N be the number of accounts created so far. For each $i \in [t]$, suppose the i-th action specifies a ring R_i of size at least n, and exactly $n - 1$ accounts in R_i are used. Further suppose that all rings $R_1, \ldots, R_t$ are disjoint, which forces $N \geq nt$. The number of unused accounts is $N - (n-1)t$, while the number of potentially unused accounts is N, i.e. none of the accounts can be ruled out. The sustainability parameter satisfies

$$\frac{1}{s} \leq \frac{N-(n-1)\cdot t}{N} = 1 - \frac{(n-1)\cdot t}{N} \leq 1 - \frac{n-1}{n} = \frac{1}{n}, \text{ i.e. } s \geq n.$$

Our users can pick rings of size at most k, we have $s \geq k$, the chunk size, which matches the k-sustainability of our construction, i.e. sustainability deteriorates with anonymity.

[7] This corresponds to the one-time account generation in Omniring [Lai+19].

$\underline{\mathsf{Setup}(1^\lambda)}$

$\mathsf{pp}_{\mathsf{TAG}} \leftarrow \mathsf{TAG.Setup}(1^\lambda)$

$\mathsf{pp}_{\mathsf{COM}} \leftarrow \mathsf{COM.Setup}(1^\lambda)$

$\mathsf{pp}_{\mathsf{ARG}} \leftarrow \mathsf{ARG.Setup}(1^\lambda)$

$\mathsf{pp} := (\mathsf{pp}_{\mathsf{TAG}}, \mathsf{pp}_{\mathsf{COM}}, \mathsf{pp}_{\mathsf{ARG}})$

return $(\mathsf{pp}, \mathsf{st} := \emptyset)$

$\underline{\mathsf{MKGen}(\mathsf{pp})}$

$\mathsf{msk} \leftarrow \mathsf{TAG.SKGen}(\mathsf{pp}_{\mathsf{TAG}})$

$\mathsf{mpk} \leftarrow \mathsf{TAG.PKGen}(\mathsf{msk})$

return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{AKDer}(\mathsf{msk}, \mathsf{tk})}$

parse tk **as** (δ, x, r)

$\mathsf{sk} \leftarrow \mathsf{TAG.SKDer}(\mathsf{msk}, \delta)$

$\mathsf{ask} := (\mathsf{sk}, r)$

return (ask, x)

$\underline{\mathsf{GC}(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}) \quad /\!/ \text{ subroutine of } \mathsf{Vf}}$

parse st **as** $\langle \mathsf{Tag}_C \rangle_{C \in \mathsf{Partition}(U)}$; $U' := U$

for $C \in \mathsf{Partition}(U)$ **do**

  if $|\mathsf{Tag}[C]| = |C|$ **then** $U' := U' \setminus C$

return $(\langle \mathsf{Tag}_C \rangle_{C \in \mathsf{Partition}(U')}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U'})$

$\underline{\mathsf{SChk}(\mathsf{acc}, \mathsf{ask}, x)}$

parse acc **as** $(\mathsf{pk}, \mathsf{com})$

parse ask **as** (sk, r)

return $\begin{cases} \mathsf{pk} = \mathsf{TAG.PKGen}(\mathsf{sk}) \\ \mathsf{com} = \mathsf{COM.Com}(x; r) \end{cases}$

$\underline{\mathsf{TChk}(\mathsf{acc}, \mathsf{mpk}, \mathsf{tk}, y)}$

parse acc **as** $(\mathsf{pk}, \mathsf{com})$

parse tk **as** (δ, x, r)

return $\begin{cases} x = y \\ \mathsf{com} = \mathsf{COM.Com}(x; r) \end{cases}$

$\underline{\mathsf{Act}\left(\begin{array}{c} \mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}, p_{m,n}, \\ \langle R_i, \mathsf{aid}_i, \mathsf{ask}_i, x_i \rangle_{i \in [m]}, \langle \mathsf{mpk}_i, y_i \rangle_{i \in [n]}, \mathsf{aux} \end{array}\right)}$

parse st **as** $\langle \mathsf{Tag}_C \rangle_{C \in \mathsf{Partition}(U)}$

for $i \in [m]$ **do**

  parse ask_i **as** (sk_i, r_i)

  $\mathsf{tag}_i \leftarrow \mathsf{TAG.TagEval}(\mathsf{sk}_i)$

for $i \in [n]$ **do**

  $(\overline{\mathsf{pk}}_i, \delta_i) \leftarrow \mathsf{TAG.PKDer}(\mathsf{mpk}_i)$

  $\overline{\mathsf{com}}_i \leftarrow \mathsf{COM.Com}(y_i; s_i) \quad /\!/ \text{ with uniform randomness } s_i$

  $\overline{\mathsf{acc}}_i := (\overline{\mathsf{pk}}_i, \overline{\mathsf{com}}_i); \quad \mathsf{tk}_i := (\delta_i, y_i, s_i)$

$R := \bigcup_{i \in [m]} R_i$

$\mathsf{stmt} := \left(p_{m,n}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in R}, \{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \langle \overline{\mathsf{acc}}_i \rangle_{i \in [n]}\right)$

$\mathsf{wit} := \left((\mathsf{aid}_i, \mathsf{sk}_i, r_i, x_i)_{i \in [m]}, (\mathsf{mpk}_i, \delta_i, y_i, s_i)_{i \in [n]}, \mathsf{aux}\right)$

$\pi \leftarrow \mathsf{ARG.Prove}(\mathsf{stmt}, \mathsf{wit})$

$\mathsf{tx} := \left(\{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \pi\right)$

return $\left(\mathsf{tx}, \langle \overline{\mathsf{acc}}_i, \mathsf{tk}_i \rangle_{i \in [n]}\right)$

$\underline{\mathsf{Vf}(\mathsf{st}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U}, p_{m,n}, q_m, \mathsf{tx}, \langle \overline{\mathsf{acc}}_i \rangle_{i \in [n]})}$

parse st **as** $\langle \mathsf{Tag}_C \rangle_{C \in \mathsf{Partition}(U)}$

parse tx **as** $\left(\{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \pi\right)$

$R := \bigcup_{i \in [m]} R_i$

$\mathsf{stmt} := \left(\begin{array}{c} p_{m,n}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in R}, \\ \{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \langle \overline{\mathsf{acc}}_i \rangle_{i \in [n]} \end{array}\right)$

if $\begin{cases} p_{m,n} \in \mathcal{P} \wedge q_m(U, (R_i)_{i \in [m]}) = 1 \\ q_m \in \mathcal{Q} \wedge \mathsf{ARG.Vf}(\mathsf{stmt}, \pi) = 1 \\ \{\mathsf{tag}_i\}_{i \in [m]} \cap \bigcup_{C \in \mathsf{Partition}(U)} \mathsf{Tag}_C = \emptyset \end{cases}$ **then**

  $U' := U \cup ([n] + \max(U) + 1); U'' := \mathsf{Partition}(U')$

  $\mathsf{acc}_{i + \max(U) + 1} := \overline{\mathsf{acc}}_i, \ \forall i \in [n]$

  $\mathsf{Tag}_C := \mathsf{Tag}_C \cup \{\mathsf{tag}_i\}_{i \in [m] : R_i \subseteq C}, \ \forall C \in U''$

  $\mathsf{st}'' \leftarrow \mathsf{GC}(\langle \mathsf{Tag}_C \rangle_{C \in \mathsf{Partition}(U')}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in U'})$

  return $(1, \mathsf{st}')$

else return $(0, \mathsf{st})$

Figure 7: Generic construction of a Decentralised Anonymous System

4.3 Instantiations

To use the DAS in Figure 7, one needs to specify how rings are sampled, such that they satisfy a predicate $q_m \in \mathcal{Q}$, and instantiate the building blocks COM, TAG, and ARG.

Partitioning Ring Samplers A natural way to sample rings with respect to the predicate family $\mathcal{Q}$ is to use a partitioning sampler [Ron+21]. In more detail, let Π be a partitioning sampler hardwired with the same partition function $\mathsf{Partition}$ as in $\mathcal{Q}$. On input an account identifier aid, $\Pi(\mathsf{aid})$ simply returns the unique chunk $C \in \mathsf{Partition}(\mathbb{N}_0)$ containing aid. The partitioning sampler achieves near-optimal anonymity [Ron+21] and is robust against graph-based attacks [Egg+22]. We refer to those works for more details.

