On Sustainable Ring-based Anonymous Systems

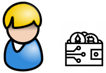
Sherman S. M. Chow, Christoph Egger, Russell W. F. Lai,
Viktoria Ronge, **Ivy K. Y. Woo**

Aalto University, Finland

13 July 2023 @ CSF, Dubrovnik

Ring-based Anonymous System

– The Example of Anonymous Cryptocurrencies

Alice 

Choose decoy accounts from system.

Action $=$ (I own one among ring which contains 1 coin, and I'm sending 1 coin to Bob's .)

 Bob

$\text{tx} \leftarrow \text{Prove}(\text{Action})$ $\xrightarrow{\text{tx}}$

 Blockchain/Everyone

Ring-based Anonymous System

– The Example of Anonymous Cryptocurrencies

Alice  

Choose decoy accounts from system.

Action $=$ (I own one among ring which contains 1 coin, and I'm sending 1 coin to Bob's .)

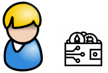
$\longrightarrow$  Bob

tx $\leftarrow$ Prove(Action) $\xrightarrow{\text{tx}}$  Blockchain/Everyone

- Single-use: Each can only be used once. (prevent double-spending)
  New balance = New

Ring-based Anonymous System

– The Example of Anonymous Cryptocurrencies

Alice 

Choose decoy accounts from system.

Action = (I own one among ring which contains 1 coin, and I'm sending Bob a new with 1 coin.)

 Bob

 Blockchain/Everyone

$\text{tx} \leftarrow \text{Prove}(\text{Action})$ $\xrightarrow{\text{tx}}$

- Single-use: Each can only be used once. (prevent double-spending)
  New balance = New

(Un-)Sustainability

- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping

(Un-)Sustainability

- Actions are anonymous $\Rightarrow$ Unknown which have been used
- Need to maintain the growing set of ... $\Rightarrow$ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity ?

(Un-)Sustainability

- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem

(Un-)Sustainability

- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity + availability ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem ✗

(Un-)Sustainability

- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity + availability ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem ✗

Our Simple Solution

Partition accounts in system into chunks of equal size k. [REL^{+}21]

Each account pick decoys from the same chunk.

k actions from a chunk ⇒ All k accounts in the chunk are used ⇒ Remove

(Un-)Sustainability

- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity + availability ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem ✗

Our Simple Solution

Partition accounts in system into chunks of equal size k. [REL+21]

Each account pick decoys from the same chunk.

k actions from a chunk ⇒ All k accounts in the chunk are used ⇒ Remove

(Un-)Sustainability

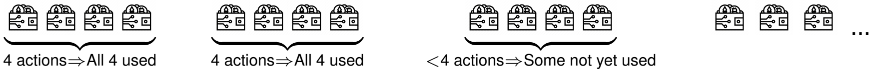
- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity + availability ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem ✗

Our Simple Solution

Partition accounts in system into chunks of equal size k. [REL$^+$21]

Each account pick decoys from the same chunk.

k actions from a chunk ⇒ All k accounts in the chunk are used ⇒ Remove



(Un-)Sustainability

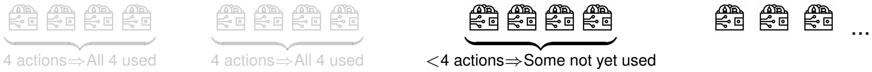
- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity + availability ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem ✗

Our Simple Solution

Partition accounts in system into chunks of equal size k. [REL+21]

Each account pick decoys from the same chunk.

k actions from a chunk ⇒ All k accounts in the chunk are used ⇒ Remove



(Un-)Sustainability

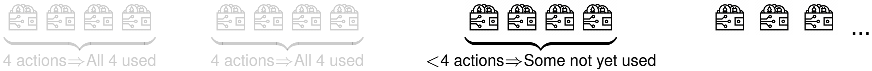
- Actions are anonymous ⇒ Unknown which have been used
- Need to maintain the growing set of ... ⇒ Unsustainable bookkeeping
- Question: How to achieve sustainable bookkeeping + anonymity + availability ?
  - Quisquis [FMMO19]: Sustainable + anonymous, but suffer from availability problem ✗

Our Simple Solution

Partition accounts in system into chunks of equal size k. [REL$^+$21]

Each account pick decoys from the same chunk.

k actions from a chunk ⇒ All k accounts in the chunk are used ⇒ Remove



- Simple counting: N unused accounts ⇒ at most kN accounts in system
- Our work: Formalise the above idea

