When the Recursive Diversity Anonymity Meets the Ring Signature

Wangze Ni [†], Peng Cheng [*], Lei Chen [†], Xuemin Lin [#]
[†] Hong Kong University of Science and Technology, Hong Kong SAR, China
{wniab, leichen}@cse.ust.hk
[*]East China Normal University, Shanghai, China
pcheng@sei.ecnu.edu.cn
[#]The University of New South Wales, Australia
lxue@cse.unsw.edu.au

ABSTRACT

In privacy-preserving blockchain systems, to protect a sender's identity of a transaction in privacy-preserving blockchain systems, ring signature (RS) schemes have been widely implemented, which allow users to obscure consumed tokens via including "mixin" (i.e., chaff tokens). However, recent works point out that existing RS schemes are vulnerable to the "chain-reaction" analysis, where adversaries eliminate mixins of RSs by utilizing the fact that each token can only be consumed in a RS. By "chain-reaction" analysis, adversaries can find some definite token-RS pair sets (DTRSs) to confirm the sender's identity of a RS. Besides, the existing RS schemes do not consider the diversity of mixins when generating a RS. Moreover, since the transaction fee is proportional to the number of mixins, a use is motivated to use a RS with the minimum number of mixins. In this paper, we formally define the diversity-aware mixins selection (DA-MS) problem, which aims to generate a RS with the minimum number of mixins satisfying the constraints of its diversity and the anonymity of other RSs. We prove the DA-MS problem is #*P* and propose a breadth-first search algorithm to get the optimal solution. Furthermore, to efficiently solve the DA-MS problem, we propose two practical configurations and two approximation algorithms with theoretic guarantees. Through comprehensive experiments on real data sets as well as synthetic data sets, we illustrate the effectiveness and the efficiency of our solutions.

CCS CONCEPTS

• **Security and privacy → Privacy protections**.

KEYWORDS

Blockchain, Ring Signature, Privacy

ACM Reference Format:
Wangze Ni [†], Peng Cheng [*], Lei Chen [†], Xuemin Lin [#]. 2021. When the Recursive Diversity Anonymity Meets the Ring Signature. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, Xi'an, Shaanxi, China, 13 pages. https://doi.org/10.1145/3448016.3452825

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3452825

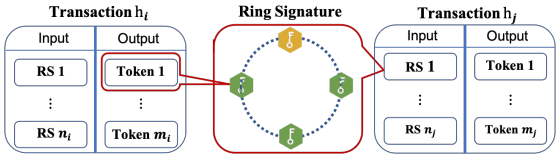


Figure 1: The Unspent Transaction Output (UTXO) model.

1 INTRODUCTION

Recently, blockchain technologies attract much attention from both academia and industries. From the perspective of the database community, blockchain is a kind of the distributed transaction management solution, where nodes keep replicas data of publicly agreed execution order of the transactions [1]. Many blockchain applications, such as cryptocurrency [2], e-voting [3], healthcare [4], and storage sharing [5], are in the Unspent Transaction Output (UTXO) model, where each transaction will consume some input token(s) from its sender(s) to its receiver(s). Thus, one token can only be consumed once. In blockchain systems, tokens represent programmable assets or access rights, which are operable only by the persons who have their private keys. For example, in e-voting systems [3], a token indicates a ballot of a user.

Since the transaction data is available for all the nodes in the blockchain system, the privacy issue (e.g., the ownership of tokens) needs to be handled for specific applications (e.g., e-voting or healthcare systems). By analyzing transactions on blockchains, adversaries can link users to transactions/tokens [6–8]. For instance, if a transaction h_2 consumes a token which is generated in a historical transaction (HT) h_1, adversaries can link the receiver of h_1 to the sender of h_2. Although current blockchain systems conceal users' identities by pseudonymities, this kind of link is still risk. Once adversaries link one pseudonymity with a user's identity, they can link the user's identity to all pseudonymities that are linked to the revealed pseudonymity [9]. Furthermore, they can infer who are trade partners with this user.

To protect privacy, ring signature (RS) schemes are widely used in various blockchain systems, where users can utilize RSs to obscure the ownership of input tokens by adding chaff tokens, namely "*mixins*". In some scenarios, users use RSs to hide the consumed

tokens in transactions to avoid adversaries linking their identities to transactions/tokens. As illustrated in Figure 1, there are multiple input RSs and output tokens in a transaction. A RS contains only one consumed token marked in yellow color and many other tokens (mixins) marked in green color. The details of RS schemes will be introduced in Section 2. Currently, existing RS schemes measure the anonymity of a RS as its number of mixins [10, 11]. However, there are two defects of these existing methods. The first is that, if in a RS the consumed token and its mixins are generated by the same transaction h_i, the source of the consumed token can still be inferred as h_i, which is called *Homogeneity Attack* [12]. The second is that, since each token can only be consumed once, adversaries can infer the consumed token in a RS by analyzing other related RSs, which is called *Chain-reaction Analysis* [10]. Besides, since the transaction fee is proportional to the number of mixins, a user is motivated to find an anti-attack RS with the smallest size.

To overcome the weakness of the existing solutions, in this paper, we will consider a RS generation problem for blockchain systems, namely diversity-aware mixins selection (DA-MS), which selects a minimal set of mixins for a given token to generate a valid anti-attach RS. Here is a motivation example.

Example 1 (An Mixins Selection Example). *Suppose there are four tokens (t_1, t_2, t_3, t_4) and two RSs, $r_1 = \{t_1, t_2\}$ and $r_2 = \{t_1, t_2\}$. The tokens t_1 and t_3 are the outputs of the same HT h_1. The tokens t_2 and t_4 are outputs of another two HTs h_2 and h_3, respectively. Now we want to generate a RS for consuming t_3.*

The first solution is using $r_3 = \{t_1, t_3\}$. Since t_1 and t_3 are all generated in h_1, adversaries need not to determine the consumed token in r_3 and can directly know the consumed token of r_3 is from h_1, which does not resist the homogeneity attack.

The second solution is using $r_3 = \{t_2, t_3\}$. Recall that each token can only be consumed once. Since $r_1 = r_2 = \{t_1, t_2\}$, adversaries can conclude that t_1 and t_2 must have been consumed. Thus, the consumed token in $r_3 = \{t_2, t_3\}$ must be t_3, which means chain-reaction analysis is not resisted in this solution.

The third solution is using $r_3 = \{t_1, t_2, t_3, t_4\}$. The consumed tokens of r_1, r_2, and r_3 cannot be inferred. However, r_3's size is 4.

A good solution is using $r_3 = \{t_3, t_4\}$. The consumed tokens in r_1, r_2, and r_3 cannot be inferred. Besides, this r_3 contains only 2 tokens.

Motivated by the above example, we will formalize the DA-MS problem, which targets on selecting a minimal set of mixins for a given token to consist a RS such that the new RS and existing RSs can resist attacks (e.g., homogeneity attacks and chain-reaction analysis attacks). In this paper, we first define a concept called "definite token-RS pair set" (DTRS) to resist the attack based on the chain-reaction analysis. Besides, inspired by the recursive diversity principle [13], we define the concept of a "recursive diversity RS" to measure the anonymity of a RS. Then, based on these concepts, we formally define the DA-MS problem as to find a recursive diversity RS with minimal size while retaining the existing RSs' privacy. We prove the DA-MS problem is #P through a reduction from the problem of enumerating all perfect matchings in bipartite graphs [14]. We first propose an exact breadth-first search algorithm. However, the time complexity is exponential high. To efficiently tackle DA-MS, three challenges need to be addressed: (a) how to efficiently verify if DTRSs of a RS satisfy the diversity requirement; (b) how to guarantee that a user can always consume a token with a RS that retains existing RSs' diversities since the blockchain system is distributed; and (c) how to pick an eligible set of mixins with a minimal size. To overcome the first two challenge, we propose two practical configurations which require that every new RS must be the superset of an existing RS or be disjoint with it; and the diversity requirement of a new RS should be a little higher than the diversity requirement of its DTRSs. To tackle the third challenge, we propose two approximation algorithms, namely the Progressive Algorithm and the Game-theoretic Algorithm, with theoretic guarantees.

Specifically, we make the following contributions:

- We define the threat model and the notion of a recursive (ϵ, l)-diversity RS in Section 2.
- We formally define the diversity-aware mixins selection (DA-MS) problem and give the proof of its hardness in Section 3.
- We propose a breadth-first search algorithm to achieve the exact solution of DA-MS in Section 5.
- We propose two practical configurations and two approximation algorithms, the Progressive Algorithm and the Game-theoretic Algorithm, with theoretic guarantees in Section 6.
- We conduct comprehensive experiments on real and synthetic data sets to demonstrate the effectiveness and efficiency of our solutions in Section 7.

Besides, we propose a framework, TokenMagic, in Section 4, discuss the related work in Section 8, and conclude in Section 9.

2 PRIVACY SEMANTICS

In blockchain systems, since RSs can be verified by all users (e.g., miners and transaction users), adversaries can access all the data of historical RSs. Thus, different from many privacy scenarios where users preserve privacy through bringing in noises [6–8], users must guarantee that RSs can be accurately executed in blockchain systems. Since users in blockchain systems are usually anonymous to some extent (i.e., using different unknown public keys), adversaries want to link some transactions on blockchains together and infer more information about transactions [9] (e.g., the addresses of partners of trades). Moreover, blockchain systems are distributed. Each user customizes RSs for her/his own benefits (i.e., minimizing the sizes of her/his own RSs) and privacy requirements. In this section, we introduce the threat model in blockchain scenarios and define the notion of a recursive (c, ℓ)-diversity RS.

2.1 The Ring Signature Scheme

Generally, a RS scheme can be divided into 3 parts: selecting a set of mixins, generating a ring signature, and verifying the signature:

- Step 1. Selecting a set of mixins M. For example, in Monero [2], a user inputs an integer ζ (larger than 10) to algorithm *SM*, then *SM* retrieves half of ζ mixins from the blocks generated within recent 1.8 days and other mixins from other blocks.
- Step 2. Generating a ring signature. With transaction message mg, a set of mixins M, the consuming token t and its private key pk, the algorithm *Gen* outputs a RS r, which contains the set of mixins M, a token image I, and a set of auxiliary parameters ω.
- Step 3. Verifying the signature. When other users receive r, they verify if r is eligible. If I had already been used (indicating the corresponding token having been consumed), r will be rejected. With ω, the algorithm *Ver* can verify if the user has the private key of t. Besides, verifiers can check if r satisfies some extra

configurations (e.g., Monero requires half of mixins must come from blocks generated in recent 1.8 days). If r conflicts these configurations, r will also be rejected.