Cryptographic Building Blocks A strategy to efficiently instantiate the cryptographic building blocks is to instantiate the commitment scheme COM and the tagging scheme TAG over a common mathematical structure, e.g. a cyclic group $\mathbb{G}$ of prime-order q, and pick an argument system ARG which supports proving relations over the said structure, e.g. $\mathbb{G}$, natively. This avoids the concretely expensive NP reductions from the relations induced by COM and TAG and those natively supported by ARG.

Group-based Instantiation We could instantiate COM by (the vector-generalisation of) Pedersen's commitment [Ped92] over $\mathbb{G}$, TAG by any constructions in [Lai+19] over $\mathbb{G}$, and ARG by a zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) for $(\mathbb{Z}_q, \mathbb{G})$-arithmetic relations (e.g. [LMR19; ACR21]). Instantiated as such, an account acc is of a fixed polynomial size independent of the size of its associated attribute x. Furthermore, since rings output by partitioning sampler Π admits a succinct representation, e.g. a chunk identifier, and proofs generated by a zk-SNARG are succinct, i.e. of size poly-logarithmic in the statement size, an action transcript tx is also of size poly-logarithmic in the statement size.

The above generic construction of DAS is similar to and generalises that of the RingCT scheme Omniring [Lai+19]. Many of the differences are inherited from the syntactical differences between DAS and RingCT highlighted in Section 3. Below, we summarise the similarities and key differences:

- The setup, master key generation, account key derivation, source checking, and target checking algorithms are almost identical to their RingCT counterparts (up to naming).
- The Action algorithm combines the one-time account generation and spend algorithms in RingCT, and supports more general actions (than just RingCT transactions).
- The verification algorithm differs from its RingCT counterpart in two crucial ways (on top of checking more general attribute predicates than RingCT's balance predicate): 1. It additionally checks that a ring predicate is satisfied. 2. It attempts to shrink the state via garbage collection.

The inclusion of ring predicates and garbage collection are the core components for achieving sustainability.

5 Applications

We discuss two instances of DAS in our framework by specifying their *attribute* spaces and classes of *predicates*. Specifically, we consider anonymous cryptocurrencies as a simple example, as well as the more complex example of decentralised anonymous social networks, which requires distinguishing different *types* of accounts.

We do not intend to formalise the security properties for these high-level applications. Despite being highly application-specific, they are natural extensions of the generic security properties that should be trivially implied by combining the latter with other basic tools. Our goal is to highlight general techniques of using our generic construction blueprint to build more complex applications.

5.1 Anonymous Cryptocurrencies

Cryptocurrencies are a means of tracking financial transactions in a decentralised system. A user holds several accounts—often called unspent transaction outputs—which are considered coins of different values. A transaction is accepted by the system if it does not create money but instead only redistributes it, i.e. the value of new accounts should be equal to the value of the used accounts. We encode the account balance as a bounded integer and verify the balance by the predicate $(\sum_{i\in[m]} x_i - \sum_{i\in[n]} y_i = 0) \wedge (\forall i \in [n],\ 0 \le y_i \le \beta)$ where β is a public upper bound of an account balance. With this, we essentially recover RingCT [NM16; Lai+19], now equipped with sustainability. This naturally generalises to a setting with multiple types of assets in the same system where the predicate needs to verify the equality of each type of asset.

5.2 Decentralised Anonymous Social Networks

DAS is not confined to ring-signature applications but applies to systems allowing users to hide within an anonymity set. With the recent rise of decentralised social networks and the need to block misbehaving users for quality services, we exemplify with *Decentralised Anonymous Social Network* (DASN).

5.2.1 Overview

A DASN is a decentralised means of allowing users to engage in different activities, e.g. posting messages and rating other users' messages. We will use "post" as a wildcard to refer to any action that a user can perform. A new user can dynamically join the system by registering themself and obtaining an account. The account encodes, say, the score of the user, along with other information used by the DASN internally.

A user Alice can make an anonymous post if her score satisfies a posting policy and she has only a few unclaimed ratings. Each post is given a unique identifier which is bound to the author, and is subject to evaluation by other users governed by some publicly defined evaluation policies. After a publicly specified evaluation period elapsed, all ratings of the post will be aggregated, finalised, and are to be claimed by the author. Each user Alice will be forced by the DASN to anonymously claim the (possibly negative) ratings of these posts, i.e. updating her account to encode her new score.

DASN is also known as anonymous blocklisting/reputation. Existing works focus on attaining complexity sublinear in the size of the ever-growing rating list [RMM22; MC23]. Naturally, a DASN can be constructed from a DAS, where joining, posting, rating, and claiming simply correspond to performing actions with different classes of instructions with different predicates. It is intuitive that such a DASN construction inherits the sustainability and availability of the base scheme.

5.2.2 Construction

Setup First, the DASN public parameters additionally specify a pseudorandom function PRF, an upper bound $n \in \mathbb{N}$ of unclaimed ratings allowed, a finalising mechanism determining when the evaluation period of a post should end, and the families of admissible predicates for joining, posting, rating, and claiming. The attribute (t, k, s, I) encoded in an account consists of a type $t \in \{\texttt{user}, \texttt{rating}\}$, a PRF secret key k, a score s, and a set I storing at most n post identifiers. By construction, the type t of an account is public, and each rating account (i.e. with $t = \texttt{rating}$) is associated with a unique post identifier uid. Let F denote the set of post identifiers that the rating accounts are associated with.

Joining To join the DASN, Alice generates a DAS master key pair $(\mathsf{mpk}, \mathsf{msk})$. She also picks a fresh PRF secret key k. She then performs an action which does not input any source account and outputs a single target account under mpk encoding $(t = \texttt{user}, k, s = 0, I = \emptyset)$. The joining predicate that Alice picks for this transaction should check that $t = \texttt{user}$, score $s = 0$ (or other admissible initial scores), and $I = \emptyset$. Alice then publishes the action transcript. After the action is verified by other users, Alice's account will be recognised and included in the DASN state. In the following, we will assume that Alice owns a user account acc encoding $(\texttt{user}, k, s, I)$.

Posting To write a post, Alice generates a fresh DAS master key pair $(\mathsf{mpk}', \mathsf{msk}')$ where the master public key mpk' serves as the post identifier, i.e. $\mathsf{uid} = \mathsf{mpk}'$. She then performs an action which takes her

user account as the source account and outputs a target account under mpk encoding $(\texttt{user}, k, s, I \cup \{\mathsf{uid}\})$. The posting predicate checks that the score s is admissible for writing a post, none of the rating accounts indexed by F is owned by her (i.e. $F \cap I = \emptyset$)[8], the target attribute is correctly computed, and she has at most $n-1$ unclaimed posts before posting, i.e. $|I| \leq n-1$. Alice then publishes the action transcript along with the post identifier uid (and the post itself, which the DASN does not care about).

To verify the post, the verifiers check that the identifier uid is not associated with any post under evaluation or waiting to be claimed, in addition to that the action is valid. The target account created by this action will become Alice's new user account, while the old user account can no longer be used.

Rating Suppose that each user can rate each post at most once, and that a rating takes the form of a (possibly negative) score to be added to the post author's score. Let $\mathsf{uid} = \mathsf{mpk}'$ be the identifier of the post under evaluation. To rate it with a score s', Alice computes an "evaluation tag" $y = \mathsf{PRF}(k, \mathsf{uid})$ bound to uid and her PRF key k. She then performs an action which inputs her user account acc as the source account and a re-randomised version of acc as the target account. The rating predicate is hardwired with uid and tag y, and checks that the source and target attributes are identical and $y = \mathsf{PRF}(k, \mathsf{uid})$. Alice then publishes the action transcript along with score s' and tag y. The verifiers check that the action transcript is valid, y is unique, and post uid is still under evaluation. If so, they agree to aggregate s' to the score of uid.

Finalising When the finalising mechanism concludes the evaluation period of post $\mathsf{uid} = \mathsf{mpk}'$, which got an aggregated score s', the verifiers agree to perform an action that takes no source accounts and outputs a target account under mpk' encoding $(\texttt{rating}, k' = \bot, s', I' = \emptyset)$. The action is performed using public randomness. After verification, the verifiers agree to insert uid into the set F.

Claiming Scores Alice cannot post if she already has too many unclaimed ratings. To keep using the DASN, she must claim any unclaimed finalised ratings of her past posts.