Our Contributions

- General model for "Decentralised Anonymous Systems" (DAS)
  Captures e.g. anonymous cryptocurrencies, anonymous credentials
- Formal definition of "Sustainability" + other desirable properties of DAS
- Construction of DAS from cryptographic building blocks:
  Achieves various desirable properties incl. sustainability

Our Contributions

- General model for "Decentralised Anonymous Systems" (DAS)
  Captures e.g. anonymous cryptocurrencies, anonymous credentials
- Formal definition of "Sustainability" + other desirable properties of DAS
- Construction of DAS from cryptographic building blocks:
  Achieves various desirable properties incl. sustainability
- Efficient "garbage collector" algorithm:
  Detects all surely-used accounts in a ring-based anonymous system (first of its kind)
- Experiment: Mimic Monero's ring-sampling strategy, investigate its (un-)sustainability

Decentralised Anonymous Systems (DAS)

- (pp, state)
- Alice owns $(\mathsf{mpk}_A, \mathsf{msk}_A)$ and ask_i for her acc_i with attribute x_i
- Action: Alice wants to transfer 1 coin from her source acc_i to Bob's target acc_j

Decentralised Anonymous Systems (DAS)

mpk_A msk_A | acc_i ask_i, x_i

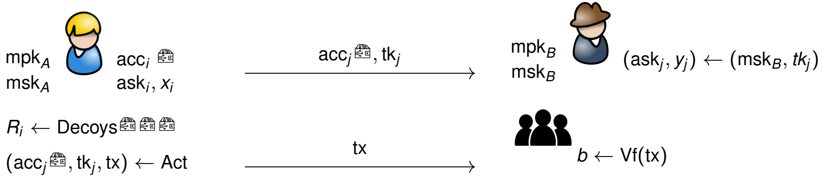
$R_i \leftarrow \mathsf{Decoys}$

$(\mathsf{acc}_j, \mathsf{tk}_j, \mathsf{tx}) \leftarrow \mathsf{Act}$

mpk_B msk_B

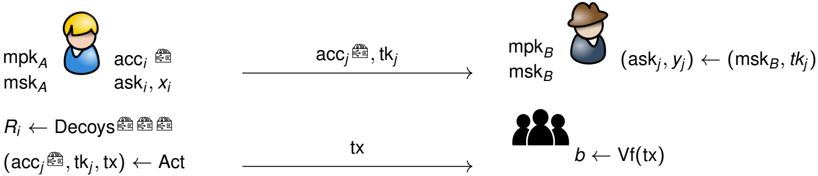
- $(\mathsf{pp}, \mathsf{state})$
- Alice owns $(\mathsf{mpk}_A, \mathsf{msk}_A)$ and ask_i for her acc_i with attribute x_i
- Action: Alice wants to transfer 1 coin from her source acc_i to Bob's target acc_j
  - Choose a ring R_i for her acc_i
  - Given her $(\mathsf{ask}_i, x_i = 1\text{ coin})$ and Bob's $(\mathsf{mpk}_B, y_j = 1\text{ coin})$, generate $(\mathsf{acc}_j, \mathsf{tk}_j, \mathsf{tx})$

Decentralised Anonymous Systems (DAS)



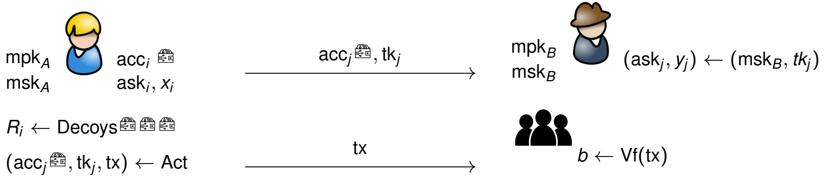
- $(\mathsf{pp}, \mathsf{state})$
- Alice owns $(\mathsf{mpk}_A, \mathsf{msk}_A)$ and ask_i for her acc_i with attribute x_i
- Action: Alice wants to transfer 1 coin from her source acc_i to Bob's target acc_j
  - Choose a ring R_i for her acc_i
  - Given her $(\mathsf{ask}_i, x_i = 1 \text{ coin})$ and Bob's $(\mathsf{mpk}_B, y_j = 1 \text{ coin})$, generate $(\mathsf{acc}_j, \mathsf{tk}_j, \mathsf{tx})$
    - For acc_j, Bob can use $(\mathsf{msk}_B, \mathsf{tk}_j)$ to derive (ask_j, y_j)
    - Everyone can verify Alice's action by transcript tx

Decentralised Anonymous Systems (DAS)