Step 1 and Step 2 are conducted offline by users themselves. Step 3 is verified by miners when they block transactions. Thus, only the time complexity of Step 3 has an impact on the throughput of the blockchain system. A RS is a combination of a sorted sequence of public keys of tokens ($M \cup t$), a token image I of t, and a zero-knowledge proof ω. For a token t, its owner can use its private key pk to generate one token image I, which is explicit to all users. Observers cannot infer t from I. For a token, its image is unique. When an image I was used, we know the corresponding token was used and cannot be used again (i.e., prevent the double-spending attack [15]), although we do not know which token it is. The zero-knowledge proof [16] is used to prove that the user knows the index of the corresponding token of the image in the sorted sequence, while the proof will not leak any information about the token or its index. A user can use a RS to prove she/he has the right to use an unconsumed token and do not let others know what the token is. In this paper, we focus on how to select a desired set of mixins in Step 1 to enhance anonymity. This paper does not involve any change of the encryption algorithm in Step 2 and the verification algorithm in Step 3. In the rest of this paper, we simply consider a RS as a set of tokens consisting of a consuming token and its mixins.

2.2 The related RS set of a RS

Definition 1 (The Related RS Set of a RS)**.** For a RS r_k, at timestamp π, the related RS set of r_k is $R_\pi^{r_k} = \bigcup_{i=0} R_\pi^{r_k,i}$ where $R_\pi^{r_k,i}$ is the set of RSs which are proposed before timestamp π and contain some tokens in common with some RS(s) in $R_\pi^{r_k,i-1}$.

When $i = 0$, $R_\pi^{r_k,0}$ is the set of RSs containing any tokens in r_k.

Example 2. *At time stamp π, there are four RSs, $r_1 = \{t_1, t_2, t_5\}$, $r_2 = \{t_1, t_3\}$, $r_3 = \{t_1, t_3\}$, $r_4 = \{t_2, t_4\}$, and $r_5 = \{t_4, t_5, t_6\}$. The tokens t_5 and t_6 are from the same historical transaction h_1.*

By Definition 1, in Example 2, $R_\pi^{r_4} = \{r_1, r_2, r_3, r_5\}$. Specifically, $R_\pi^{r_4,0} = \{r_1, r_5\}$ and $R_\pi^{r_4,1} = \{r_2, r_3\}$. Thus, in some cases, the related RS set would be very large. In Section 4, we propose a framework to limit the size of the related RS set for a given RS.

2.3 The definite token-RS pair set (DTRS)

Definition 2 (The Definite Token-RS Pair Set)**.** At timestamp π, given a RS r_k, a *definite token-RS pair set* (DTRS) of r_k is a minimum set of token-RS pairs $d^{\pi,k} = \{p_1, p_2, \cdots, p_n\}$ that can determine the HT of the consumed token of r_k, where $p_i = \langle t_i, r_i \rangle$ is a token-RS pair indicating the token t_i is consumed in the RS r_i.

Some researchers have proposed methods (e.g., chain-reaction analysis) to cluster users through their historical transactions on the blockchain [17–19]. In chain-reaction analysis, the main purpose of adversaries is to infer the link between a RS and an entity [19]. For a given RS r_k in timestamp π, if the token-pairs of its DTRS $d^{\pi,k}$ are revealed to adversaries, they can exactly determine the consumed token in r_k. In Example 2, $\{\langle t_2, r_1 \rangle\}$ is a DTRS of r_5. Because when t_2 is consumed in r_1, the consumed token in r_4 must be t_4. Thus, the consumed token in r_5 must be t_5 or t_6, who are from HT h_1.

2.4 The Threat Model

Since blockchain systems are distributed, users can use different random strategies. Besides, since a token can only be consumed once, the procedure of generating a RS for a token happens only once. Thus, adversaries cannot guess the probability distribution of consumed tokens. However, because all data of RSs are recorded on the blockchain and a token can only be consumed in a RS, adversaries can guess the consumed token for a RS by analyzing the transactions on the blockchain (i.e., chain-reaction analysis) [10].

As common adversary assumptions [20], we assume adversaries know the algorithms used in blockchain systems, except for random procedures of algorithms. Thus, adversaries cannot exactly infer the real consumed token for a given RS without using any other information. Besides, we assume adversaries have some side information, i.e., they may know some token-RS pairs are revealed.

Definition 3 (The side information of adversaries)**.** Before proposing a new RS r_i, the side information of adversaries is a set of revealed token-RS pairs, $SI_i = \{p_1, p_2, \cdots, p_n\}$. A pair $p_j = \langle t_j, r_j \rangle \in SI_i$ means that adversaries know the consumed token of r_j is t_j.

The side information of adversaries can be partitioned into two disjoint parts $SI_i^\#$ and SI_i^* (i.e., $SI_i = SI_i^\# \cup SI_i^*$ and $SI_i^\# \cap SI_i^* = \emptyset$), where $SI_i^\#$ is the set of token-RS pairs that adversaries know directly, and SI_i^* is the set of token-RS pairs that adversaries infer from $SI_i^\#$. For example, adversaries may be the generator of some RSs, then their token-RS pairs $SI_i^\#$ are known to themselves [11].

Adversaries have two methods to get $SI_i^\#$. Firstly, they can use side information to eliminate tokens in the target RS and infer possible token-RS pairs by the frequency of HTs of remaining tokens (similar to the homogeneity attacks [13]). In Example 1, if adversaries know t_2 and t_4 are not consumed in $r_3 = \{t_1, t_2, t_3, t_4\}$, they can infer that the generator of r_3 is the receiver of the HT h_1 [9], since t_1 and t_3 are the outputs of h_1. The second method is to eliminate wrong DTRSs. In Example 2, there are three DTRSs of r_4, $d_1^{\pi,4} = \{\langle t_4, r_5 \rangle\}$, $d_2^{\pi,4} = \{\langle t_5, r_5 \rangle\}$, and $d_3^{\pi,4} = \{\langle t_2, r_1 \rangle\}$. If adversaries know t_5 is consumed in r_5, they can eliminate $d_1^{\pi,4}$ and $d_3^{\pi,4}$, and conclude that t_4 is the consumed token of r_4. *Thus, the more tokens of a RS and its possible DTRSs are from different HTs, the better anonymity of a RS would be.* This provides a natural motivation to apply the recursive diversity technique to enhance anonymity. However, if the side information of adversaries is strong enough (e.g., they know all consumed tokens of existing RSs), we cannot protect users' privacy. In Theorem 6.2, we prove that if the cardinality of SI is smaller than a threshold, with TokenMagic framework, adversaries cannot confirm the consumed tokens.

2.5 A Recursive (c, ℓ)-Diversity RS

We borrow the principle of recursive (c, ℓ)-diversity [13] to define the notion of a recursive (c, ℓ)-diversity RS. The recursive (c, ℓ)-diversity requires that $q_1 < c \cdot (q_\ell + \cdots + q_\theta)$, where q_i is the number of times that the i^th most-frequent sensitive value appears in the data set, and θ is the number of possible sensitive values in the data set. In RS scenarios, the sensitive value is the HT of a token. Parameter c and ℓ are set by users according to their requirements.

Definition 4 (A Recursive (c, ℓ)-Diversity RS)**.** At timestamp π, a RS r_k is a recursive (c, ℓ)-diversity RS, if (1) the HTs of tokens in r_k satisfy the recursive (c, ℓ)-diversity; and (2) the HTs of tokens in any DTRS of r_k satisfy the recursive (c, ℓ)-diversity.

Table 1: Symbols and Descriptions.

Symbol	Description
T	The universe of tokens t_i
h_i	The HT that outputs the token t_i
$R_\pi^{r_k}$	The related RS set of a RS r_k at the time stamp π
(c_k, ℓ_k)	The diversity requirement of a RS r_k
p_k	A token-RS pair $\langle t_k, r_k \rangle$ that means tp_k is consumed in rp_k
$d^{\pi,k}$	A DTRS of a RS r_k at the time stamp π

Suppose at timestamp π, there are three RSs, $r_1 = \{t_1, t_2\}$, $r_2 = \{t_2, t_3\}$, and $r_3 = \{t_1, t_3, t_4\}$. The tokens t_1 and t_3 are from the same HT h_1, and t_4 is from h_2. Thus, the HTs of tokens in r_3 are h_1, h_1 and h_2, respectively. Thus, for the HTs of tokens in r_3, $q_1 = 2$ and $q_2 = 1$. For r_3, there is only one DTRS, $d^{\pi,3} = \{\langle t_1, r_1\rangle, \langle t_3, r_2\rangle\}$. If a user requires a recursive (2, 1)-diversity RS, r_3 is eligible for both two conditions, i.e., $2 < 2 \cdot (2 + 1)$ and $2 < 2 \cdot 2$. If a user requires a recursive (3, 2)-diversity RS, r_3 satisfies the first condition, i.e., $2 < 3 \cdot 1$, but violates the second condition, i.e., $2 \geq 3 \cdot 0$.

3 PROBLEM DEFINITION

In this section, we formally formulate the Diversity-aware mixins selection (DA-MS) problem and prove that DA-MS is #P.

3.1 Diversity-aware Mixins Selection Problem

After a RS is blocked on the blockchain, its DTRSs and its anonymity may still be changed. For example, in Example 2, if a new RS $r_6 = \{t_2, t_4\}$ is proposed, adversaries can infer that the consumed token of r_1 is t_5 and the consumed token of r_5 is t_6. To maintain privacy, a user can claim the anonymity requirement when committing a RS to the blockchain. When a new RS is proposed, it should guarantee that the DTRSs of each existing RS still satisfy its claimed anonymity requirement. We define the DA-MS problem as follows:

Definition 5 (The Diversity-aware Mixins Selection (DA-MS) Problem)**.** Given a mixin universe T, a token t_τ that will be consumed at timestamp π, and a privacy requirement (c_τ, ℓ_τ), a user wants to select a mixin set, to generate a new RS r_τ, such that its cardinality is minimized and the following constraints are satisfied:

- *Diversity constraint:* r_τ is a recursive (c_τ, ℓ_τ)-diversity RS;
- *Non-eliminated constraint:* no token of any RS can be eliminated by the "Chain-Reaction" analysis;
- *Immutability constraint:* after proposing r_τ, each RS r_i in $R_\pi^{r_\tau}$ can maintain its requirement of recursive (c_i, ℓ_i)-diversity.

To guarantee each RS has desired anonymity, the system will require that each new RS must not violate the recursive diversity requirements of the existing RSs.

3.2 Hardness of the DA-MS Problem

We prove that the DA-MS problem is #P by reducing the problem of enumerating all perfect matchings in bipartite graphs [14].

Theorem 3.1. *The DA-MS problem is #P.*

Proof. We first prove the decision version of the DA-MS problem (DDA-MS) is #P. Given a new RS r_τ, a set of related RS set $R_\pi^{r_\tau}$, and a privacy requirement (c_τ, ℓ_τ), the DDA-MS aims to prove if r_τ satisfies the tree constraints in Definition 5.