Let $\bar{I} \subseteq I$ be the identifiers of Alice's posts which have been finalised. Recall that each identifier $\mathsf{uid} \in \bar{I}$ is actually a DAS master public key mpk' whose corresponding msk' is known to Alice. She can thus recover the secret key of the corresponding rating account and the score s_{uid} encoded within. To claim these ratings, Alice performs an action using her user account acc as well as all rating accounts associated with $\bar{I}$ as source accounts. The action outputs, as the target account, an updated version of Alice's account encoding the updated attribute $(\texttt{user}, k, s + \sum_{\mathsf{uid} \in \bar{I}} s_{\mathsf{uid}}, I \setminus \bar{I})$. The claiming predicate verifies that one source account is of type $\texttt{user}$ and the others are of type $\texttt{rating}$, and that the target attribute is computed correctly. After verification, if some rating accounts are removed from the DASN state, e.g. by GC with the construction in Figure 7, the corresponding post identifiers are removed from F.

6 Garbage Collection

We propose a general algorithm for detecting used accounts given a set U of account identifiers and a sequence $R = (r_i)_i$ of rings with $r_i \subseteq U$. (Section 6 uses lowercase letters for rings to align with the notation of [Egg+22].) We highlight that, unlike (the core component of) the GC algorithm in Figure 7, which only works for rings sampled by a partitioning sampler, the algorithm proposed in this section can detect surely-used accounts regardless of how the rings are sampled. In what follows we call this algorithm a garbage collector.

Section 6.1 gives a high-level description of how the garbage collection problem can be viewed as a graph problem. Section 6.3 formally shows that our garbage collector is correct, i.e. its outputs are used accounts, and it is also optimal, in the sense that it outputs all surely-used accounts that one can infer from the sequence of ring-memberships $R = (r_i)_i$. Information other than ring-memberships is not taken into account by our garage collector. Other information sources could be, for example, the action

[8] By the sustainability of the underlying DAS, the cardinality of F is not much larger than the number of finalised but unclaimed ratings. Therefore, the computation complexity of proving and verifying $F \cap I = \emptyset$ should be moderate. In case an even higher efficiency is desired, anonymity can be traded for efficiency by making F structured, e.g. sorted, and having Alice reveal a subset $J \subseteq F$ in plain such that proving and verifying $(J \cap I = \emptyset \wedge (F \setminus J) \cap I = \emptyset)$ is more efficient than doing so for $F \cap I = \emptyset$.

$\mathsf{GC}(U, R)$

$E := \{(u, r) : r \in R, u \in r\}$

$G := (U, R, E),\ G' \leftarrow \mathsf{Core}(G)$

$\mathcal{C} := \{C = (U_C, R_C, E_C) : C \text{ is a component in } G'\}$

$R^* := \bigcup_{C \in \mathcal{C}: |U_C| = |R_C|} R_C;\ U^* := \mathsf{nb}_E(R^*),\ E^* := E \cap (U^* \times R^*)$

return $G^* := (U^*, R^*, E^*)$

Figure 8: Garbage collector.

behaviours of users, or deployment issues such that a ring no longer computationally hides its source account.

6.1 Overview

Our garbage collector is based on the notion of transaction graphs [Egg+22] and their Dulmage-Mendelsohn (DM) decompositions [DM58], also known as their cores. The transaction graph induced by (U, R) is a bipartite graph $G = (U, R, E)$ where the nodes of G are labelled by elements of U and R, and an edge (u, r) is in E if $u \in r$, i.e. u is a member of the ring r. In the general case where r contains $\ell > 1$ source accounts, r is duplicated into ℓ copies of itself in the transaction graph.

There exists at least one maximum matching in G in which all rings in R are saturated (maximum matching M saturates v if v is in M), reflecting the fact that each ring must have been sampled by a source account. Many maximum matchings could exist in G, each representing a possible assignment from a source account to a ring. The core of G, denoted by $\mathsf{Core}(G)$, is the union of all maximum matchings of G. Edges in G but missing in $\mathsf{Core}(G)$ represent impossible assignments from source accounts to rings.

Given the set of all maximum matchings, whether an account u has been used or not can be determined by logical deduction. If there exists a maximum matching M in G which does not saturate u, u is possibly yet to be used (for M being a possible source-account-to-ring assignment). The converse is also true: If u is saturated in every maximum matching in G (i.e. all possible source-account-to-ring assignments include u), then u has surely been used in an anonymous action. Consequently, the set of all surely-used accounts is equivalent to the set of all accounts that are saturated in every maximum matching in G. Directly computing the set of all maximum matchings of a bipartite graph is, however, computationally hard[9].

An alternative view on detecting used accounts, which we will show to be equivalent, is to look at the set of accounts covered by a set of rings. If the cardinalities of the two sets are equal, surely every account in the covered set has been used. For example, it is certain that both account 0 and 1 have been used if a ring $r = \{0, 1\}$ appears twice. Generalising, we define the notion of closed ring-induced subgraphs. A subgraph $H = (U', R', E')$ of a transaction graph $G = (U, R, E)$ is said to be ring-induced if E' contains all edges in E connecting to R', written as $U' = \mathsf{nb}_E(R')$. The graph H is closed if the number of neighbours of the rings in R' equals the number of rings in R', i.e. $|\mathsf{nb}_{E'}(R')| = |R'|$. If H is a closed ring-induced subgraph of G, all accounts in U' must have been used.

In Figure 8, we propose a sub-quadratic-time garbage collector GC for finding closed ring-induced subgraphs of a transaction graph G induced by any given accounts U and sequence of rings R. Our garbage collector first computes the core $\mathsf{Core}(G)$ of the transaction graph G induced by (U, R). It then outputs the subgraph of G ring-induced by $\bigcup_C R_C$, where $C = (U_C, R_C, E_C)$ runs through all closed components of $\mathsf{Core}(G)$. The run-time of GC is dominated by the time of computing $\mathsf{Core}(G)$, which takes time $O(|U|^{1/2} \cdot |E|)$ [Tas12]. In Section 6.3, we prove that: 1) GC finds the largest closed ring-induced subgraph of G, and 2) this is equivalent to finding all accounts saturated in every maximum matching in G. These imply GC is optimal – it returns all surely-used accounts that we can infer from (the graph representation of) the sets (U, R).

[9]The counting problem of all maximum matchings in a bipartite graph is #P-complete [Vad01], and the search problem is no easier.

In Section 6.2, we define all notions required to analyse our garbage collector. We then outline the strategy for proving its correctness and optimality in Section 6.3 and defer the proofs of the technical lemmas to Appendix B.

6.2 Definitions

Definition 6.1 (Transaction Graph [Egg+22])**.** *A transaction graph $G = (U, R, E)$ is a bipartite graph with a maximum matching of size $|R|$.*

Definition 6.2 (Core [DM58])**.** *The core of a bipartite graph $G = (U, R, E)$, denoted by $\mathsf{Core}(G) = (U, R, E')$, is a subgraph of G where E' is the union of all maximum matchings in G.*

Definition 6.3 (Neighbours)**.** *For a bipartite graph $G = (U, R, E)$ and a node $r \in R$, the set of neighbours of r under edges $E' \subseteq E$ is $\mathsf{nb}_{E'}(r) := \{u \in U : (u, r) \in E'\}$. Generalising, the set of neighbours of $R' \subseteq R$ under $E' \subseteq E$ is $\mathsf{nb}_{E'}(R') := \{u \in U : (u, r) \in E', r \in R'\}$.*

Definition 6.4 (Ring-Induced Subgraphs)**.** *For a transaction graph $G = (U, R, E)$, a subgraph $H = (U', R', E')$ of G is said to be ring-induced by R', if $U' = \mathsf{nb}_{E'}(R')$ and $E' = \{(u, r) \in E : r \in R'\}$. We say that H is ring-induced if it is ring-induced by some R'.*

Definition 6.5 (Closeness)**.** *A transaction graph $G = (U, R, E)$ is said to be closed if $|\mathsf{nb}_E(R)| = |R|$.*

6.3 Analysis

We first show that the garbage collector in Figure 8 outputs the largest closed ring-induced subgraph of the transaction G induced by the input (U, R). Note that the notion of largest is well-defined for closed ring-induced subgraphs since the union of two such graphs is also a closed ring-induced subgraph.