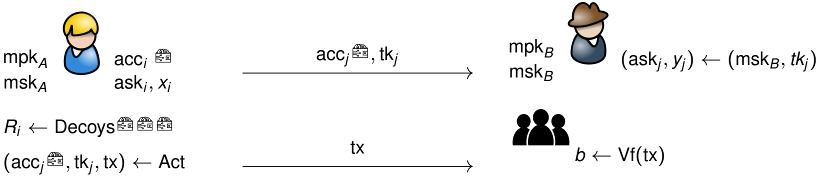
- $(\mathsf{pp}, \mathsf{state})$
- Alice owns $(\mathsf{mpk}_A, \mathsf{msk}_A)$ and ask_i for her acc_i with attribute x_i
- Action: Alice wants to transfer 1 coin from her source acc_i to Bob's target acc_j
  - Choose a ring R_i for her acc_i
  - Given her $(\mathsf{ask}_i, x_i = 1 \text{ coin})$ and Bob's $(\mathsf{mpk}_B, y_j = 1 \text{ coin})$, generate $(\mathsf{acc}_j, \mathsf{tk}_j, \mathsf{tx})$
    - For acc_j, Bob can use $(\mathsf{msk}_B, \mathsf{tk}_j)$ to derive (ask_j, y_j)
    - Everyone can verify Alice's action by transcript tx
- What to verify:
  - Predicates: Admissible actions $\mathcal{P}$, e.g. $x_i = y_j$; Admissible rings $\mathcal{Q}$, e.g. $|R_i| \geq 11$

Decentralised Anonymous Systems (DAS)



- $(\mathsf{pp}, \mathsf{state})$
- Alice owns $(\mathsf{mpk}_A, \mathsf{msk}_A)$ and ask_i for her acc_i with attribute x_i
- Action: Alice wants to transfer 1 coin from her source acc_i to Bob's target acc_j
  - Choose a ring R_i for her acc_i
  - Given her $(\mathsf{ask}_i, x_i = 1 \text{ coin})$ and Bob's $(\mathsf{mpk}_B, y_j = 1 \text{ coin})$, generate $(\mathsf{acc}_j, \mathsf{tk}_j, \mathsf{tx})$
    - For acc_j, Bob can use $(\mathsf{msk}_B, \mathsf{tk}_j)$ to derive (ask_j, y_j)
    - Everyone can verify Alice's action by transcript tx
- What to verify:
  - Predicates: Admissible actions $\mathcal{P}$, e.g. $x_i = y_j$; Admissible rings $\mathcal{Q}$, e.g. $|R_i| \geq 11$
  - ask_i, x_i are valid for source acc_i
  - Target $\mathsf{acc}_j, \mathsf{tk}_j$ are created for mpk_j, y_j

Decentralised Anonymous Systems (DAS)



- $(\mathsf{pp}, \mathsf{state}) \leftarrow \mathsf{Setup}$
- Alice owns $(\mathsf{mpk}_A, \mathsf{msk}_A) \leftarrow \mathsf{MKGen}$ and ask_i for her acc_i with attribute x_i
- Action: Alice wants to transfer 1 coin from her source acc_i to Bob's target acc_j
  - Choose a ring R_i for her acc_i
  - Given her $(\mathsf{ask}_i, x_i = 1 \text{ coin})$ and Bob's $(\mathsf{mpk}_B, y_j = 1 \text{ coin})$, generate $(\mathsf{acc}_j, \mathsf{tk}_j, \mathsf{tx}) \leftarrow \mathsf{Act}$
    - For acc_j, Bob can use $(\mathsf{msk}_B, \mathsf{tk}_j)$ to derive $(\mathsf{ask}_j, y_j) \leftarrow \mathsf{AKDer}$
    - Everyone can verify Alice's action by transcript tx $b \leftarrow \mathsf{Vf}$
- What to verify:
  - Predicates: Admissible actions $\mathcal{P}$, e.g. $x_i = y_j$; Admissible rings $\mathcal{Q}$, e.g. $|R_i| \geq 11$
  - ask_i, x_i are valid for source acc_i $b \leftarrow \mathsf{SChk}$
  - Target $\mathsf{acc}_j, \mathsf{tk}_j$ are created for mpk_j, y_j $b \leftarrow \mathsf{TChk}$

Overview of Properties

1. Integrity: “Correctness” of system for honest users (in presence of malicious users)
2. Authenticity: Alice can authenticate that her action is valid
3. Privacy
4. Availability
5. Sustainability

Overview of Properties

1. Integrity: "Correctness" of system for honest users (in presence of malicious users)
   - Action integrity: Alice's action is valid $\Rightarrow$ Vf $=1$
   - Derivation integrity: Passes TChk $\Rightarrow$ Bob can recover (ask_j, y_j) from AKDer
2. Authenticity: Alice can authenticate that her action is valid
3. Privacy
4. Availability
5. Sustainability