We prove DDA-MS is #P by a reduction from the problem of enumerating all perfect matchings in bipartite graphs (EPMBG) [14]. Let $G = (S_1, S_2, E)$ be a bipartite graph with row vertex set S_1, column vertex set S_2, satisfying that $|S_1| = |S_2| = n$, and edge set $E \subseteq S_1 \times S_2$. A matching M is a subset of E, where any two edges do not intersect at the same vertex. If each vertex in G is covered by an edge in M, M is termed as a perfect matching. The EPMBG problem aims to enumerate all perfect matchings in the G.

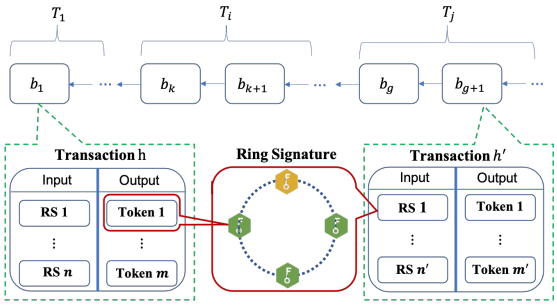


Figure 2: TokenMagic.

Given an EPMBG problem instance, we reduce it to a DDA-MS problem instance as follows: Generate a related RS set $R_\pi^T = \{r_1, \cdots, r_{n-1}\}$, where r_i is the set of vertexes in S_2 which are connected with the i^{th} vertex in S_1. The new RS r_τ is the set of vertexes in S_2 which are connected with the n^{th} vertex in S_1. To verify if r_τ is a recursive (c, ℓ)-diversity RS, we need to enumerate all possible token-RS pairs of RSs in $R_\pi^T \cup r_\tau$. Since each token can only be consumed once, the set of perfect matchings in G is the set of token-RS combinations when the consumed token in r_τ is t_τ.

Thus, if the transformed DDA-MS problem instance can be solved, the EPMBG problem instance also can be solved. Since the EPMBG problem is #P [14], the DDA-MS problem is also an #P problem. Since the DA-MS problem is an optimization problem, the hardness of the DA-MS problem is at least #P. □

4 TOKENMAGIC FRAMEWORK

Since a new RS's anonymity is impacted by the related RSs, before selecting mixins, a user needs to retrieve the mixin universe T and the related RS set $A_\pi^{r_\tau}$. The mixin universe can be retrieved by the user's interest or the blockchain system's requirement (e.g., Monero [2] requires half of the mixins are generated within the recent 1.8 days). As discussed in Subsection 2.2, by this method, the size of a related RS set may be very large and can unlimitedly increase over time. Take an extreme example, if any two existing RSs contain at least one same token, the related RS set would contain all RSs on the blockchain. To generate a new RS, a user needs to retrieve all RSs and find DTRSs among these RSs, which is costly.

In this paper, we propose a novel framework, namely TokenMagic, to retrieve the mixin universe T as well as the related RS set $R_\pi^{r_\tau}$ and generate a new RS r_τ for a given token t_τ at timestamp π. As shown in Figure 2, TokenMagic partitions the blocks in a blockchain into disjoint and sequential batches $\mathbb{B} = [B_1, B_2, \cdots]$. The number of tokens in each batch B_i is bounded in a range λ specified by the system. Specifically, when we build the batch list $\mathbb{B}$, we scan from the first block b_1 in the blockchain system and initialize the first batch B_1 as empty. We traverse the blocks in ascending order. Every time we visit a block b_x, we check whether the current batch B_i has enough tokens after adding b_x (i.e., $t(b_x) + \sum_{b_j \in B_i} t(b_j) \geq \lambda$, where $t(b_i)$ indicates the number of tokens in b_i). If B_i has enough tokens, we move to build the next batch B_{i+1}. For batch B_i, let T_i be the tokens in the blocks in B_i.

For full-node users [21] who store full blockchain data in their devices, they can build the batch list by themselves and update the

Algorithm 1: TokenMagic Framework.

Input: the consuming token t_τ, and the required diversity constraint (c, ℓ)

Output: an new RS to consume t_τ

1 retrieve the tokens T of the batch where t_τ is generated;
2 initialize $Cand_i = \emptyset$ for each $t_i \in T$;
3 **foreach** *token* t_i *in* T **do**
4 use BFS, Progressive, or Game-theoretic approaches to generate a new RS r satisfying the constraints in Definition 5;
5 **foreach** *token* t_i *in* r **do**
6 add r to $Cand_i$;
7 **return** a RS from $Cand_\tau$ randomly;

batch list when they update their blockchain state from the local state to the global state. Since λ is a public system parameter and users have a consensus about the block list in the blockchain system, users can have a consensus about the batch list too. For light-node users [21] who do not store full blockchain data in their devices, they can query the wanted batch data from full-node users. For each consuming token, its mixins are only selected within the same batch where the consuming token is generated. Note that there is a gap between the timestamp when a token is generated and the timestamp when the token is consumed by a RS in a transaction. For example, in Figure 2, a token is generated in the block b_1, but the RS which contains b_1 is proposed in the block b_{g+1}.

Thus, to generate a RS for a token t, the mixin universe is the set of tokens that are generated in the same batch of t. Thus, cardinality of the mixin universe is limited to the number of tokens in the corresponding batch. Since batches are disjoint, the mixin universes of tokens in different batches are disjoint. Furthermore, since each RS selects mixins only from its mixin universe, the related RS sets of RSs in different batches are also disjoint. Because each token can only be consumed once, the cardinality of the related RS set of a RS is bounded by the cardinality of the mixin universe. Thus, the cardinality of the related RS set is bounded by the number of tokens in the corresponding batch.

Algorithm 1 is the pseudocode of the TokenMagic framework. We first retrieve the batch b where t_τ is (line 1). To avoid adversaries inferring the consumed token by the framework and algorithms, we generate a new RS with a random procedure (line 2-5). Specifically, we first initialize candidate set $Cand_i$ of each token t_i as empty (line 1). Then, for each token t_i, we use our BFS, Progressive, or Game-theoretic approach to generate a new RS r satisfying the constraints of diversity, immutability, and non-eliminated (line 3). Since the new RS r may be eligible for multiple tokens, we add r to candidate sets of these tokens (line 4-5). After that, we randomly select a RS from $Cand_\tau$, where $Cand_\tau$ is the candidate RS set of the input token t_τ (line 6). Since blockchain systems are distributed, and the random selection in line 6 is made by users themselves, adversaries cannot observe or guess the probability distribution of selection. Therefore, adversaries cannot further infer the consumed token of a RS by the framework and algorithms.

Since the diversity requirement is customized by the user, if users think the returned RS is not desirable (e.g., the size is too large) or the framework cannot return an eligible RS, they can relax the diversity requirement by increasing c or decreasing ℓ. Besides, since a RS only selects mixins within the same batch where the consumed token is, there could be a situation that a user cannot find a RS satisfying constraint of non-eliminated. Although we cannot confirm the consumed token of each RS, we still can find out that some tokens have been consumed. In Example 1, suppose the user chooses $r_3 = \{t_1, t_3\}$ and the mixin universe is $T = \{t_1, t_2, t_3, t_4\}$. When another user wants to consume t_4, she/he cannot find a RS r satisfying non-eliminated constraint, since adversaries can easily find that t_1, t_2, and t_3 have been consumed in the previous three RSs (i.e., r_1, r_2, r_3) and t_4 must be the consumed token in the new RS r. To avoid this issue, we require that when there are i RSs in the possible related RS set R_π^T, the number of infer-able consumed tokens cannot exceed $i - \eta \cdot (|T| - i)$, where η is a system parameter.

Theorem 4.1. *Given a set of RSs, $R^* = \{r_1, \cdots, r_n\}$, suppose $T^* = \bigcup_{i=1}^{n} r_i$. If $|T^*| = n$, the tokens in T^* are all consumed.*

Proof. If a token in T^* has not been consumed, there are only $n-1$ tokens have been consumed in R^*. By the Pigeonhole principle [22], there is at least a token that has been consumed in two RSs, which is impossible. Thus, the proof is completed. □

Let $ns_j =< r_1, \cdots, r_m >$ be a *"neighbor"* set of token t_j where r_i contains t_j and RSs are sorted by the proposed timestamps. Let $T_j^\# = \bigcup_{r_i \in ns_j} r_i$. By Theorem 4.1, if the number of tokens in a neighbor set is equal to the number of RSs in the neighbor set (i.e., $|T_j^\#| = |ns_j|$), the corresponding token has been consumed. We store a neighbor set for each token and append RSs which contain the token to the neighbor set. Suppose, when the number of RSs that consumes token in T is i, the number of tokens that can be determined to have been consumed is μ_i. To avoid that users cannot find eligible RSs to consumed their tokens, we require that at any moment $i - \mu_i \geq \eta \cdot (|T| - i)$. When the number of tokens in a batch is smaller than λ, we consider $|T|$ as λ+λ'-1.

The overhead of TokenMagic framework. The TokenMagic is implemented for the Step 1 of a RS scheme introduced in Section 2.1. The Step 1 is made offline and its time complexity does not affect the throughput of a blockchain system. TokenMagic brings extra overhead to the Step 1 of RS scheme, but no extra overhead to the throughput of the online blockchain system. In conclusion, our framework only affects the speed to offline generate new RSs, but does not affect the online blockchain processes at all.

5 AN EXACT APPROACH FOR DA-MS

As illustrated in Subsection 3.2, the DA-MS problem is #P. However, if the size of the input (i.e, $|T|$) is small enough, brute-force methods still can find an optimal solution. In this section, we propose a breadth-first search (BFS) algorithm, to illustrate the problem space of DA-MS. We first introduce a basic concept as follows:

Definition 6 (Token-RS Combination)**.** Given a RS set R, a token-RS combination of R is a set of $|R|$ token-RS pairs, denoted by $u = \{p_1, p_2, \cdots, p_{|R|}\}$ where $\forall p_i, p_j \in u$, $t_i \neq t_j$ and $r_i \neq r_j$.

In Example 1, when the new RS is $r_3 = \{t_2, t_3, t_4\}$, $\{\langle t_1, r_1\rangle, \langle t_2, r_2\rangle, \langle t_3, r_3\rangle\}$ is a token-RS combination of the RSs.

5.1 The BFS Approach

Algorithm 2 illustrates the BFS approach, which applies the breadth first search strategy to check all possible valid RSs and returns the

Algorithm 2: BFS Approach.