Theorem 6.6. *If a transaction graph G has a closed ring-induced subgraph H, then $\mathsf{Core}(H) = \bigcup_i H_i$ where each H_i is a closed component of $\mathsf{Core}(G)$.*

Theorem 6.7. *Let G be a transaction graph. If $\{H_i = (U_i, R_i, E_i)\}_i$ is the collection of all closed components of $\mathsf{Core}(G)$, then the subgraph of G induced by $\bigcup_i R_i$ is closed.*

We defer the technical proof of Theorem 6.6 and Theorem 6.7 to Appendices B.1 and B.2, respectively.

To see why Theorems 6.6 and 6.7 imply that GC given in Figure 8 outputs the largest closed ring-induced subgraph of G, we observe the following: On the one hand, Theorem 6.7 guarantees that GC always returns a closed ring-induced subgraph H of G. On the other hand, Theorem 6.6 implies that the subgraph H found by GC is the largest.

Using the above, we next show that finding the set of accounts saturated in every maximum matching in G (i.e. the surely-used accounts) is equivalent to finding the set of accounts in the largest closed ring-induced subgraph of G. This concludes that GC in Figure 8 is correct and optimal.

Theorem 6.8. *Let G be a transaction graph. An account u is saturated in all maximum matchings M in G if and only if u belongs to the biggest closed ring-induced subgraph H of G.*

Proof. For the if part, u is an account node in H, which is a closed ring-induced subgraph of G; Theorem 6.6 implies that u belongs to a closed component C of $\mathsf{Core}(G)$. Since C is closed, all maximum matchings in C, and hence G, saturate u.

For the only-if part, given that u is saturated in all maximum matchings M in G, we claim that the unique component C in $\mathsf{Core}(G)$, which contains u, must be closed. Suppose that is the case. Theorem 6.7 implies that u belongs to a closed ring-induced subgraph H' of G. Since H is the biggest closed ring-induced subgraph of G, H' is a subgraph of H, so u is in H. It remains to prove that the unique component C in $\mathsf{Core}(G)$ which contains u must be closed. C is a transaction graph. We rely on a technical lemma, Lemma 4, which implies that the subgraph C' of C formed by removing u and all edges in C connected to it is also a transaction graph, which means that C' has at least one maximum matching in which all rings in C' are saturated. Let M' be one such maximum matching. Note that M' is a maximum matching of C also, but it does not saturate u. This means that there exists a maximum matching of G which does not saturate u, violating our assumption. □

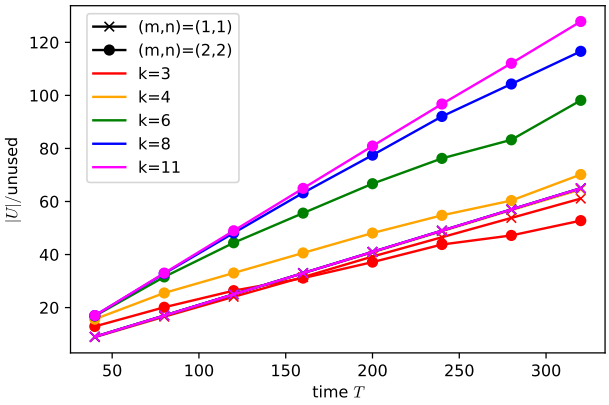

Figure 9: Ratio of uncollected to unused accounts against T.

7 On The Sustainability of Monero

We conclude by empirically evaluating the (un)sustainability of Monero. To clarify, we do not intend to study whether their concrete system is sustainable, but instead, whether the abstract ring sampling strategy of Monero is likely to be sustainable.

7.1 Modelling

The ring sampling strategy currently adopted by Monero can be abstracted as a mimicking sampler [Ron+21], which samples ring members according to an approximated distribution that "mimics" the source account distribution: Each account u_i is sampled as a ring member with probability p_i, where p_i is the (estimated) probability that u_i is a source account at the sampling time. This process repeats until the ring is populated to a predetermined ring size.

More concretely, we consider the following experiment parametrised by an initial number of accounts a (at time $t = 0$), a ring size k, a time bound T, the number of source and target accounts (m, n) in each anonymous action, and a sequence $(S_t)_{t\in[T]}$ of source account distributions. To model the formation of rings, at each time $t \in [T]$, m source accounts are sampled according to the distribution S_t, each of which samples a ring, and n target accounts are spawned. The ring members are sampled using the mimicking sampler, which samples k ring members according also to the distribution S_t.

After T time steps have elapsed, let U be the set containing all initial accounts and all target accounts spawned during the whole process, and $R = (R_i)_{i=1}^{mT}$ be the sequence of all sampled rings. To answer whether a mimicking sampler can provide sustainability, we run GC in Figure 8 on (U, R) to check how many used accounts can be detected.

In our experiment, we let the source account distribution S_t be that induced by the following sampling process: First, an age is sampled from the empirical age distribution [Ron+21][10], which is a shifted Pareto distribution with PDF $\mathsf{Pr}[\mathsf{age} = t] \propto (t + 1)^{-(\alpha+1)}$ and $\alpha = 0.172$. A uniformly random account of the sampled age is then chosen. If no account of the sampled age exists or the sampled account has already been used in a previous action, the procedure is repeated until an unused account is sampled. We picked $a \in \{2, 3, \ldots, 10\}$, $k \in \{2, 3, \ldots, 11\}$, $T \in \{40, 80, \ldots, 320\}$, and $(m, n) \in \{(1, 1), (2, 2)\}$. For each combination of parameters, we performed 1000 runs. Our result is summarised in Figure 9, where we plot the ratio $|U|/\mathsf{unused}$ (following notations in Figure 6) against T for $a = 5$.

7.2 Result and Evaluation

Recall that by definition, a DAS is sustainable only if the ratio $|U|/\mathsf{unused}$ is upper-bounded by some constant for all T. For example, with the proven sustainable partitioning sampler with ring size k, we would obtain a plot where the curve never grows beyond k (for any choice of the other parameters).

[10]The age distribution of Monero is unknown due to its anonymous nature. The empirical distribution [Ron+21] is from Bitcoin data as an estimation for Monero.

In our experiment, in contrast, the concerned ratio strictly increases with T for all sets of parameters, which means the system is keeping an ever-growing set of accounts even when the number of unused accounts remains constant. We remark that as the number of initial accounts a increases (not shown), the ratios for the same (m, n) under various k's converge (to the magenta curve in Figure 9), which is the case when essentially no used account can be collected by GC. Thus, our result says that mimicking sampler is unlikely to offer sustainability.

In both Monero and our experiment, a ring is sampled from a distribution that mimics the source account distribution. Monero differs from our experiment mainly in two ways: 1) Their approximation on the source account distribution is some $\hat{S}_t$ different from our S_t, which is a code-induced distribution that cannot be expressed analytically. 2) Historically, Monero has been updating their ring-sampling strategy continuously, from allowing a ring size of 1 (i.e. no anonymity), to mandating some minimum ring size (and increasing it). Our experiment does not intend to replicate the ring-sampling specifics of Monero, but to answer the more general question of whether a mimicking sampler could offer sustainability. We also note that, the effect of using different approximations S_t and $\hat{S}_t$ should have minimal impact on (the resulting transaction graph and hence) the (un)sustainability result, since any reasonable approximation would by nature assign to a source account some decoy accounts that are spawned in adjacent time. From our result, Monero is unlikely to be sustainable with the adoption of a mimicking sampler. Complementary to our result is the empirical work of [Vij21], which analysed the transaction graph constructed from actual Monero data. Their result implies that, upon running our GC on Monero, no used account would be collected, coinciding with our experimental result.

As additional references, we provide in Figures 11 and 12 further results on mimicking sampler, where we plot the fraction of used accounts that can be collected (i.e. $|U^*|/|R|$ for $(U^*, *, *) \leftarrow \mathsf{GC}(U, R)$) against ring size k and the number of initial accounts a. Our plots show that this fraction quickly approaches zero as k and a increaseFor comparison, the current recommended ring size of Monero is $k = 11$.

Acknowledgments

Chow is supported by General Research Fund (CUHK 14210621 and 14209918), University Grants Committee, Hong Kong. Egger received funding by the European Commission under the Horizon2020 research and innovation programme, Marie Sklodowska-Curie grant agreement No 101034255. Ronge was supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant number 393541319/GRK2475/1-2019), and by the state of Bavaria at the Nuremberg Campus of Technology (NCT).