Overview of Properties

1. Integrity: "Correctness" of system for honest users (in presence of malicious users)
   - Action integrity: Alice's action is valid $\Rightarrow \mathsf{Vf} = 1$
   - Derivation integrity: Passes TChk $\Rightarrow$ Bob can recover (ask_j, y_j) from AKDer
2. Authenticity: Alice can authenticate that her action is valid
   - Source binding: Passes SChk $\Rightarrow$ acc_i 🔗 (ask_i, x_i)
   - Target binding: Passes TChk $\Rightarrow$ acc_j 🔗 y_j
   - Knowledge sound: $\mathsf{Vf} = 1 \Rightarrow$ Alice knows input to Act that gives tx
3. Privacy
4. Availability
5. Sustainability

Overview of Properties

1. Integrity: "Correctness" of system for honest users (in presence of malicious users)
   - Action integrity: Alice's action is valid $\Rightarrow \mathsf{Vf} = 1$
   - Derivation integrity: Passes TChk $\Rightarrow$ Bob can recover (ask_j, y_j) from AKDer
2. Authenticity: Alice can authenticate that her action is valid
   - Source binding: Passes SChk $\Rightarrow \mathsf{acc}_i$ 🔗 (ask_i, x_i)
   - Target binding: Passes TChk $\Rightarrow \mathsf{acc}_j$ 🔗 y_j
   - Knowledge sound: $\mathsf{Vf} = 1 \Rightarrow$ Alice knows input to Act that gives tx
3. Privacy: Alice's acc_i hidden within ring R_i given tx (source anonymity)
   Bob's mpk_B hidden from acc_j (target anonymity)
4. Availability
5. Sustainability

Overview of Properties

1. Integrity: "Correctness" of system for honest users (in presence of malicious users)
   - Action integrity: Alice's action is valid $\Rightarrow \mathsf{Vf} = 1$
   - Derivation integrity: Passes TChk $\Rightarrow$ Bob can recover (ask_j, y_j) from AKDer
2. Authenticity: Alice can authenticate that her action is valid
   - Source binding: Passes SChk $\Rightarrow$ acc_i 🔗 (ask_i, x_i)
   - Target binding: Passes TChk $\Rightarrow$ acc_j 🔗 y_j
   - Knowledge sound: $\mathsf{Vf} = 1 \Rightarrow$ Alice knows input to Act that gives tx
3. Privacy: Alice's acc_i hidden within ring R_i given tx (source anonymity)
   Bob's mpk_B hidden from acc_j (target anonymity)
4. Availability: Action is valid w.r.t. state at time $t \Rightarrow$ Valid w.r.t. state at anytime afterwards
   - Captures DOS attacks, e.g. that against Quisquis [FMMO19]
5. Sustainability

Overview of Properties

1. **Integrity**: "Correctness" of system for honest users (in presence of malicious users)
   - Action integrity: Alice's action is valid $\Rightarrow \mathsf{Vf} = 1$
   - Derivation integrity: Passes TChk $\Rightarrow$ Bob can recover (ask_j, y_j) from AKDer
2. **Authenticity**: Alice can authenticate that her action is valid
   - Source binding: Passes SChk $\Rightarrow$ acc_i 🔗 (ask_i, x_i)
   - Target binding: Passes TChk $\Rightarrow$ acc_j 🔗 y_j
   - Knowledge sound: $\mathsf{Vf} = 1 \Rightarrow$ Alice knows input to Act that gives tx
3. **Privacy**: Alice's acc_i hidden within ring R_i given tx (source anonymity)
   Bob's mpk_B hidden from acc_j (target anonymity)
4. **Availability**: Action is valid w.r.t. state at time $t \Rightarrow$ Valid w.r.t. state at anytime afterwards
   - Captures DOS attacks, e.g. that against Quisquis [FMMO19]
5. **Sustainability**: # potentially unused account $\approx$ # truly unused account

Closer Look at Sustainability

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

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

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

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

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

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

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

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

- A DAS is k-sustainable:
  $\exists \beta(\lambda) \in \mathsf{poly}(\lambda)$ s.t. for any PPT $\mathcal{A}$, $\Pr\left[\mathsf{Sustainability}_{\Omega,\mathcal{A},k,\beta}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$