Input: the consuming token t_τ, a mixin universe T, and the required diversity constraint (c,ℓ)
Output: an eligobile RS to consume t_τ

1 $\sigma = T \backslash t_\tau$;
2 **for** i = l_τ -1 *to* $|\sigma|$ **do**
3 $CR^i \leftarrow$ possible RSs with i tokens from σ;
4 **for** $cr \in CR^i$ **do**
5 $rs = t_\tau \cup cr$, retrieve R_π^{rs};
6 $H \leftarrow$ historical transactions of tokens in rs;
7 **if** H *does not satisfy the recursive* (c, ℓ)*-diversity* **then**
8 continue;
9 $U \leftarrow$ all token-RS combinations of $rs \cup R_\pi^{rs}$; $k = 0$;
10 **while** $k \leq |R_\pi^{rs}|$ **do**
11 $ST = \emptyset$, $k++$;
12 **foreach** u_j *in* U **do**
13 **if** $w_{j,k} \notin ST$ **then**
14 $ST = ST \cup w_{j,k}$;
15 **if** $ST \neq r_k$ **then**
16 go to Line 4 for the next for-iteration
17 **for** k = 1 *to* $|R_\pi^{rs}| + 1$ **do**
18 $D_k \leftarrow$ GetDTRSs(k, U, $|R_\pi^{rs}|$); $g = 0$;
19 **while** $g < |D_k|$ **do**
20 $g++$, $H \leftarrow$ HTs of tokens in d_g;
21 **if** H *violates recursive* (c_k, ℓ_k)*-diversity* **then**
22 go to Line 4 to the next for-iteration
23 **return** rs;

optimal one. Since t_τ has to be contained in the new RS, we let the set of candidate tokens σ for the new RS be $\sigma = T \backslash t_\tau$ (line 1). We search an eligible RS following the ascending order of the sizes of RSs, and i indicates the number of mixins in the RS (lines 2-23). In each iteration of lines 2-23, there is a set CR^i of different RSs, where each $cr \in CR^i$ contains i different tokens of σ (line 3). For each cr, the corresponding candidate RS is $rs = t_\tau \cup cr$, and the related RS set is R_π^{rs} (line 5). Let H be the set of HTs of tokens in rs (line 6). H should satisfy the recursive (c_τ, ℓ_τ)-diversity, otherwise, we go to check the next RS (lines 7-8). If H is valid, we go to check if rs satisfies the non-eliminated constraint (lines 9-16). We retrieve all valid token-RS combinations over $R_\pi^{rs} \cup rs$ as U (line 9). For each $r_k \in R_\pi^{rs} \cup rs$, we check if there are some tokens in r_k can be eliminated by traversing U (lines 10-16). ST stores all consumed token of r_k in any token-RS combination $u_j \in U$, and $w_{j,k}$ indicates the consumed token of r_k in u_j. If it satisfies the non-eliminated constraint, we check if, for any $r_k \in R_\pi^{rs} \cup rs$, its DTRSs all satisfy the recursive (c_k, ℓ_k)-diversity, where $r_{|R_\pi^T|+1} = rs$ (lines 17-22). We retrieve all DTRSs of r_k by the GetDTRSs procedure (line 18). Let H be the set of HTs of tokens in d_g (line 20). If H violates recursive (c_τ, l_τ)-diversity, we go to check the next RS (lines 21-22). If rs satisfies all constraints, we return it as the result (line 23).

The GetDTRSs Procedure. Algorithm 3 illustrates the GetDTRSs procedure. We set the candidate DTRS sets D^* and $D^\#$ as empty sets. Let DH be the set of determined HT of consumed tokens of r_k by DTRSs in D^*. Given a DTRS d_i, we can determine that the HT of

Algorithm 3: GetDTRSs

Input: a integer k, a set of token-RS combinations U, and a integer n
Output: a set of DTRSs whose size is k

1 $D^* = D^\# = \emptyset$, $DH = \emptyset$;
2 **foreach** $u \in U$ **do**
3 $p^* \leftarrow$ the token-RS pair of r_k in u, $u^* \leftarrow u \backslash p^*$;
4 **for** i = 1 *to* n *do* **do**
5 $D^* = D^*$ + the set of combinations of choosing i pairs from u^*;
6 **for** j = 1 *to* $C(n, i)$ *do* **do**
7 $DH = DH$ + the historical transaction of the token in p^*;
8 **foreach** $d_i \in D^*$ **do**
9 **for** j = 1 *to* $|U|$ **do**
10 $p^\# \leftarrow$ the token-RS pair of r_k in u_j
11 **if** $d_i \notin u_j$ **then**
12 continue;
13 **if** *the HT of the token in* $p^\#$ *is not equal to* dh_i **then**
14 go to Line 8 to next for-iteration
15 $D^\# = D^\# + d_i$;
16 remove the super set in $D^\#$;
17 **return** $D^\#$;

the consumed token in r_k is dh_i. For each token-RS combination u in U and a size i, we enumerate all combinations of i pairs in u (lines 2-7). Let p^* be the token-RS pair of r_k in u and u^* be the u excluding p^* (line 3). We add these candidate DTRSs into D^* (line 5) and their corresponding dhs in DH are the HTs of the tokens in p^* (lines 6-7). These candidate DTRSs in D^* may not be the "true" DTRSs as Definition 2 defined. For each d_i in D^*, we check all token-RS combinations in U (lines 9-14). If a DTRS d_i is in a u_j but the HT of the consumed token of r_k in u_j is not dh_i, d_i is a "fake" DTRS (lines 13-14), otherwise, we add it to $D^\#$ (line 15). Since the supersets of "true" DTRSs are also added to $D^\#$, we remove them and return the updated $D^\#$ as the result (line 16-17).

Time Complexity. Let $m = |R_\pi^{rs}| = O(n)$. There are $\sum_{i=\ell_\tau-1}^{n-1} C(n-1, i) = O(2^n)$ possible RSs in $\bigcup_{i=\ell_\tau-1}^{|\sigma|} CR_i$, where $C(n-1, i)$ is the number of combinations of selecting i elements from $n-1$ elements [23]. For each rs, the cardinality of U is $O(\prod_{r_i \in R_\pi^{rs} \cup rs} |r_i|) = O(n^m)$. In the GetDTRSs procedure, the time complexity of getting D^* is $O(n^m) \cdot \sum_{i=1}^{m} C(m, i) = O(n^m 2^m)$ and the cardinality of D^* is $O(n^m 2^m)$. Thus, the time complexity of getting $D^\#$ is $O(n^{2m} 2^m)$. Thus, the total time complexity of the BFS approach is $O(m \cdot 2^{m+n} \cdot n^{2m}) = O(n^n)$.

6 PRACTICAL SOLUTIONS FOR DA-MS

Although the BFS approach can get the optimal solution, its time complexity is unacceptable for most real-world applications. Besides, users sometimes cannot find a new RS satisfying the immutability constraint. For example, $T = \{t_1, \cdots, t_4\}$, where four tokens are from four different HTs. The first user uses $r_1 = \{t_1, t_2, t_3\}$ with recursive $(1, 2)$-diversity to consume t_1. The second user uses $r_2 = \{t_1, t_2, t_4\}$ with recursive $(2, 3)$-diversity to consume t_4. The third user uses $r_3 = \{t_1, t_2, t_3, t_4\}$ with recursive $(1, 3)$-diversity to consume t_3. Then, the fourth user cannot find an eligible RS to consume t_2 satisfying the immutability constraint.

To efficiently solve the DA-MS problem, we have *three challenges*: *(1)* how to efficiently verify if DTRSs of a RS satisfy the anonymity requirement; *(2)* how to guarantee that a user can always find a RS satisfying the immutability constraint; and *(3)* how to pick a desired set of mixins with the minimal cardinality under the constraints in Definition 5. In this section, we propose two practical configurations to tackle the first two challenges. To solve the third challenge, we design two approximation algorithms, the Progressive Algorithm and the Game-theoretic Algorithm.

6.1 Practical Configurations

In this section, we propose two practical configurations to help solve the first two challenges. These two configurations guide the mixin selection in the Step 1 of a RS scheme. In the Step 3 of a RS scheme, verifiers would check if a RS satisfies these two configurations. The *first practical configuration* is that each new RS should be the superset of some RSs in R_π^T and be disjoint with the other RSs in R_π^T, where R_π^T is the set of RSs containing tokens in T. With this configuration, we have some special RSs in R_π^T, called as *super RSs*.

Definition 7 (A Super Ring Signature)**.** Given a related RS set R_π^T, a RS r_i is a super RS if for any r_j in R_π^T that is proposed after r_i, r_j is not the super set of r_i. Let v_i be the number of RSs in R_π^T that are the subsets of r_i.

Let $S_\pi^T = \{s_1, \cdots, s_n\}$ be the set of super RSs in R_π^T. Some tokens in T may not be contained in any RS in R_π^T. We call these tokens as *fresh tokens*. Let $F_\pi^T = \{f_1, \cdots, f_n\}$ be the set of fresh tokens in T.

Definition 8 (Fresh Token)**.** Given a related RS set R_π^T, if a token t is not contained in any RS in R_π^T, t is a fresh token.

For example, $r_1 = \{t_1, t_2\}$ is proposed at time π, $r_2 = \{t_1, t_2, t_3\}$ is proposed at time $\pi + 1$, and $r_3 = \{t_4, t_5\}$ is proposed at time $\pi + 2$. Suppose $T = \{t_1, \cdots, t_6\}$ and $R_\pi^T = \{r_1, r_2, r_3\}$. Then, r_2 and r_3 are two super RSs while r_1 is not a super RS. The subset number v_i of r_2 is 2 (i.e., r_1 and r_2). Besides, t_6 is a fresh token.

Thus, by the first practical configuration, each new RS is made up of some super RSs and some fresh tokens. Then, we show how to get the token set of a DTRS of a RS.

Theorem 6.1. *In a blockchain system, a new RS is made up of some super RSs and some fresh tokens. For a related RS set R, let the super RS of a RS $r_i \in R$ be r_{i*} and its subset number be v_{i^*}. Let $\widetilde{T_{i,j}}$ be the set of tokens in r_i whose historical transaction is h_j. Let $\psi_{i,j} = r_i \backslash \widetilde{T_{i,j}}$. If $v_{i^*} < |r_i| - |\widetilde{T_{i,j}}| + 1$, there is no DTRS of r_i that can determine the historical transaction of the consumed token of r_i is h_j. Otherwise, $\psi_{i,j}$ is the token set of a DTRS of r_i that can determine the HT of the consumed token of r_i is h_j.*

Proof. To determine the historical transaction of the consumed token of r_i is h_j, $r_i \backslash \widetilde{T_{i,j}}$ should be eliminated. If $v_{i^*} < |r_i| - |\widetilde{T_{i,j}}| + 1$, by the Pigeonhole principle [22], there is at least one token in $\widetilde{T_{i,j}}$ have not been consumed. Thus, if $v_{i^*} < |r_i| - |\widetilde{T_{i,j}}| + 1$, there is no DTRS of r_i that can determine the historical transaction of the consumed token of r_i is h_j. Let d be a DTRS that determine the historical transaction of the consumed token of r_i is h_j and t^* is the set of tokens in d. Thus, $t^* \subseteq r_i \backslash \widetilde{T_{i,j}}$. Let u^* be a set of token-RS pairs whose token's union is $r_i \backslash \widetilde{T_{i,j}}$. Assume $\bar{p} = (\bar{t}, \bar{r})$ is a token-RS pair in u^* and not in d. Since $\bar{t} \in r_i$, $\bar{t}$ can be the consumed token of r_i. Since $\bar{t} \notin \widetilde{T_{i,j}}$, when $\bar{t}$ is the consumed token of r_i, the historical transaction of r_i is not h_j, which violates the definition of a DTRS. Thus, the assumption does not hold. Thus, if $v_{i^*} \geq |r_i| - |\widetilde{T_{i,j}}| + 1$, $\psi_{i,j}$ is the token set of a DTRS of r_i that can determine the historical transaction of the consumed token of r_i is h_j. □

Thus, to verify if all DTRSs of a RS satisfy the recursive diversity, we just need to verify if each $\psi_{i,j}$ satisfies the recursive diversity, which can be solved in polynomial time. Thus, the first challenge is solved. Besides, we prove that if the cardinality of the side information of an adversary is smaller than a threshold, she/he cannot confirm the historical transaction of the consumed token of a RS.