References

[ACR21] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. "Compressed Σ-Protocols for Bilinear Group Arithmetic Circuits and Application to Logarithmic Transparent Threshold Signatures". In: *ASIACRYPT 2021, Part IV*. Vol. 13093. LNCS. Springer, Heidelberg, Dec. 2021, pp. 526–556. DOI: 10.1007/978-3-030-92068-5_18.

[Bün+20] Benedikt Bünz et al. "Zether: Towards Privacy in a Smart Contract World". In: *FC 2020*. Vol. 12059. LNCS. Springer, Heidelberg, Feb. 2020, pp. 423–443. DOI: 10.1007/978-3-030-51280-4_23.

[DM58] A. L. Dulmage and N. S. Mendelsohn. "Coverings of Bipartite Graphs". In: *Canadian Journal of Mathematics* 10 (1958), 517–534. DOI: 10.4153/CJM-1958-052-0.

[Egg+22] Christoph Egger et al. "On Defeating Graph Analysis of Anonymous Transactions". In: *PoPETs* 2022.3 (July 2022), pp. 538–557. DOI: 10.56553/popets-2022-0085.

[Fau+19] Prastudy Fauzi et al. "Quisquis: A New Design for Anonymous Cryptocurrencies". In: *ASIACRYPT 2019, Part I*. Vol. 11921. LNCS. Springer, Heidelberg, Dec. 2019, pp. 649–678. DOI: 10.1007/978-3-030-34578-5_23.

[Lai+19] Russell W. F. Lai et al. "Omniring: Scaling Private Payments Without Trusted Setup". In: *ACM CCS 2019*. ACM Press, Nov. 2019, pp. 31–48. DOI: 10.1145/3319535.3345655.

[LMR19] Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. "Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography". In: *ACM CCS 2019*. ACM Press, Nov. 2019, pp. 2057–2074. DOI: 10.1145/3319535.3354262.

[MC23] Jack P. K. Ma and Sherman S. M. Chow. "SMART Credentials in the Multi-queue of Slackness". In: *EuroS&P*. IEEE, 2023.

[NM16] Shen Noether and Adam Mackenzie. "Ring Confidential Transactions". In: *Ledger* 1 (Dec. 2016), pp. 1–18. DOI: 10.5195/ledger.2016.34.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO'91*. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[RMM22] Michael Rosenberg, Mary Maller, and Ian Miers. "SNARKBlock: Federated Anonymous Blocklisting from Hidden Common Input Aggregate Proofs". In: *S&P 2022*. IEEE, 2022, pp. 948–965. DOI: 10.1109/SP46214.2022.9833656.

[Ron+21] Viktoria Ronge et al. "Foundations of Ring Sampling". In: *PoPETs* 2021.3 (July 2021), pp. 265–288. DOI: 10.2478/popets-2021-0047.

[Tas12] Tamir Tassa. "Finding all maximally-matchable edges in a bipartite graph". In: *Theoretical Computer Science* 423 (2012), pp. 50–58. DOI: 10.1016/j.tcs.2011.12.071.

[Vad01] Salil P Vadhan. "The complexity of counting in sparse, regular, and planar graphs". In: *SIAM Journal on Computing* 31.2 (2001), pp. 398–427.

[Vij21] Saravanan Vijayakumaran. *Analysis of CryptoNote Transaction Graphs using the Dulmage-Mendelsohn Decomposition*. Cryptology ePrint Archive, Report 2021/760. 2021.

[Wij+18] Dimaz Ankaa Wijaya et al. "Monero ring attack: Recreating zero mixin transaction effect". In: *TrustCom/BigDataSE 2018*. IEEE. 2018, pp. 1196–1201.

[Yu+19] Zuoxia Yu et al. "New Empirical Traceability Analysis of CryptoNote-Style Blockchains". In: *FC 2019*. Vol. 11598. LNCS. Springer, Heidelberg, Feb. 2019, pp. 133–149. DOI: 10.1007/978-3-030-32101-7_9.

A Basic Cryptographic Primitives

We recall the standard definitions of pseudorandom functions, commitment schemes and argument systems.

Definition A.1 (Pseudorandom Functions)**.** *A pseudorandom function* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is associated with an efficiently sampleable key space* $\mathcal{K}$*, a domain* $\mathcal{X}$*, and a codomain* $\mathcal{Y}$*. It is said to be secure if for any PPT adversary* $\mathcal{A}$

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}^{\mathsf{PRF}(k,\cdot),\mathcal{O}_0}(1^\lambda) = 1 \;\middle|\; k \leftarrow \mathcal{K}, f \leftarrow \mathcal{Y}^{\mathcal{X}}\right] \\ - \Pr\left[\mathcal{A}^{\mathsf{PRF}(k,\cdot),\mathcal{O}_1}(1^\lambda) = 1 \;\middle|\; k \leftarrow \mathcal{K}, f \leftarrow \mathcal{Y}^{\mathcal{X}}\right] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{Y}^{\mathcal{X}}$ *denotes the set of all functions from* $\mathcal{X}$ *to* $\mathcal{Y}$*,* $\mathcal{O}_b$ *inputs* $x \in \mathcal{X}$ *and outputs* $\mathsf{PRF}(k, x)$ *if* $b = 0$ *and else* $f(x)$*.*

Definition A.2 (Commitments)**.** *A commitment scheme for message space* $\mathcal{X}$*, randomness space* $\mathcal{R}$*, and commitment space* $\mathcal{C}$ *is a tuple of PPT algorithms* $\mathsf{COM} = (\mathsf{Setup}, \mathsf{Com})$*. The setup algorithm* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *generates the public parameters* pp*. The commitment algorithm* $\mathsf{com} \leftarrow \mathsf{Com}(x)$ *inputs (implicitly) the public parameters* pp*, a message* $x \in \mathcal{X}^\ell$*, and (implicitly) a randomness* $r \in \mathcal{R}$ *and outputs a commitment* $\mathsf{com} \in \mathcal{C}$*. Unless specified, it is understood that* r *is sampled uniformly at random from* $\mathcal{R}$*.*

Definition A.3 (Binding)**.** *A commitment scheme* COM *is binding if for any PPT adversary* $\mathcal{A}$ *it holds that* $\Pr\left[\mathsf{Binding}_{\mathsf{COM},\mathcal{A}}(1^\lambda) = 1\right]$ *is negligible in* λ*, where* $\mathsf{Binding}_{\mathsf{COM},\mathcal{A}}$ *is defined in Figure 10.*

$\mathsf{Binding}_{\mathsf{COM},\mathcal{A}}(1^\lambda)$	$\mathsf{Hiding}^b_{\mathsf{COM},\mathcal{A}}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(x, r, x', r') \leftarrow \mathcal{A}(\mathsf{pp})$	$(x_0, x_1) \leftarrow \mathcal{A}(\mathsf{pp})$
$b_0 := ((x, r) \neq (x', r'))$	$\mathsf{com} \leftarrow \mathsf{Com}(x_b)$
$b_1 := (\mathsf{com}(x; r) = \mathsf{com}(x'; r'))$	$b' \leftarrow \mathcal{A}(\mathsf{com})$
return $b_0 \wedge b_1$	**return** b'

Figure 10: Biding and Hiding experiments for Commitments.