Closer Look at Sustainability

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

$(\text{pp}, \text{state}_0) \leftarrow \text{Setup}(1^\lambda)$

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

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

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

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

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

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

- A DAS is k-sustainable:
  $\exists\beta(\lambda) \in \text{poly}(\lambda)$ s.t. for any PPT $\mathcal{A}$, $\Pr\left[\text{Sustainability}_{\Omega,\mathcal{A},k,\beta}(1^\lambda) = 1\right] \le \text{negl}(\lambda)$
- Sustainable state:
  $|\text{state}_t| \le k \cdot \beta(\lambda) \times$ # truly unused accounts,
  # potentially unused accounts = $|U_t| \le k \times$ # truly unused accounts

Closer Look at Sustainability

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

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

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

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

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

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

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

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

- A DAS is k-sustainable:
  $\exists \beta(\lambda) \in \mathsf{poly}(\lambda)$ s.t. for any PPT $\mathcal{A}$, $\Pr\left[\mathsf{Sustainability}_{\Omega,\mathcal{A},k,\beta}(1^\lambda) = 1\right] \le \mathsf{negl}(\lambda)$
- Sustainable state:
  $|\mathsf{state}_t| \le k \cdot \beta(\lambda) \times$ # truly unused accounts,
  # potentially unused accounts = $|U_t| \le k \times$ # truly unused accounts
- Next: Construct k-sustainable DAS for constant k

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR+19]

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR+19]
- TAG: Generate $(\mathsf{mpk}, \mathsf{msk})$ and tk_j, Derive ask_j from $(\mathsf{msk}_j, \mathsf{tk}_j)$
  MKGen, AKDer

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR+19]
- TAG: Generate (mpk, msk) and tk_j, Derive ask_j from $(\mathsf{msk}_j, \mathsf{tk}_j)$
  MKGen, AKDer
  COM: Commit x_i for acc_i, y_j for acc_j SChk, TChk
  ARG: Generate and verify transcript tx:
  Act, Vf
  - COM and TAG are correctly evaluated
  - Alice's action satisfies $\mathcal{P}$ and passes SChk, TChk

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR⁺19]
- TAG: Generate $(\mathsf{mpk}, \mathsf{msk})$ and tk_j, Derive ask_j from $(\mathsf{msk}_j, \mathsf{tk}_j)$
  MKGen, AKDer   Derivation Integrity
  COM: Commit x_i for acc_i, y_j for acc_j SChk, TChk   Source/Target binding
  ARG: Generate and verify transcript tx:
  Act, Vf   Action integrity, Knowledge sound, Privacy
  - COM and TAG are correctly evaluated
  - Alice's action satisfies $\mathcal{P}$ and passes SChk, TChk

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR$^+$19]
- TAG: Generate $(\mathsf{mpk}, \mathsf{msk})$ and tk_j, Derive ask_j from $(\mathsf{msk}_j, \mathsf{tk}_j)$, Generate tag_j ask_j
  MKGen, AKDer Derivation Integrity, Availability
  COM: Commit x_i for acc_i, y_j for acc_j SChk, TChk Source/Target binding
  ARG: Generate and verify transcript tx:
  Act, Vf Action integrity, Knowledge sound, Privacy
  - COM and TAG are correctly evaluated
  - Alice's action satisfies $\mathcal{P}$ and passes SChk, TChk

  *Vf also checks tag_j is used only once

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR+19]
- TAG: Generate (mpk, msk) and tk_j, Derive ask_j from $(\mathsf{msk}_j, \mathsf{tk}_j)$, Generate tag_j ask_j

  MKGen, AKDer   Derivation Integrity, Availability

  COM: Commit x_i for acc_i, y_j for acc_j SChk, TChk   Source/Target binding

  ARG: Generate and verify transcript tx:

  Act, Vf   Action integrity, Knowledge sound, Privacy

  - COM and TAG are correctly evaluated
  - Alice's action satisfies $\mathcal{P}$ and passes SChk, TChk

  *Vf also checks tag_i is used only once and R_i satisfies $\mathcal{Q}$
- Sustainability: Predicate $\mathcal{Q}$ = partitioning sampler [REL+21]

Building Blocks of Construction

- Cryptographic building blocks:
  Commitment COM, NIZK argument system ARG, Tagging scheme TAG [LRR+19]
- TAG: Generate (mpk, msk) and tk_j, Derive ask_j from $(\mathsf{msk}_j, \mathsf{tk}_j)$, Generate tag_j 🔗 ask_j
  MKGen, AKDer Derivation Integrity, Availability
  COM: Commit x_i for acc_i, y_j for acc_j SChk, TChk Source/Target binding
  ARG: Generate and verify transcript tx:
  Act, Vf Action integrity, Knowledge sound, Privacy
  - COM and TAG are correctly evaluated
  - Alice's action satisfies $\mathcal{P}$ and passes SChk, TChk