Theorem 6.2. *If the cardinality of the side information of an adversary is smaller than $|r_i| - q_M$, she/he cannot confirm the HT of the consumed token of a RS r_i, where q_M is the number of times that the most-frequent HT of tokens appears in r_i.*

Proof. By Theorem 6.1, to confirm the historical transaction of the spent token of r_i is h_j, an adversary needs to know at least r_i - $|\widetilde{T_{i,j}}|$ token-RS pairs. Since q_M is the number of times the most-frequent HT of tokens in r_i, if the cardinality of the side information of an adversary is smaller than $|r_i|$ - q_M, the adversary cannot confirm the historical transaction of the consumed token of r_i. □

By Theorem 6.2, to increase the threshold of the cardinality of an adversary's side information, when the size of a RS is fixed, a user should decrease the number of times the most-frequent HT of tokens in the RS. It exactly meets the idea of the recursive (c, ℓ)-diversity (e.g, let the set of HTs are not dominated by the most-frequent HT), which provides the motivation for us to use the recursive (c, ℓ)-diversity principle to measure the anonymity of a RS. Besides, we prove that, if before observing a new RS r, an adversary cannot confirm if t is the spent token of a RS r', after observing r, she/he still cannot confirm if t is the spent token of r'.

Theorem 6.3. *Let t be a token in a RS r'. Suppose before observing a new RS r, an adversary cannot confirm if t is the spent token of r'. The adversary still cannot confirm if t is the spent token of r' after observing r.*

Proof. Suppose D'_b is the set of DTRSs of r' before observing a new RS r and D'_a is the set of DTRSs of r' after observing r. By the first practical configuration, $r \cap r' = \emptyset$ or $r' \subseteq r$. By Theorem 6.1, if $r \cap r' = \emptyset$, $D'_b = D'_a$. Thus, if $r \cap r' = \emptyset$, after observing r, the adversary still cannot confirm if t is the spent token of r'. If $r' \subseteq r$, $t \in r$. Since the adversary cannot confirm if t is the spent token of r', the adversary also cannot confirm if t is the spent token of r. Thus, t at least may be consumed in r' or r. Thus, the adversary still cannot confirm if t is the spent token of r' after observing r. □

For the second challenge, we propose the *second practical configuration*. Specifically, if a user wants to generate a RS and guarantee that each DTRS satisfies the recursive (c, ℓ), she/he should ask the HT set of tokens of new RS satisfies the recursive $(c, \ell + 1)$-diversity.

Theorem 6.4. *If the HT set of tokens of r satisfies the recursive $(c, \ell + 1)$-diversity, the HT set of tokens of any one of its DTRSs must satisfy the recursive (c, ℓ)-diversity.*

Algorithm 4: Progressive Algorithm.

Input: the consumed token t_τ, a mixin universe T, the realted RS set R_π^T, and the required diversity constraint (c,ℓ)
Output: An eligible RS r_τ

1 get the super RS set S and the fresh token set F, get X, $r_\tau = x_\tau$, get H;
2 **while** $|H| < \ell_\tau$ **do**
3 calculate α_i for each $x_i \in X$;
4 $r_\tau \leftarrow r_\tau + x_i$ in X wit the minimal α_i, update H and X;
5 **while** $|H|$ *violates the recursive* (c_τ, ℓ_τ)*-diversity requirement* **do**
6 calculate β_i for each $x_i \in X$;
7 $r_\tau \leftarrow r_\tau + x_i$ in X wit maximal β_i, update H and X;
8 **return** r_τ

Proof. Suppose the tokens of r come from θ historical transactions. Let $q_{r,i}$ be the number of times the j^{th} most-frequent HT of tokens in r. Suppose d is a DTRS of r. By Theorem 6.1, the tokens of d come from $\theta - 1$ historical transactions. Let $q_{d,i}$ be the number of times the j^{th} most-frequent HT of tokens in d. Since the HT set of tokens of r satisfies the recursive $(c, \ell+1)$-diversity, $q_{r,1} < c \cdot (q_{r,\ell+1} + \cdots q_{r,\theta})$. Since $q_{d,1} \le q_{r,1}$ and $q_{r,\ell+1} + \cdots q_{r,\theta} \le q_{d,\ell} + \cdots q_{d,\theta-1}$, $q_{d,1} < c \cdot (q_{d,\ell} + \cdots q_{d,\theta-1})$. Thus, if the HT set of tokens of r satisfies the recursive $(c, \ell+1)$-diversity, the HT set of tokens of its any one of its DTRSs satisfies the recursive (c, ℓ)-diversity. □

With Theorem 6.4, if the HT set of tokens of a RS r satisfies the recursive $(c_\tau, \ell_\tau + 1)$-diversity, other users always can find new RSs to maintain r's recursive (c_τ, ℓ_τ)-diversity. Thus, the second challenge is solved.

6.2 Progressive Algorithm

In this subsection, to tackle the third challenge and approximately get a solution of the DA-MS problem, we propose the Progressive Algorithm. It first greedily finds a set of mixins coming from more than ℓ_τ HTs. Then it greedily changes the candidate RS to satisfy the recursive (c_τ, ℓ_τ)-diversity.

Algorithm 4 illustrates the pseudo-code of our Progressive Algorithm. We first retrieve the set of super RSs and the set of fresh tokens (line 1). With the first configuration in Subsection 6.1, each RS is consisted of some super RSs and some fresh tokens. Thus, we consider a super RS or a fresh token as a *module* which can be selected in r_τ and X is the set of modules (line 1). The module x_τ is the super RS which contains t_τ or the t_τ itself when t_τ is a fresh token (line 1). Let H be the set of HTs outputting tokens in r_τ and H_i is the set of HTs outputting tokens in x_i. We first let the tokens in r_τ come from at least ℓ_τ different HTs (line 2-4). For each $x_i \in X$, we calculate $\alpha_i = \frac{|x_i|}{\min\{\ell_\tau - |H|, |H_i \setminus (H \cap H_i)|\}}$ (line 3). We add the module x_i with the minimal α_i to r_τ and update H and X (line 4). Next, we let H satisfy the recursive (c_τ, ℓ_τ)-diversity (line 5-7). Let $tr_i = r_\tau \cup x_i$ and tH_i be the set of HTs outputting tokens in tr_i. Let $q_{i,j}$ be the number of times the j^{th} most-frequent HT of tokens in tr_i. For each x_i in X, we calculate $\beta_i = \frac{\delta - \delta_i}{|x_i|}$, where $\delta = q_{\tau,1} - c_\tau \cdot (q_{\tau,\ell_\tau} + \cdots + q_{\tau,|H|})$, $\delta_i = q_{i,1} - c_\tau \cdot (q_{i,\ell_\tau} + \cdots + q_{i,|tH_i|})$, and $|x_i|$ is the size of x_i (line 6). We greedily add the module whose β_k is maximal and update H and X(line 7). Finally, we return r_τ (line 8).

Algorithm 5: Game-theoretic Algorithm.

Input: the consumed token t_τ, a mixin universe T, the realted RS set R_π^T, and the required diversity constraint (c,ℓ)
Output: An eligible RS r_τ

1 get the super RS set S and the fresh token set F, get A, $r_\tau = a_\tau$, get H;
2 **while** $|H| < \ell_\tau$ **do**
3 calculate γ_i for each $a_i \in A$;
4 set the strategy of a_i with the minimal γ_i as ϕ, update r_τ and H;
5 **repeat**
6 **foreach** *play* $a_i \in A$ **do**
7 $e_i = \phi$;
8 **if** *the cost of* $\bar{\phi}$ *is lower* **then**
9 $e_i = \bar{\phi}$;
10 update r_τ and H;
11 **until** *reaching a Nash equilibrium*
12 **return** r_τ

Example 3. *There are four super RSs,* $s_1 = \{t_1, \cdots, t_6\}$, $s_2 = \{t_7, \cdots, t_{10}\}$, $s_3 = \{t_{11}, t_{12}\}$, *and* $s_4 = \{t_{13}, \cdots, t_{15}\}$. *The tokens* t_1, t_2, t_7 *and* t_8 *are from* h_1. *The tokens* t_3, t_4, *and* t_9 *are from* h_2. *The tokens* t_5, t_{13} *and* t_{14} *are from* h_3. *The tokens* t_6 *and* t_{10} *are from* h_6. *The token* t_{11} *and* t_{15} *are from* h_4. *The token* t_{12} *are from* h_5. *A user wants to generate a RS to consume* t_{11} *with the recursive (1,4)-diversity. Then,* $x_\tau = s_3$. *In the first iteration of the first while-loop, we add* s_2 *to* r_τ *since* α_2 *is minimal. After the first while-loop, we get* $r_\tau = s_3 \cup s_2$. *In the first iteration of the second while-loop, we add* s_4 *to* r_τ, *since* $\beta_4 = \frac{1}{3}$ *and* $\beta_1 = -\frac{1}{6}$.