Definition A.4 (Hiding)**.** *A commitment scheme* COM *is hiding if for any PPT adversary* $\mathcal{A}$

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

is negligible in λ*, where* $\mathsf{Hiding}^b_{\mathsf{COM},\mathcal{A}}$ *is defined in Figure 10.*

Definition A.5 (Non-Interactive Arguments)**.** *A non-interactive argument system is a tuple of PPT algorithms* $\mathsf{ARG} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf})$*. The setup algorithm* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *generates the public parameters. The proving algorithm* $\pi \leftarrow \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$ *inputs (implicitly) the public parameters* pp*, a statement* stmt*, and a witness* wit *and outputs a proof* π*. The verification algorithm* $\mathsf{Vf}(\mathsf{stmt}, \pi)$ *inputs (implicitly) the public parameters* pp*, a statement* stmt*, and a proof* π *and outputs a bit* b*.*

Definition A.6 (Completeness)**.** *A non-interactive argument system* ARG *is complete for a relation* $\mathcal{R}$ *if for any* $\lambda \in \mathbb{N}$*, any public parameters* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$*, any pair of statement and witness* $(\mathsf{stmt}, \mathsf{wit})$ *satisfying* $\mathcal{R}$*, and any proof* $\pi \in \mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$*, it holds that* $\mathsf{Vf}(\mathsf{stmt}, \pi) = 1$*.*

Definition A.7 (Knowledge Soundness)**.** *A non-interactive argument system* ARG *is knowledge-sound for a relation* $\mathcal{R}$ *if for any PPT prover* $\mathcal{A}$ *there exists an expected polynomial time knowledge extractor* $\mathcal{E}_\mathcal{A}$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{Vf}(\mathsf{stmt}, \pi) = 1 \\ \wedge\ \mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 0 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ ((\mathsf{stmt}, \pi), \mathsf{wit}) \leftarrow (\mathcal{A}, \mathcal{E}_\mathcal{A})(\mathsf{pp}) \end{array}\right]$$

is negligible in λ*, where we wrote* $(\mathcal{A}, \mathcal{E}_\mathcal{A})$ *for the joint execution with common input and randomness.*

Definition A.8 (Zero-Knowledge)**.** *A non-interactive argument system* ARG *is zero-knowledge for a relation* $\mathcal{R}$ *if there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ *such that for any PPT* $\mathcal{A}$

$$\left|\begin{array}{l} \Pr\left[\mathcal{A}^{\mathcal{O}_0}(\mathsf{pp}) \;\middle|\; \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)\right] \\ -\ \Pr\left[\mathcal{A}^{\mathcal{O}_1}(\mathsf{pp}) \;\middle|\; (\mathsf{pp}, \mathsf{td}) \leftarrow \mathcal{S}_0(1^\lambda)\right] \end{array}\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{O}_b(\mathsf{stmt}, \mathsf{wit})$ *asserts that* $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$ *and returns* $\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$ *if* $b = 0$ *and* $\mathcal{S}_1(\mathsf{td}, \mathsf{stmt})$ *if* $b = 1$*.*

B Proofs for Garbage Collector

B.1 Proof of Theorem 6.6

We restate and prove Theorem 6.6. The required propositions and technical lemmas together with their proofs are provided.

Theorem 6.6. *If a transaction graph* G *has a closed ring-induced subgraph* H*, then* $\mathsf{Core}(H) = \bigcup_i H_i$ *where each* H_i *is a closed component of* $\mathsf{Core}(G)$*.*

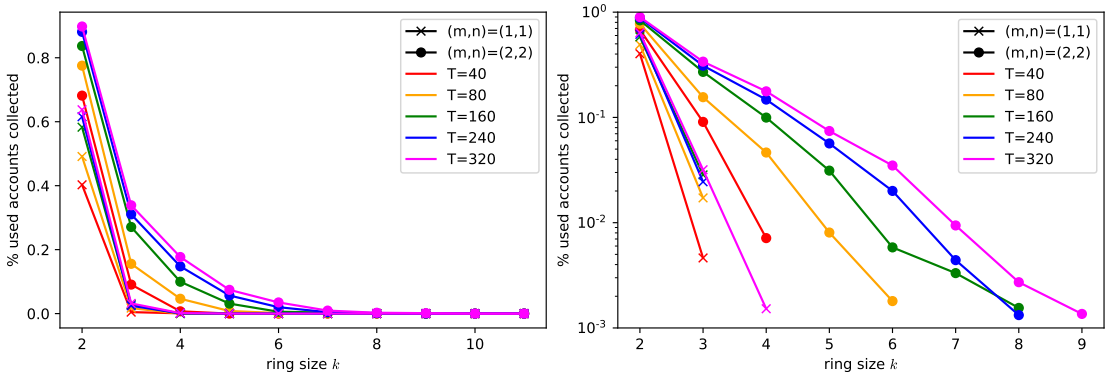

Figure 11: Percentage of unused accounts collected against k for $a = 6$.

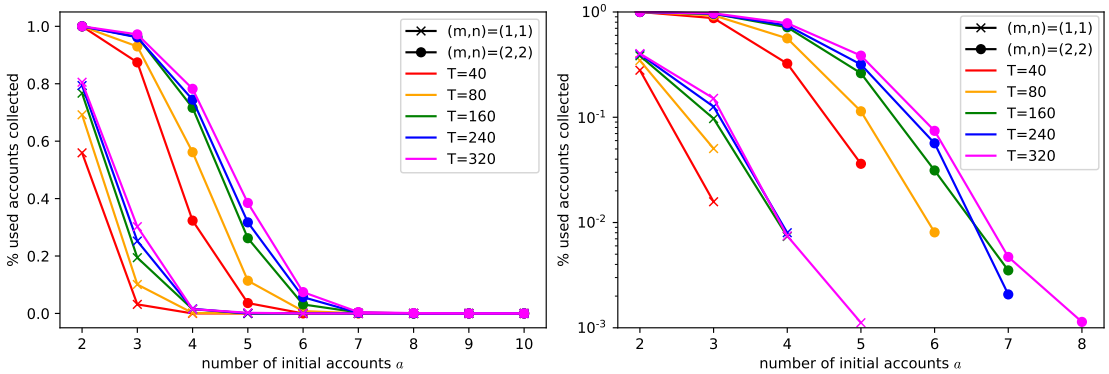

Figure 12: Percentage of unused accounts collected against a for $k = 5$.

Proof. The theorem follows from two technical lemmas, Lemmas 1 and 2. Lemma 1 states that $\mathsf{Core}(H)$ is a closed ring-induced transaction subgraph of $\mathsf{Core}(G)$. Lemma 2 states that, if a transaction graph G' satisfies $G' = \mathsf{Core}(G')$, and H' is a closed ring-induced subgraph of G', then $H' = \bigcup_i H_i$ where each H_i is a closed component of G'. The conditions of two lemmas are satisfied by $G' = \mathsf{Core}(G)$ and $H' = \mathsf{Core}(H)$ due to Lemma 1, the fact that $\mathsf{Core}(G') = \mathsf{Core}(\mathsf{Core}(G)) = \mathsf{Core}(G) = G'$, and the fact that the core of a transaction graph is itself a transaction graph. We can thus apply Lemma 2 to reach the desired conclusion. □

Proposition 1. *If G is a transaction graph and H is a closed ring-induced subgraph of G, then H is also a transaction graph.*

Proposition 2. *If G is a transaction graph, then $\mathsf{Core}(G)$ is also a transaction graph.*

The two propositions above are trivial.

Proposition 3. *If a transaction graph G is closed, then $\mathsf{Core}(G)$ is closed.*

Proof. Let $G = (U, R, E)$, and $\mathsf{Core}(G) = (U, R, E')$. Suppose $\mathsf{Core}(G)$ is not closed, then $|\mathsf{nb}_{E'}(R)| \neq |R|$. Since $\mathsf{Core}(G)$ is also a transaction graph, we have $|U| = |\mathsf{nb}_{E'}(R)| \geq |R|$. Consequently $|\mathsf{nb}_{E}(R)| \geq |\mathsf{nb}_{E'}(R)| > |R|$, contradicting that G is closed. □

Proposition 4. *If a transaction graph G has a closed ring-induced subgraph H, then $\mathsf{Core}(H)$ is a subgraph of $\mathsf{Core}(G)$.*

Proof. It suffices to show that a maximum matching in G can be formed by adding edges in G to any maximum matching in H. Let $G = (U, R, E)$ and $H = (U', R', E')$ where $|U'| = |\mathsf{nb}(R')| = |R'|$ (since H is closed and ring-induced). Let M be any maximum matching in H. Assume for now that $G \setminus H$ is also

a transaction graph. Then, there exists a maximum matching M' in $G \setminus H$. We have that $M \cup M'$ is a maximum matching of G, and the claim follows.

It remains to show that $G \setminus H$ is a transaction graph. Suppose not, then there exists a ring r^* in $G \setminus H$ such that all edges connecting r^* in G are pointing to U'. However, this means the subgraph of G induced by $R' \cup \{r^*\}$ has U' as the set of users, where $|R' \cup \{r^*\}| = |R'| + 1 = |U'| + 1 > |U'|$, contradicting that this ring-induced subgraph is a transaction graph. □

Lemma 1. *If a transaction graph G has a closed ring-induced subgraph H, then $\mathsf{Core}(H)$ is a closed ring-induced transaction subgraph of $\mathsf{Core}(G)$.*

Proof. By Propositions 1 and 2, $\mathsf{Core}(H)$ is a transaction graph. By Proposition 4, $\mathsf{Core}(H)$ is a subgraph of $\mathsf{Core}(G)$. By Proposition 3, $\mathsf{Core}(H)$ is closed.