  *Vf also checks tag_i is used only once and R_i satisfies $\mathcal{Q}$
- Sustainability: Predicate $\mathcal{Q}$ = partitioning sampler [REL+21]

k-Partitioning Sampler

k accounts spawned at adjacent time forms a "chunk"
To use an account: Choose ring members from the same chunk

Sustainability via Partitioning Sampler

k-Partitioning Sampler

k accounts spawned at adjacent time forms a “chunk”

To use an account: Choose ring members from the same chunk

Sustainability via Partitioning Sampler

k-Partitioning Sampler

k accounts spawned at adjacent time forms a “chunk”
To use an account: Choose ring members from the same chunk

- k accounts in a chunk are used $\Leftrightarrow$ All accounts in the chunk are used
  $\Rightarrow$ Remove whole chunk
- Simple “Garbage Collector”: Check if there are k valid actions in a chunk
- N truly unused accounts $\Rightarrow$ at most N chunks $=$ kN accounts stored in system
  $\Rightarrow$ k-sustainable

Sustainability via Partitioning Sampler

k-Partitioning Sampler

k accounts spawned at adjacent time forms a "chunk"
To use an account: Choose ring members from the same chunk

- k accounts in a chunk are used $\Leftrightarrow$ All accounts in the chunk are used
  $\Rightarrow$ Remove whole chunk
- Simple "Garbage Collector": Check if there are k valid actions in a chunk
- N truly unused accounts $\Rightarrow$ at most N chunks $=$ kN accounts stored in system
  $\Rightarrow$ k-sustainable
- Not need to know which of the k accounts an action is made from

Sustainability via Partitioning Sampler

k-Partitioning Sampler

k accounts spawned at adjacent time forms a "chunk"
To use an account: Choose ring members from the same chunk

- k accounts in a chunk are used $\Leftrightarrow$ All accounts in the chunk are used
  $\Rightarrow$ Remove whole chunk
- Simple "Garbage Collector": Check if there are k valid actions in a chunk
- N truly unused accounts $\Rightarrow$ at most N chunks $= kN$ accounts stored in system
  $\Rightarrow$ k-sustainable
- Not need to know which of the k accounts an action is made from
- Sustainability vs. Privacy:
  s-sustainable $\Rightarrow s \geq$ minimum ring size ever used (counting argument)
  Partitioning: Fix ring size = chunk size $k \Rightarrow s \geq k$; We achieve $s = k$

Construction

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

$\mathsf{pp}_{\mathsf{TAG}} \leftarrow \mathsf{TAG.Setup}(1^\lambda)$
$\mathsf{pp}_{\mathsf{COM}} \leftarrow \mathsf{COM.Setup}(1^\lambda)$
$\mathsf{pp}_{\mathsf{ARG}} \leftarrow \mathsf{ARG.Setup}(1^\lambda)$
$\mathsf{pp} := (\mathsf{pp}_{\mathsf{TAG}}, \mathsf{pp}_{\mathsf{COM}}, \mathsf{pp}_{\mathsf{ARG}})$
return $(\mathsf{pp}, \mathsf{st} := \emptyset)$

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

$\mathsf{msk} \leftarrow \mathsf{TAG.SKGen}(\mathsf{pp}_{\mathsf{TAG}})$
$\mathsf{mpk} \leftarrow \mathsf{TAG.PKGen}(\mathsf{msk})$
return $(\mathsf{mpk}, \mathsf{msk})$

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

parse tk **as** (δ, x, r)
$\mathsf{sk} \leftarrow \mathsf{TAG.SKDer}(\mathsf{msk}, \delta)$
$\mathsf{ask} := (\mathsf{sk}, r)$
return (ask, x)

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

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**
  $(\overline{\mathsf{pk}}_i, \delta_i) \leftarrow \mathsf{TAG.PKDer}(\mathsf{mpk}_i)$
  $\overline{\mathsf{com}}_i \leftarrow \mathsf{COM.Com}(y_i; s_i)$ // with uniform randomness s_i
  $\overline{\mathsf{acc}}_i := (\overline{\mathsf{pk}}_i, \overline{\mathsf{com}}_i)$; $\mathsf{tk}_i := (\delta_i, y_i, s_i)$
$R := \bigcup_{i \in [m]} R_i$
$\mathsf{stmt} := \left(p_{m,n}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in R}, \{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \langle \overline{\mathsf{acc}}_i \rangle_{i \in [n]}\right)$
$\mathsf{wit} := \left((\mathsf{aid}_i, \mathsf{sk}_i, r_i, x_i)_{i \in [m]}, (\mathsf{mpk}_i, \delta_i, y_i, s_i)_{i \in [n]}, \mathsf{aux}\right)$
$\pi \leftarrow \mathsf{ARG.Prove}(\mathsf{stmt}, \mathsf{wit})$
$\mathsf{tx} := \left(\{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \pi\right)$
return $\left(\mathsf{tx}, \langle \overline{\mathsf{acc}}_i, \mathsf{tk}_i \rangle_{i \in [n]}\right)$