Theorem 6.5. *The Progressive Algorithm's approximation ratio is* $\epsilon + \frac{q_M \cdot z_M}{10^{-\gamma}}$, *where* $\epsilon = \sum_{i=1}^{\ell_\tau} \frac{1}{i}$, q_M *is the number of times the most-frequent HT of tokens in* T, z_M *is the maximal size of a super RS in* R_π^T, *and* γ *is the minimum integer such that* $10^\gamma \cdot \epsilon_\tau$ *is an integer.*

Proof. Suppose OPT is the size of the optimal result of the input DA-MS problem instance. Let z^* be the minimal size of a selection from X whose tokens are from at least ℓ_τ different HTs. Let q_{min} be the number of times the most-infrequent HT of tokens in T. Thus, $OPT \ge z^* \ge \ell_\tau q_{min}$. Let $z^\#$ be the size of r_τ when the first while-loop terminates. By [24], $\frac{z^\#}{z^*} \le \epsilon$. Let x_k be the last module which is added to r_τ and z_r is the size of r_τ that the Progressive Algorithm returns. In each iteration of the second while-loop, δ decrease at least $10^{-\gamma}$. Since $|x_k| \le z_M$ and $\delta \le q_M$, $z_r \le z^\# + \frac{q_M \cdot z_M}{10^{-\gamma}}$. Thus, $\frac{z_r}{OPT} \le \frac{z^\#}{OPT} + \frac{q_M \cdot z_M}{10^{-\gamma} \cdot \ell_\tau \cdot q_{min}} \le \epsilon + \frac{q_M \cdot z_M}{10^{-\gamma}}$. □

In Example 3, $q_M = 4$, $q_{min} = 1$, $z_M = 6$, and $\gamma = 0$. These parameters are determined by the problem instance. When ℓ_τ is larger, it is harder to satisfy the diversity constraint. Then, the first while-loop (lines 2-4) would get a RS with higher size, which is denoted by the first term of the approximation ratio (i.e., ϵ). When c_τ is smaller, it is harder to satisfy the diversity constraint. Then, the second while-loop (lines 5-7) would get a RS with higher size, which is represented by the second term of the approximation ratio (i.e., $\frac{q_M \cdot z_M}{10^{-\gamma}}$).

Time Complexity. Let the cardinality of the mixin universe be n, i.e., $n = |T|$. In each iteration of the first while-loop, $|H|$ increases by at least 1. Thus, the round of the first while-loop is $O(\ell_\tau)$. For each round, there are $O(n)$ modules in X. For each module, the

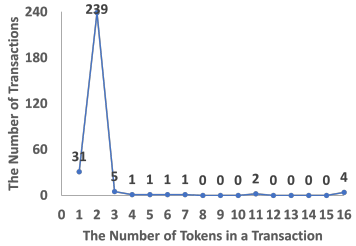


Figure 3: The Distribution of the Number of Tokens.

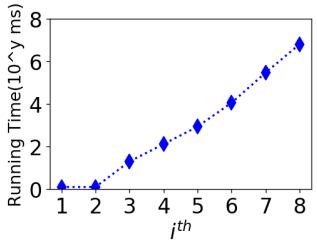


Figure 4: The running time of the i^{th} RS by the TM_B Approach.

calculation of α_k is $O(n)$. Compared with n, ℓ_τ usually can be ignored. Thus, the time complexity of the first while-loop is $O(n^2)$. In each iteration of the second while-loop, δ decreases by at least $10^{-\gamma}$. Thus, the round of the second while-loop is $O(\frac{q_M}{10^{-\gamma}})$. For each module, the calculation of β_k is $O(n)$. Compared with n, $\frac{q_M}{10^{-\gamma}}$ usually can be ignored. Thus, the time complexity of the second while-loop is $O(n^2)$. Thus, the total time complexity is $O(n^2)$.

6.3 Game-theoretic Algorithm

Since the Progressive Algorithm greedily picks modules (super RSs or fresh tokens), it may result in local optimal solutions. In this subsection, we develop a game-theoretic algorithm, where super RSs and fresh tokens will be considered as players to find an eligible r_τ whose size is as small as possible until the game reaches a Nash equilibrium [25]. In this section, we prove that the game can reach a Nash equilibrium within polynomial time and the size of r_τ is theoretically guaranteed when a Nash equilibrium is converged.

Algorithm 5 illustrates the pseudo-code of our Game-theoretic Algorithm. We first retrieve the set of super RSs and the set of fresh tokens (line 1). With the first configuration in Subsection 6.1, each new RS should be made up of some super RSs and some fresh tokens. Thus, we consider a super RS or a fresh token as a player who competes the right to be picked in r_τ with other players. Each player has two strategies, ϕ and $\bar{\phi}$, which indicate being selected and being not selected in the new RS, respectively. The player a_τ is the super RS which contains t_τ or the t_τ itself when t_τ is a fresh token (line 1). Since a_τ has to been contained in r_τ, we remove it from A. Suppose H is the set of HTs outputting tokens in r_τ and H_i is the set of HTs outputting tokens in a_i. Then, we first let the tokens in r_τ be from at least ℓ_τ different HTs (line 2-4). For each player $a_i \in A$, we calculate $\gamma_i = \frac{|a_i|}{\min\{\ell_\tau - |H|, |H_i \setminus (H \cap H_i)|\}}$ (line 3). We add the player a_i with the minimal γ_i to r_τ and update H (line 4). Next, we let H satisfy the recursive (c_τ, ℓ_τ)-diversity (line 5-11). For each player, we calculate the cost of two strategies when other users' strategies are given and chose the strategy with the minimal cost (line 6-10). When the cost of two strategies are the same, we chose the strategy ϕ (line 7). Specifically, for each player a_i, the cost function of strategies is $c_i(e_i, \overline{e_i}) = \begin{cases} \frac{|\widetilde{r_\tau}|}{|A|}, & H \text{ satisfies the recursive } (c_\tau, \ell_\tau)\text{-diversity} \\ \infty, & \text{otherwise} \end{cases}$, where $\overline{e_i}$ is the set of strategies of players in A except a_i, and $\widetilde{r_\tau}$ is the RS generated by e_i and $\overline{e_i}$ (line 8). Finally, we return r_τ as the result of the Game-theoretic Algorithm (line 12).

In Example 3, by the Game-theoretic Algorithm, the RS is $r_\tau = s_1 \cup s_3$. Specifically, after the first while-loop, we get $r_\tau = s_3 \cup s_2$. Then, for the player s_1, the $c_1(\phi, \overline{e_1}) = c_1(\bar{\phi}, \overline{e_1}) = \infty$. Thus, $e_1 = \phi$ and $r_\tau = s_1 \cup s_3$. For the player s_2, the $c_2(\bar{\phi}, \overline{e_2}) = \frac{8}{3}$ and $c_2(\phi, \overline{e_2}) = \infty$. Thus, $e_2 = \bar{\phi}$ and $r_\tau = s_1 \cup s_3$.

Next, we prove that the Game-theoretic Algorithm can converge on a Nash equilibrium within polynomial time.

Theorem 6.6. *Game-theoretic Algorithm can converge on a Nash equilibrium within $O(n^3)$, where $n = |T|$.*

Proof. Define the potential function of the game as $\Phi(\tilde{e}) = \begin{cases} \frac{|\widetilde{r_\tau}|}{|A|}, & H \text{ satisfies the recursive } (c_\tau, \ell_\tau)\text{-diversity} \\ \infty, & \text{otherwise} \end{cases}$, where $\tilde{e}$ is the set of strategies of players in A, $\widetilde{r_\tau}$ is the RS which is generated according to $\tilde{e}$, and H is the set of HTs outputting tokens in r_τ. Thus, $\forall a_i \in A$, $c_i(e_i, \overline{e_i}) - c_i(e_i', \overline{e_i}) = \Phi(\tilde{e}) - \Phi(\tilde{e}')$, where $\tilde{e} = e_i + \overline{e_i}$ and $\tilde{e}' = e_i' + \overline{e_i}$. By [26], when $\forall a_i \in A$, $c_i(e_i, \overline{e_i}) - c_i(e_i', \overline{e_i}) = \Phi(\tilde{e}) - \Phi(\tilde{e}')$ and the strategy set of each player is finite, the game can converge on a Nash equilibrium. Since in each iteration, $\Phi(\tilde{e})$ decreases by at least $\frac{1}{A}$ and $\Phi(\tilde{e}) \le \frac{n}{|A|}$, the number of iteration is $O(n)$. For each iteration, we calculate the cost function of $O(n)$ players and the time complexity of each calculation is $O(n)$. Thus, the total time complexity is $O(n^3)$. □

Then, we prove the approximation ratio of the Game-theoretic Algorithm. Suppose OPT is the cost of the optimal solution to the problem where the the sum of each user's cost function is minimized. As standard measurements, *price of stability (PoS)* and *price of anarchy (PoA)* are often used to evaluate the quality of an equilibrium [27–29]. Specifically, *PoS* of a game is the ratio of the minimized cost when it reaches an equilibrium to the *OPT*. Besides, *PoA* of a game is the ratio of the maximized cost when it reaches an equilibrium to the *OPT*.

Theorem 6.7. *The PoS is bounded by* 1 *and the PoA is bounded by $q_M \cdot (1 + \frac{1}{c_\tau \cdot \ell_\tau}) + \frac{z_M}{\ell_\tau}$, where q_M is the number of times that the most-frequent HT of tokens appears in T, and z_M is the maximal size of a super RS in R_π^T.*

Proof. By the definition of the potential function in the proof of Theorem 6.6, when a Nash equilibrium is converged, $\Phi(\tilde{e}) = \frac{|\widetilde{r_\tau}|}{|A|}$. Let OPT be the size of the optimal result, denoted by r_o, of the input DA-MS problem instance and let $\tilde{e}^*$ be the corresponding strategy set of players in A. Let $\tilde{e}^\#$ be the strategy set of players in A that yields the minimum of $\Phi(\tilde{e})$. In other words, when the strategy set of players in A is $\tilde{e}^\#$, the game reaches the best Nash equilibrium. Let $r^\#$ be the RS when the strategy set of players is $\tilde{e}^\#$. Thus, $OPT = |A| \cdot \Phi(\tilde{e}^*) \ge |A| \cdot \Phi(\tilde{e}^\#) = |r^\#|$. Thus, $PoS = \frac{|r^\#|}{OPT} \le 1$.

Let r_c be a RS that the Game-theoretic Approach converges, a_k be the last player who changes her/his strategy before the converging, and r_k be the RS before the last change of r_c. Thus, $|r_c| \le |r_k| + |a_k|$.

Table 2: Experimental Settings (Real).

Parameters	Values
the c_τ of the recursive (c_τ, ℓ_τ)-diversity	0.2, 0.4, **0.6**, 0.8, 1
the ℓ_τ of the recursive (c_τ, ℓ_τ)-diversity	20, 30, **40**, 50, 60

Table 3: Experimental Settings (Synthetic).