It remains to show that $\mathsf{Core}(H)$ is ring-induced. Suppose not, then there exists an edge $e^* = (u^*, r^*)$ in $\mathsf{Core}(G) \subseteq G$ connecting $r^* \in R'$ to $u^* \in U$, which is not in $\mathsf{Core}(H)$. Since H is ring-induced, we have $e^* \in E'$, hence $u^* \in U'$. Since e^* is in $\mathsf{Core}(G)$, it is in a maximum matching M of G. Let M' be the subset of M connected to R'. Since e^* is connecting R', $e^* \in M'$. Since H is ring-induced, all edges of M' are connecting R' to U'. Therefore, M' is a maximum matching of H and $e^* \in \mathsf{Core}(H)$, a contradiction. □

Proposition 5. *Let $G = (U, R, E)$ be a transaction graph, and $H = (U', R', E')$ a proper subgraph of G, i.e. $U' \subset U$ or $R' \subset R$ or $E' \subset E$. If G is connected, then $E' \subset E$.*

Proof. Suppose $E = E'$. Then the only ways for H to be a proper subgraph of G are U' being a proper subset of U or R' being a proper subset of R. If the former is true, i.e. $U' \subset U$, then there exists $u \in U \setminus U'$ and an edge $e \in E \setminus E'$ connecting u, for otherwise G is not connected. If the latter is true, i.e. $R' \subset R$, then there exists $r \in R \setminus R'$ and an edge $e \in E \setminus E'$ connecting r, for otherwise G is not connected. Both cases contradict $E = E'$. □

Proposition 6. *Let $G = (U, R, E)$ be a transaction graph, and $H = (U', R', E')$ a closed ring-induced subgraph of G. If $E' \subset E$, then $U' \subset U$ and $R' \subset R$.*

Proof. For any $e = (u, r) \in E \setminus E'$, since H is ring-induced, we have $r \notin R'$; therefore, R' is a proper subset of R (and $|R'| < |R|$). Since H is closed, we have $|U'| = |R'|$. Since G is a transaction graph, we have $|U| \geq |R|$. We immediately arrive at that $|U'| < |U|$, i.e. U' is a proper subset of U. □

Lemma 2. *Let G be a transaction graph where $G = \mathsf{Core}(G)$. If H is a closed ring-induced subgraph of G, then $H = \bigcup_i H_i$ where each H_i is a closed component of G.*

Proof. Write $H = (U_H, R_H, E_H) = \bigcup_{i \in I} H_i$ where H_i is a component of H, and assume for now that each H_i is a component of G.

We first show that each H_i is closed. Suppose that there exists $j \in I$ such that H_j is not closed. Let $J \subseteq I$ be maximal such that H_j is not closed for all $j \in J$. Write $H_j = (U_j, R_j, E_j)$. We have $|U_j| = |\mathsf{nb}(R_j)| \neq |R_j|$. Since H_j is a component of G and, therefore, a transaction graph, we have $|U_j| \geq |R_j|$, hence $|\mathsf{nb}(R_j)| > |R_j|$. Then $|\mathsf{nb}(R_H)| = \sum_{i \in I} |\mathsf{nb}(R_i)| = \sum_{i \in I \setminus J} |R_i| + \sum_{j \in J} |\mathsf{nb}(R_j)| > \sum_{i \in I \setminus J} |R_i| + \sum_{j \in J} |R_j| = |R_H|$, contradicting that H is closed.

It remains to show that H_i is a component of G for all $i \in I$.

Note that H_i is a ring-induced subgraph of H, therefore also a ring-induced subgraph of G_i for some component G_i of G. We argue below that $H_i = G_i$ for all $i \in I$.

Suppose not, then there exists $i^* \in I$ such that $H_{i^*} = (U'_{i^*}, R'_{i^*}, E'_{i^*})$ is a proper subgraph of $G_{i^*} = (U_{i^*}, R_{i^*}, E_{i^*})$. Since G_{i^*} is connected, from Proposition 5, we know that $E'_{i^*} \subset E_{i^*}$. Further, since H_{i^*} is a closed ring-induced subgraph of G_{i^*}, from Proposition 6, we have $U'_{i^*} \subset U_{i^*}$ and $R'_{i^*} \subset R_{i^*}$.

Observe that there exist no edge $e = (u, r) \in E_{i^*}$ where $r \in R'_{i^*}$ and $u \in U_{i^*} \setminus U'_{i^*}$, since H_{i^*} is ring-induced. Consequently, there exists an edge $e^* = (u^*, r^*) \in E_{i^*}$ such that $u^* \in U'_{i^*}$ and $r \in R_{i^*} \setminus R'_{i^*}$, for otherwise, G is not connected and hence not a component.

Now note that since $G = \mathsf{Core}(G)$, e^* belongs to a maximum matching M of G_{i^*}. By the pigeonhole principle, there exists an edge $e^\dagger = (u^\dagger, r^\dagger)$ in M where $r^\dagger \in R'_{i^*}$ and $u^\dagger \in U_{i^*} \setminus U'_{i^*}$. However, H_{i^*} is ring-induced by the rings R'_{i^*}, so $e^\dagger$ is an edge in H_{i^*}, or $e^\dagger \in U'_{i^*}$. We have arrived at a contradiction. □

B.2 Proof of Theorem 6.7

We restate and prove Theorem 6.7. The required propositions and technical lemmas together with their proofs are provided.

Theorem 6.7. *Let G be a transaction graph. If $\{H_i = (U_i, R_i, E_i)\}_i$ is the collection of all closed components of $\mathsf{Core}(G)$, then the subgraph of G induced by $\bigcup_i R_i$ is closed.*

Proof. The proof of this theorem relies on a technical lemma, Lemma 5, which states that if there exists an edge connecting a ring r in a closed component of $\mathsf{Core}(G)$ to a node u outside the component, then u belongs to a closed component of $\mathsf{Core}(G)$. By Lemma 5, any edge e in G connected to a ring r in H_i for some i must be pointing to a node u in H_j for some j. Therefore, the subgraph of G induced by $\bigcup_i R_i$ has neighbours $\mathsf{nb}_E(\bigcup_i R_i) = \bigcup_i U_i$. For all i, since H_i is closed and connected, we have $|U_i| = |R_i|$. Consequently, $|\mathsf{nb}_E(\bigcup_i R_i)| = |\bigcup_i U_i| = |\bigcup_i R_i|$, i.e. then the subgraph of G induced by $\bigcup_i R_i$ is closed. □

Lemma 3. *Let $G = (U, R, E)$ be a transaction graph. If $G = \mathsf{Core}(G)$ and G is connected, then for all ring $r \in R$, it holds that $\deg(r) \geq 2$.*

Proof. Since G is a transaction graph, $\deg(r) \geq 1$ for all $r \in R$. Suppose the claim is false, let r^* be a ring in R with $\deg(r^*) = 1$. Let u^* be the unique neighbour of r^*. We consider two cases.

Case 1: $\deg(u^*) = 1$. In this case, u^* and r^* are not connected to any other nodes in G, violating that G is connected.

Case 2: $\deg(u^*) > 1$. Let $r' \neq r^*$ be another neighbour of u^*. Since $G = \mathsf{Core}(G)$, there exists a maximum matching M containing (u^*, r'). In this M, r^* is not connected to node $u \in U$, since its only neighbour u^* is connected to r'. However, since G is a transaction graph, any maximum matching of G must involve all $r \in R$, including r^*.

As both cases lead to a contradiction, the claim is true. □

Lemma 4. *Let $G = (U, R, E)$ be a transaction graph where $G = \mathsf{Core}(G)$, G is connected and G is not closed. If H is a subgraph of G obtained by removing any one user node $u_0 \in U$ and all edges connecting to u_0, then it holds that H is a transaction graph.*

Proof. Suppose, contrary to the claim, that H is not a transaction graph, i.e. H does not have any matching involving all rings R. Then u_0 is saturated in all maximum matchings in G. We show in the following that this is not possible.