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

parse st **as** $\langle \mathsf{Tag}_C \rangle_{C \in \mathsf{Partition}(U)}$
parse tx **as** $\left(\{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \pi\right)$
$R := \bigcup_{i \in [m]} R_i$
$\mathsf{stmt} := \left(\begin{matrix} p_{m,n}, \langle \mathsf{acc}_{\mathsf{aid}} \rangle_{\mathsf{aid} \in R}, \\ \{(R_i, \mathsf{tag}_i)\}_{i \in [m]}, \langle \overline{\mathsf{acc}}_i \rangle_{i \in [n]} \end{matrix}\right)$
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 $(0, \mathsf{st})$

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

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

parse acc **as** $(\mathsf{pk}, \mathsf{com})$
parse ask **as** (sk, r)
return $\begin{cases} \mathsf{pk} = \mathsf{TAG.PKGen}(\mathsf{sk}) \\ \mathsf{com} = \mathsf{COM.Com}(x; r) \end{cases}$

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

parse acc **as** $(\mathsf{pk}, \mathsf{com})$
parse tk **as** (δ, x, r)
return $\begin{cases} x = y \\ \mathsf{com} = \mathsf{COM.Com}(x; r) \end{cases}$

Fig. 7. Generic construction of a Decentralised Anonymous System

Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

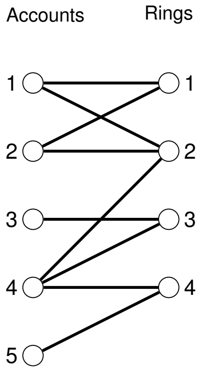
Below: Another GC for general ring-sampling mechanisms

Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

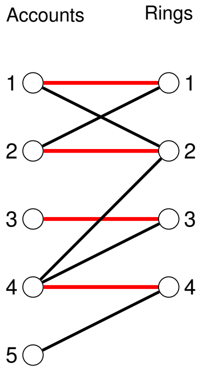
- Model ring-memberships as bipartite graph G [ELR$^+$22]



Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler
Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes = A possible accounts-rings assignment

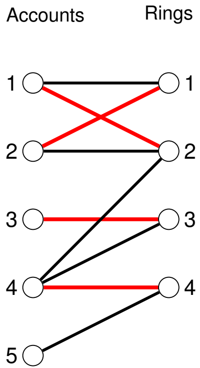


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes
  = A possible accounts-rings assignment

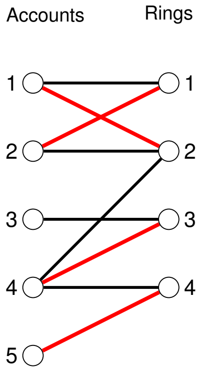


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes
  = A possible accounts-rings assignment

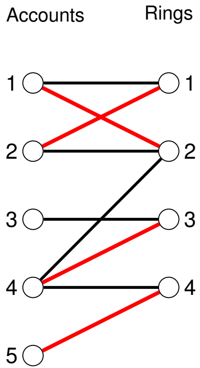


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes = A possible accounts-rings assignment
- Core(G) [DM58]: Subgraph of G s.t. edges in Core(G) = Union of all maximum matchings

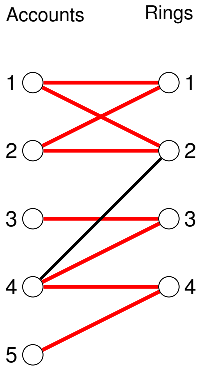


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes
  = A possible accounts-rings assignment
- Core(G) [DM58]: Subgraph of G s.t.
  edges in Core(G) = Union of all maximum matchings

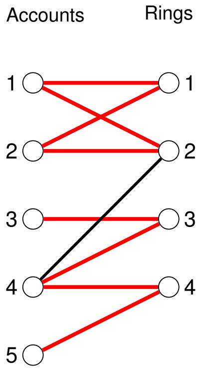


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes
  = A possible accounts-rings assignment
- Core(G) [DM58]: Subgraph of G s.t.
  edges in Core(G) = Union of all maximum matchings
- Our general GC:
  - Given ring-memberships, compute G and Core(G) [Tas12]
  - For each component in Core(G),
    check if num. account nodes = num. ring nodes
  - If yes, the accounts are used $\Rightarrow$ Can remove