Parameters	Values
the size of each super RS $\lvert s_i \rvert$	[1,10], [5,15], **[10,20]**, [15,25], [20,30]
the number of super RSs $\lvert S \rvert$	10, 30, **50**, 70, 90
the number of fresh tokens $\lvert F \rvert$	0, 5, **10**, 15, 20
the variance σ of tokens' distribution	8, 10, **12**, 14, 16

Denote $q_{c,i}$ as the number of times the i^{th} most-frequent HT of tokens in r_c and $q_{k,i}$ as the number of times the i^{th} most-frequent HT of tokens in r_k. Since r_k violates the recursive (c_τ, ℓ_τ)-diversity, $q_{k,\ell_\tau} + \cdots + q_{k,|H_k|} \leq \frac{q_{k,1}}{c_\tau} \leq \frac{q_M}{c_\tau}$, where H_k is the set of different HTs outputting tokens in r_k. Thus, $|r_c| \leq |r_k| + |a_k| = q_{k,1} + \cdots q_{k,|H_k|} + |a_k| \leq q_M \cdot (\ell_\tau - 1) + \frac{q_M}{c_\tau} + z_M$. Let q_{min} be the number of times the most-infrequent HT of tokens in T. Since r_o satisfies the recursive (c_τ, ℓ_τ)-diversity constraint, $OPT \geq \ell_\tau \cdot q_{min}$. Thus, $PoA = \frac{|r_c|}{OPT} \leq \frac{q_M \cdot (\ell_\tau - 1) + \frac{q_M}{c_\tau} + z_M}{\ell_\tau \cdot q_{min}} \leq q_M \cdot (1 + \frac{1}{c_\tau \cdot \ell_\tau}) + \frac{z_M}{\ell_\tau}$. □

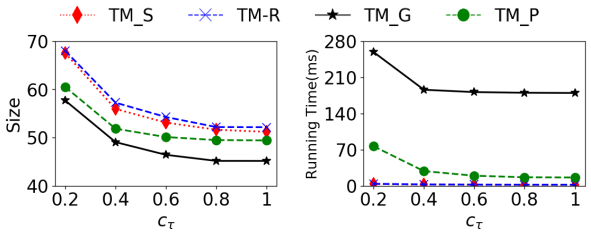
When q_M is higher, it is harder to satisfy the diversity constraint and the algorithm would pick more tokens, which would increase the approximation ratio. Because of the first practical configuration in Subsection 6.1, each super RS cannot be partly picked. Thus, when z_M is larger, the final RS which is generated by our algorithms would be larger, which would increase the approximation ratio.

7 EXPERIMENTAL STUDY

7.1 Experiment Configuration

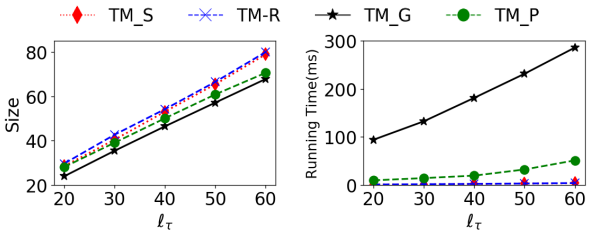
Data Sets. We test our proposed practical approaches over real data sets as well as synthetic data sets. For real data sets, we retrieve the tokens in the blocks between 2,028,242 and 2,028,273 from the Monero System, which were generated in one hour. There are 285 transactions and 633 tokens in these blocks. Since in Monero System, most RS's size is 11, we generate 57 super RSs and 6 fresh tokens. For each super RSs, it randomly selects 11 tokens. Figure 3 shows the distribution of the number of tokens in a transaction. Most transactions output two tokens. In other words, the distribution of HTs of tokens is almost uniform, and in a RS most q_i does not exceed 2. If we set c_τ large and set ℓ_τ small, the recursive (c_τ, ℓ_τ)-diversity constraint would be very relax. In experiments over the real data sets, we vary c_τ from 0.2 to 1 and vary ℓ_τ from 20 to 60.

To further examine the effects of the distribution of HTs of tokens, the number of super RSs, the number of fresh tokens, and the size of each super RS, we generate the synthetic data sets and run the experiments. For synthetic data sets, we generate $|S|$ super RSs where $|S|$ is varied from 10 to 90. We uniformly set the size of a super RS within the range $[s^-, s^+]$, which is varied from $[1, 10]$ to $[20, 30]$. We generate $|F|$ fresh tokens, where $|F|$ is varied from 0 to 20. In Monero, as shown in the real data sets, the number of tokens in an hour is less than 800. Thus, this setting is realistic and can cover most application scenarios. Moreover, we set the HT of each token by the normal distribution with variances from 8 to 16. When the variance is 16 and the number of tokens is around 800, the number of tokens from the same historical transaction is around 16. In Monero [2], the maximal number of tokens from the same historical transaction is 16. Thus, our setting is practical.



(a) Size (b) Running Time

Figure 5: Effect of c_τ of the Recursive (c_τ, ℓ_τ)-Diversity (Real).



(a) Size (b) Running Time

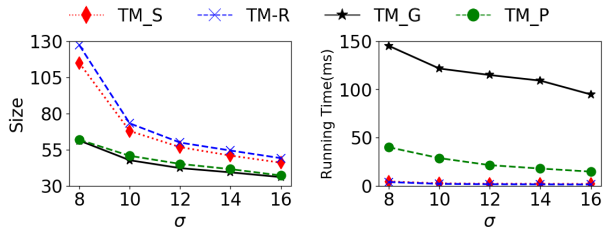
Figure 6: Effect of c_τ of the Recursive (c_τ, ℓ_τ)-Diversity (Real).

Compared Approaches. Let TM_P and TM_G indicate the TokenMagic framework with the Progressive Algorithm and the Game-theoretic Algorithm to generate RSs, respectively. We conduct experiments on both the real data sets and the synthetic data sets to evaluate the effectiveness and efficiency of our two approaches, TM_P and TM_G, in terms of the size of the new RS and the running time. We compare our approaches with two baseline algorithms, noted as TM_S and TM_R. The TM_S denotes the TokenMagic framework using the Smallest Algorithm to generate RSs, which repeatedly adds the super RS or the fresh token with smallest size to the RS until the new RS is eligible. The TM_R is the TokenMagic framework with the Random Algorithm to generate RSs, which repeatedly adds a super RS or a fresh token in random until the new RS is eligible.

Table 2 and Table 3 show experiment settings on two data sets, where the default values of parameters are in bold font. In each set of experiments, we vary one parameter, while keeping other parameters to their default values. For each experiment, we sample 1000 problem instances. We report the average value of the running time and the size of the RS. All experiments were run on an Intel CPU @2.2 GHz with 16GM RAM in Java.

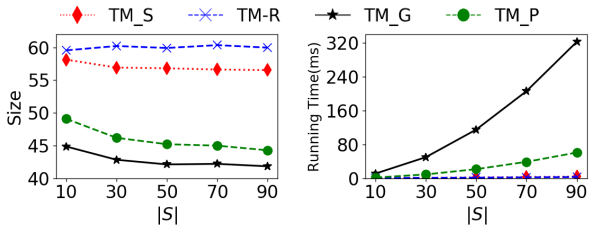
7.2 Result of the Small-Scale Data Sets

We first run the TokenMagic framework with BFS to generate RSs (noted as TM_B) on a synthetic small-scale data set. There are 20 tokens in T. Each RS requires a recursive (5,3)-diversity. Figure 4 illustrates the running time of the generation of the i^{th} RS. As illustrated in Section 5, the running time of TM_B increases exponentially. The running time of generating 8^{th} RS is around 2 hours. Thus, it is important to implement the practical configurations in Subsection 6.1 and use our practical algorithms, the Progressive Algorithm and the Game-theoretic Algorithm, to generate RSs.



(a) Size (b) Running Time

Figure 7: Effect of the variance of tokens' distribution (Synthetic).



(a) Size (b) Running Time

Figure 8: Effect of the number of super RSs $|S|$ (Synthetic).

7.3 Results on Real Data Sets

To exam the performance of TM_P and TM_G, we conduct experiments on the real data sets.

Effect of c_τ for the recursive (c_τ, ℓ_τ)-diversity requirement. Figure 5 illustrates the experimental result on different c_τ, from 0.2 to 1. In Figure 5(a), when c_τ gets larger, the sizes of the new RSs that generated by four approaches decrease. Because when c_τ is larger, it is easier to satisfy the diversity constraint and the new RS can contain fewer tokens from infrequent HTs. The sizes of the new RSs that generated by TM_G and TM_P are much less than the sizes of the new RSs that generated by two baseline algorithms. In Figure 5(b), at beginning, when c_τ increases, the running time of four approaches decreases. Because when c_τ is larger, it is easier to satisfy the diversity constraint. Then, when c_τ is large enough, the running time of four approaches keeps stable. Because when c_τ is large enough, the constraint caused by c_τ is relaxed and the running time is dominated by other factors.

Effect of ℓ_τ for the recursive (c_τ, ℓ_τ)-diversity requirement. Figure 6 illustrates the experimental result on different ℓ_τ, from 20 to 60. In Figure 6(a), when ℓ_τ increases, the sizes of the new RSs increase linearly as proved in Theorem 6.5 and 6.7. Because when ℓ_τ is larger, the new RS has to contain more tokens from infrequent HTs to meet the recursive (c_τ, ℓ_τ)-diversity constraint. In Figure 6(b), when ℓ_τ increases, the running time of four approaches increases. Among four approaches, TM_G is the slowest and it is more sensitive to the change of ℓ_τ compared with the other three algorithms.

In the experiments on the real dataset, we find that the sizes of the new RSs generated by our proposed two approaches are much less than the sizes of the new RSs that generated by two baseline algorithms. However, since the size of each super RS is the same and the distribution of HTs of tokens is almost uniform, the difference between the performance of four approaches is not obvious.

7.4 Results on Synthetic Data Sets

We further examine the effects of the distribution of HTs of tokens, the number of super RSs, the number of fresh tokens, and the size of each super RS on the synthetic data sets.

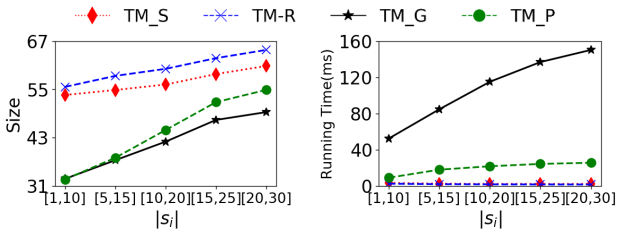
Effect of the variance σ of distribution of the HTs of tokens. Figure 7 illustrates the experimental results on different σ, from 8 to 16. In Figure 7(a), when σ gets larger, the sizes of the new RSs that generated by four approaches decrease. When σ gets larger, tokens in T would come from more different HTs and the number of times a HT outputting a token in T decrease. Thus, when σ gets larger, it is easier to satisfy the diversity constraint and the new RS can contain fewer tokens from infrequent HTs. In Figure 7(b), when σ increases, the running time of four approaches decreases. Although TM_P also takes more time than the two baseline algorithms, the time it takes is much less than that of TM_G.

Effect of the size of each super RS $|s_i|$. Figure 9 shows the experimental result on different ranges of $|s_i|$, from [1, 10] to [20, 30]. In Figure 9(a), when $|s_i|$ increases, the sizes of the new RSs generated by four approaches increase. Because of the *first practical configuration*, each super RS cannot be partially picked in the new RS. Thus, when $|s_i|$ increases, the sizes of new RSs increase. In Figure 8(b), when $|s_i|$ increases, the running time of four approaches increases. Because when $|s_i|$ increases, the number of tokens $|T|$ increases.