Let M be any maximum matching in G. Let the edge saturating u_0 in M be (u_0, r_0). By Lemma 3, all ring $r \in R$ satisfies $\deg(r) \geq 2$. Since $\deg(r_0) \geq 2$, r_0 has at least one neighbour other than u_0. If there exists a neighbour $u_1 \neq u_0$ of r_0 which is not saturated in M, then we are done since removing (u_0, r_0) and adding (u_1, r_0) results in a maximum matching in which u_0 is not saturated.

Suppose all neighbours of r_0 are saturated in M. For each neighbour $u_1 \neq u_0$, let the edge in M saturating u_1 be (u_1, r_1). Since $\deg(r_1) \geq 2$, r_1 has at least one neighbour other than u_1. If there exists a neighbour $u_2 \neq u_1$ of r_1 which is not saturated in M, then we are done since removing (u_0, r_0) and (u_1, r_1), and adding (u_1, r_0) and (u_2, r_1) results in a maximum matching in which u_0 is not saturated.

Suppose all neighbours of r_1 are saturated in M. For each neighbour $u_2 \neq u_1$, let the edge in M saturating u_2 be (u_2, r_2). Consider all infinitely many possible sequences $(u_0, u_1, u_2, \ldots)$ produced using the above procedure. Since G is not closed, $|U| = |\mathsf{nb}_E(R)| > |R|$. However, since M is a maximum matching, we have $|U(M)| = |R|$, where $U(M) = \{u \in U : (u, r) \in M\}$ denotes the set of user nodes in M. Therefore, there must exist a sequence $(u_0, u_1, \ldots, u_i, \ldots)$ such that u_i is not in $U(M)$, i.e. u_i is not saturated in M. Thus, removing $(u_0, r_0), \ldots, (u_{i-1}, r_{i-1})$ and adding $(u_1, r_0), \ldots, (u_i, r_{i-1})$ results in a maximum matching in which u_0 is not saturated. □

Lemma 5. *Let $G = (U, R, E)$ be a transaction graph and $H = (U', R', E')$ be a closed component of $\mathsf{Core}(G)$. If there exists an edge $e = (u, r) \in E$ with $u \notin U'$ and $r \in R'$, then u belongs to another closed component of $\mathsf{Core}(G)$.*

Proof. Suppose $e = (u, r) \in E$ with $u \notin U'$ and $r \in R'$ exists, then u must belong to another component I of $\mathsf{Core}(G)$. Suppose that the claim is false, i.e. I is not closed. Note that I is a transaction graph where $I = \mathsf{Core}(I)$ since I is a component of $\mathsf{Core}(G)$. Applying Lemma 4, the graph I' obtained from I by

removing u and all edges in I connected to it is also a transaction graph, i.e. there exists a maximum matching M' in I'. Note that M' is also a maximum matching in I.

Consider now a maximum matching M in H. We observe that $M \cup M'$ is a maximum matching of the transaction graph $H \cup I$. Let $(u^\dagger, r)$ be the edge in M saturating r. By removing $(u^\dagger, r)$ from and adding $e = (u, r)$ to $M \cup M'$, we obtain another maximum matching M^* in $H \cup I$. However, this implies e is an edge in $\mathsf{Core}(G)$ connecting a node in H, contradicting that H is a component of $\mathsf{Core}(G)$. □

C Proofs for DAS Properties

We restate Theorems 4.1 to 4.4 and give their proof sketches.

Theorem 4.1. *If* TAG *is correct and key-binding and* ARG *is complete for* $\mathcal{R}$*, then* Ω *has integrity.*

Proof. (Sketch.) The derivation integrity is immediate from the correctness of TAG. For the action integrity, we notice from the $\mathsf{ActionIntegrity}_{\Omega,\mathcal{A}}$ experiment for any PPT algorithm $\mathcal{A}$ that, if the single_use condition is omitted, then the correctness of TAG and the completeness of ARG guarantee that the experiment returns 0 with certainty. If the single_use condition is taken into consideration, then the experiment returns 1 if $\mathcal{A}$ is able to produce two distinct secret keys $\mathsf{sk}, \mathsf{sk}'$ included in two distinct account secret keys which generate an identical tag of the tagging scheme TAG. This however violates that TAG.TagEval is key-binding. □

Theorem 4.2. *If* COM *is binding and* ARG *is knowledge-sound for* $\mathcal{R}$*, then* Ω *has authenticity.*

Proof. (Sketch.) The source and target binding properties are immediate given the binding property of COM.

To prove the knowledge soundness of Ω, we construct a knowledge extractor $\mathcal{E}_\mathcal{A}$ which runs the knowledge extractor of ARG for each action transcript output by an adversary $\mathcal{A}$. Consider the experiment $\mathsf{Authenticity}_{\Omega,\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda)$. The experiment outputs 1 if $\mathsf{valid\_txs} = 1$ but $\mathsf{valid\_actions} = 0$ or $\mathsf{single\_use} = 0$. From the knowledge-soundness of ARG, the probability of $\mathsf{valid\_txs} = 1$ but $\mathsf{valid\_actions} = 0$ is negligible. For the event of $\mathsf{valid\_txs} = 1$ but $\mathsf{single\_use} = 0$, this happens only if $\mathcal{E}_\mathcal{A}$ can produce two distinct tags, say $\mathsf{tag} = \mathsf{TAG.TagEval}(\mathsf{sk})$ and $\mathsf{tag}' = \mathsf{TAG.TagEval}(\mathsf{sk}')$, for a single account acc, which happens only if $\mathcal{E}_\mathcal{A}$ can produce two distinct tuples (ask, x) and (ask, x') where $\mathsf{SChk}(\mathsf{acc}, \mathsf{ask}, x) = 1$ and $\mathsf{SChk}(\mathsf{acc}, \mathsf{ask}', x') = 1$. By the source binding property, this occurs with negligible probability. □

Theorem 4.3. *If* TAG *is related-key pseudorandom,* COM *is hiding, and* ARG *is zero-knowledge for* $\mathcal{R}$*, then* Ω *is private.*

Proof. (Sketch.) We first consider a static-corruption variant of $\mathsf{Privacy}^b_{\Omega,\mathcal{A}}$ where adversary $\mathcal{A}$ declares in advance all queries to $\mathsf{Corrupt}\mathcal{O}$, and then argue about privacy against adaptive corruption by complexity leveraging. The privacy of Ω against static corruption could be proven straightforwardly.

First, we modify $\mathsf{Privacy}^b_{\Omega,\mathcal{A}}$ to obtain a pair of hybrids where the ARG proof in the challenge transcript is replaced by one simulated by the zero-knowledge simulator of ARG. This modification is unnoticeable due to ARG being zero-knowledge. Next, we define another pair of hybrids where tags of the honest users included in the challenge transcript are replaced by random ones. This is unnoticeable since TAG is related-key pseudorandom. For the final pair of hybrids, the only difference between these experiments is in the target accounts included in the challenge transcript. Since COM is hiding, this difference can be noticed with negligible probability. □

Theorem 4.4. *If* TAG *is one-way and related-key pseudorandom, and* ARG *is knowledge-sound and zero-knowledge for* $\mathcal{R}$*, then* Ω *has availability.*

Proof. (Sketch.) Similar to the proof of privacy, it is easier to first consider a variant of availability against static corruption, and then argue about availability against adaptive corruption by complexity leveraging. The availability of Ω against static corruption could be proven straightforwardly.

First, we modify $\mathsf{Availability}_{\Omega,\mathcal{A}}$ to obtain a hybrid where the ARG proof in the challenge transcript is replaced by one simulated by the zero-knowledge simulator of ARG. This modification is unnoticeable due to ARG being zero-knowledge. Next, we define another hybrid where tags of the honest users included

in the challenge transcript are replaced by random ones. This modification is unnoticeable due to TAG being related-key pseudorandom. Finally, we argue that the above hybrid outputs 1 only with negligible probability. Indeed, for this experiment to output 1, the adversary must produce a valid action transcript different from the challenge one such that they contain at least one common tag tag, which was sampled at random by the experiment. Since ARG is knowledge-sound, by running the knowledge extractor of ARG, we could obtain sk satisfying $\mathsf{tag} = \mathsf{TAG.TagEval}(\mathsf{sk})$. This, however, violates the one-wayness of TAG. $\square$