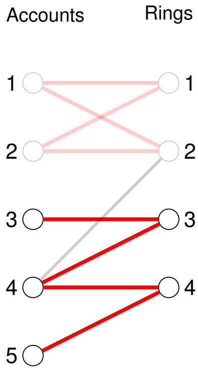


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes = A possible accounts-rings assignment
- Core(G) [DM58]: Subgraph of G s.t. edges in Core(G) = Union of all maximum matchings
- Our general GC:
  - Given ring-memberships, compute G and Core(G) [Tas12]
  - For each component in Core(G), check if num. account nodes = num. ring nodes
  - If yes, the accounts are used $\Rightarrow$ Can remove

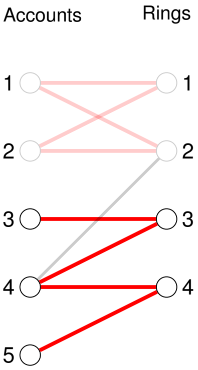


Garbage Collector for General Ring-sampling Mechanisms

Just now: GC that detects surely-used accounts if use partitioning sampler

Below: Another GC for general ring-sampling mechanisms

- Model ring-memberships as bipartite graph G [ELR$^+$22]
- A maximum matching = Matching covering all ring nodes
  = A possible accounts-rings assignment
- Core(G) [DM58]: Subgraph of G s.t.
  edges in Core(G) = Union of all maximum matchings
- Our general GC:
  - Given ring-memberships, compute G and Core(G) [Tas12]
  - For each component in Core(G),
    check if num. account nodes = num. ring nodes
  - If yes, the accounts are used $\Rightarrow$ Can remove



- We prove correctness and optimality of our GC,
  i.e. it returns all surely-used accounts (inferable from ring-memberships) and only them

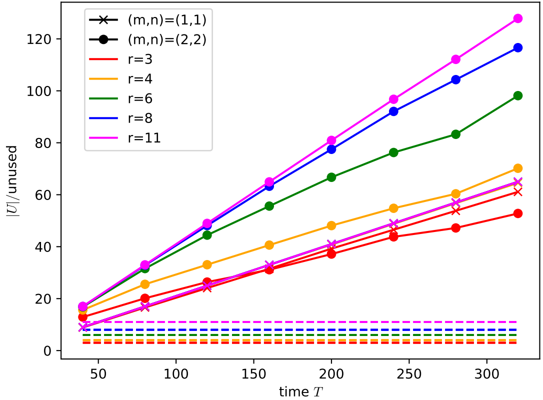
Experiment on Ring-Sampling Method of Monero

- Simulate ring-sampling mechanism used by Monero [REL+21]
- Given the simulated ring memberships, run our general GC, observe how many used accounts are detected

Experiment on Ring-Sampling Method of Monero

- Simulate ring-sampling mechanism used by Monero [REL$^+$21]
- Given the simulated ring memberships, run our general GC, observe how many used accounts are detected

If k-sustainable: ratio $\leq k$ $\longrightarrow$



$\longleftarrow$ Upper bound if use partitioning with chunk size $= r$: r-sustainable

r = ring size, (m, n) = (# used acc., # spawned acc.) in each transaction

Summary

- Model for Decentralised Anonymous Systems (DAS)
- Formal definition of “Sustainability” and other desirable properties of DAS
- First construction of DAS achieving sustainability, privacy and availability (and more)
  - Simple solution to sustainability: Partitioning sampler
- Efficient “garbage collector” for general ring-sampling mechanisms
- Experiment on Monero’s (un-)sustainability

Ivy K. Y. Woo
Aalto University, Finland
ivy.woo@aalto.fi
ivyw.ooo

Thank You!

ia.cr/2023/743

References

- A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs.
  Canadian Journal of Mathematics, 10:517–534, 1958
- Christoph Egger, Russell W. F. Lai, Viktoria Ronge, Ivy K. Y. Woo, and Hoover H. F. Yin. On defeating graph analysis of anonymous transactions.
  Proceedings on Privacy Enhancing Technologies, 3:538–557, 2022
- Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies.
  Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I 25, pages 649–678, 2019
- Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. Omniring: Scaling private payments without trusted setup.
  Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 31–48, 2019
- Viktoria Ronge, Christoph Egger, Russell W. F. Lai, Dominique Schröder, and Hoover H. F. Yin. Foundations of ring sampling.
  Proceedings on Privacy Enhancing Technologies, 3:265–288, 2021
- Tamir Tassa. Finding all maximally-matchable edges in a bipartite graph.
  Theoretical Computer Science, 423:50–58, 2012