Effect of the number of super RSs $|S|$. Figure 8 illustrates the experimental result on different numbers of super RSs, $|S|$, from 10 to 90. In Figure 8(a), when $|S|$ gets larger, the sizes of the new RS generated by TM_R keep stable but the sizes of the new RSs generated by the other three approaches decrease. Since TM_R picks mixins randomly, the increment of the number of super RSs has no effect on the size of the new RS. But when $|S|$ gets larger, there are more candidate super RSs can be picked and the other three algorithms can find RSs with smaller sizes. As shown in Figure 8(b), when $|S|$ increases, the running time of four approaches increases. Specifically, the running time of TM_P increases quadratically and the running time of TM_G increases cubically, which confirm the time complexities we analyzed in Section 6.

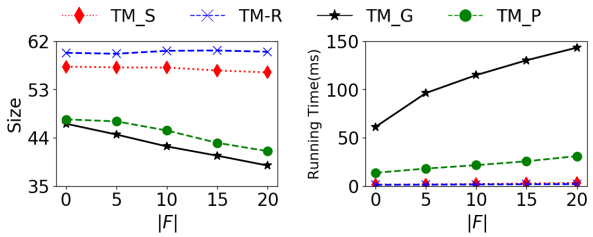
Effect of the number of fresh tokens $|F|$. Figure 10 illustrates the experimental result on different numbers of fresh tokens, $|F|$, from 0 to 20. In Figure 10(a), when $|F|$ gets larger, the sizes of the new RS generated by TM_R keep stable but the sizes of the new RSs generated by the other three approaches decrease. Since TM_R picks mixins randomly, the increment of $|F|$ has no effect on the size of the new RS. But when $|F|$ gets larger, there are more candidate fresh tokens can be picked and the other three algorithms can find RSs with smaller sizes. In Figure 10(b), when $|F|$ increases, the running time of four approaches also increases. Because when $|F|$ is larger, there are more candidate fresh tokens that can be picked, which increases the time complexity of each algorithm.

Summary of Results. For blockchain systems adopting RS schemes to preserve users' privacy, our methods can help to resist the "chain-reaction" attack. The TM_G outputs the RS with the smallest size while it is the slowest. Since RSs are generated offline and the generation time of a RS does not affect the throughput of a blockchain system, it does not matter that the running time of TM_G is a little higher, compared with other approaches. Thus, for the applications where the transaction fee (proportional to the size of RS) is high, like cryptocurrencies (e.g., Moner [2], Bytecoin[30]) and storage sharing systems (e.g., Oodrive [31]), users can save transaction fee



(a) Size (b) Running Time

Figure 9: Effect of the size of each super RS $|s_i|$ (Synthetic).



(a) Size (b) Running Time

Figure 10: Effect of the number of fresh tokens $|F|$ (Synthetic).

from using TM_G, since the size of a RS generated by TM_G is the smallest. For the applications where users need to make transactions fast, like healthcare systems (e.g., MedBlock [32]) and e-voting systems (e.g., Blockvotes [33]), TM_G is infeasible. For example, in a polling station, if the time cost of generating a RS for a vote increases by 100ms, it would delay more than one minute for a queue with 1000 voters. Thus, for the applications where users should make transactions quickly and the size of a RS is not very sensitive, like healthcare systems (e.g., MedBlock [32]) and e-voting systems (e.g., Blockvotes [33]), TM_P is suitable.

8 RELATED WORK

To solve the privacy problem, some researchers have proposed some privacy-preserved blockchain systems. These works can be classified into two categories. The works in the first category focus on developing mixing protocols. In these protocols, anonymous service providers use mixing protocols to confuse the trails of transactions. However, there is a risk of theft by the service. Researchers proposed a novel protocol, Mixcoin [34], with an accountability mechanism, to avoid such risk. Generally, the server would sign warranties to users and if the server steals users' tokens, users can publish warranties to expose the theft. However, by these mixing protocols, a user has to communicate with others, and she/he cannot make a transaction by herself/himself. When the communication channel is unsafe, users' privacy may be disclosed.

The second category focus on developing advanced encryption methods. The ZCash privacy-preserving blockchain system [35] is built on the zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) protocol [36]. However, it needs a trusted entity to initially set up the system, which is unacceptable in many distributed applications. Since the RS schemes does not require an initial set up by a trusted entity, they are widely implemented in blockchain systems in various domains, like cryptocurrency [2, 30], e-voting [3, 33], healthcare [4] and storage sharing [5, 31]. The RS schemes are used to conceal the real token of a transaction and a token can represent the identity of a user, not only cryptocurrency. In a cryptocurrency system [2], the RS scheme is adopted to hide the real token of an input in a transaction and in a RS, a mixin is a UTXO. In an e-voting system [3], the RS scheme is adopted to hide the real user of a vote, and in a RS, a mixin is a user's identity. In a healthcare system [4], the RS scheme is adopted to hide the real patient of a transaction, and in a RS, a mixin is a patient's identity. In a storage sharing system [5], the RS scheme is adopted to hide the real user of a transaction who wants to operate the data, and in a RS, a mixin is a user's identity. These works are the variants of the RS scheme adopted in the Monero system, and they all do not consider the anonymity of a RS.

As introduced in Subsection 2.1, the generation of a RS includes two processes (i.e., Step 1 and Step 2). *Step 1* picks some mixins and *Step 2* runs an encryption algorithm to generate a RS using the selected mixins. The existing works focus on decreasing the number of auxiliary parameters in *Step 2*. For the advanced RS scheme [16], the number of auxiliary parameters is $O(\log n)$ where n is the number of tokens. However, it randomly pick mixins in Step 1 [2], whose time complexity is $O(n)$ where n is the number of tokens. Our methods focus on Step 1. The time complexities of the Progressive Algorithm and the Game-theoretic Algorithm are $O(n^2)$ and $O(n^3)$, respectively. Although the time cost of our algorithms is higher, the RSs generated by our algorithms can resist the "chain-reaction" analysis and the homogeneity attack. Besides, since Step 1 is made offline, the increase of its time complexity does not affect the throughput of a system. For Step 2, we use the advanced encryption algorithm [16].

9 CONCLUSION

In this paper, we target on solving privacy weakness in block-chain systems that adopt RS schemes. We formulate the diversity-aware mixin selection problem, which aims to find a RS that satisfies the diversity requirement and contains a minimal number of tokens. We prove the problem is #P. We propose an exact algorithm to solve it when the input is small. Besides, we propose two practical configurations with two approximation algorithms, the Progressive Algorithm and the Game-theoretic Algorithm, with theoretic guarantees. When evaluated on the real and synthetic data sets, our approaches achieved clearly better performance than two baselines.

10 ACKNOWLEDGMENT

This work is partially supported by National Key Research and Development Program of China Grant no. 2018AAA0101100, the Hong Kong RGC GRF Project 16213620, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, Didi-HKUST joint research lab project, and Wechat and Webank Research Grants. Besides, Xuemin Lin is supported by the National Key R&D Program of China under grant 2018YFB1003504, NSFC61232006, ARC DP200101338, ARC DP180103096 and ARC DP170101628. Peng Cheng is supported by Shanghai Pujiang Program 19PJ1403300. Corresponding author: Peng Cheng.

REFERENCES

[1] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1085–1100, 2017.

[2] "[online] Monero." https://www.getmonero.org/.

[3] Y. Wu, "An e-voting system based on blockchain and ring signature," *Master. University of Birmingham*, 2017.

[4] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A decentralized privacy-preserving healthcare blockchain for iot," *Sensors*, vol. 19, no. 2, p. 326, 2019.

[5] S. Rahmadika and K.-H. Rhee, "Toward privacy-preserving shared storage in untrusted blockchain p2p networks," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[6] M. E. Andres, N. E. Bordenabe, K. Cjhatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, pp. 901–914, 2013.

[7] J. Lee and C. Clifton, "How much is enough? choosing ε for differential privacy," in *Proceedings of the 14th International Conference on Information Security*, ISC'11, (Berlin, Heidelberg), p. 325–340, Springer-Verlag, 2011.

[8] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," *arXiv preprint arXiv:1212.1984*, 2012.

[9] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[10] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, *et al.*, "An empirical analysis of traceability in the monero blockchain," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 143–163, 2018.

[11] J. O. M. Chervinski, D. Kreutz, and J. Yu, "Floodxmr: Low-cost transaction flooding attack with monero's bulletproof protocol.," *IACR Cryptology ePrint Archive*, vol. 2019, p. 455, 2019.

[12] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd International Conference on Data Engineering*, pp. 106–115, IEEE, 2007.

[13] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[14] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.

[15] U. W. Chohan, "The double spending problem and cryptocurrencies," *Available at SSRN 3090174*, 2017.

[16] T. H. Yuen, S.-f. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *International Conference on Financial Cryptography and Data Security*, pp. 464–483, Springer, 2020.

[17] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *International Conference on Financial Cryptography and Data Security*, pp. 469–485, Springer, 2014.

[18] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*, pp. 6–24, Springer, 2013.

[19] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *International Conference on Financial Cryptography and Data Security*, pp. 34–51, Springer, 2013.

[20] J. Giraldo, A. Cardenas, and M. Kantarcioglu, "Security and privacy trade-offs in cps by leveraging inherent differential privacy," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1313–1318, IEEE, 2017.

[21] C. Xu, C. Zhang, and J. Xu, "vchain: Enabling verifiable boolean range queries over blockchain databases," in *Proceedings of the 2019 international conference on management of data*, pp. 141–158, 2019.

[22] W. A. Trybulec, "Pigeon hole principle," *Journal of Formalized Mathematics*, vol. 2, no. 199, p. 0, 1990.

[23] C. M. Grinstead and J. L. Snell, *Introduction to probability*. American Mathematical Soc., 2012.

[24] T. Elomaa and J. Kujala, "Covering analysis of the greedy algorithm for partial cover," in *Algorithms and applications*, pp. 102–113, Springer, 2010.

[25] J. F. Nash *et al.*, "Equilibrium points in n-person games," *PNAS*, vol. 36, no. 1, pp. 48–49, 1950.

[26] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[27] W. Ni, P. Cheng, L. Chen, and X. Lin, "Task allocation in dependency-aware spatial crowdsourcing," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 985–996, IEEE, 2020.

[28] N. Armenatzoglou, H. Pham, V. Ntranos, D. Papadias, and C. Shahabi, "Real-time multi-criteria social graph partitioning: A game theoretic approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1617–1628, 2015.

[29] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[30] "[online] Bytecoin." https://bytecoin.org/.

[31] V.-H. Hoang, E. Lehtihet, and Y. Ghamri-Doudane, "Privacy-preserving blockchain-based data sharing platform for decentralized storage systems," in *2020 IFIP Networking Conference (Networking)*, pp. 280–288, IEEE, 2020.

[32] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of medical systems*, vol. 42, no. 8, p. 136, 2018.

[33] "[online] Blockvotes." http://juliancrespo.com/blockvote.html.

[34] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," pp. 486–504, 2014.

[35] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, IEEE, 2014.

[36] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 781–796, 2014.