Analysis of CryptoNote Transaction Graphs using the Dulmage-Mendelsohn Decomposition

Saravanan Vijayakumaran
Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India

Abstract

CryptoNote blockchains like Monero represent the largest public deployments of linkable ring signatures. Beginning with the work of Kumar et al. (ESORICS 2017) and Möser et al. (PoPETs 2018), several techniques have been proposed to trace CryptoNote transactions, i.e. identify the actual signing key, by using the transaction history. Yu et al. (FC 2019) introduced the closed set attack for undeniable traceability and proved that it is optimal by showing that it has the same performance as the brute-force attack. However, they could only implement an approximation of the closed set attack due to its exponential time complexity. In this paper, we show that the Dulmage-Mendelsohn (DM) decomposition of bipartite graphs gives a polynomial-time implementation of the closed set attack. Our contribution includes open source implementations of the DM decomposition and the clustering algorithm (the approximation to the closed set attack proposed by Yu et al). Using these implementations, we evaluate the empirical performance of these methods on the Monero dataset in two ways — firstly using data only from the main Monero chain and secondly using data from four hard forks of Monero in addition to the main Monero chain. We have released the scripts used to perform the empirical analysis along with step-by-step instructions.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Cryptocurrency, CryptoNote, Monero, Traceability

Digital Object Identifier 10.4230/LIPIcs.AFT.2023.1

Supplementary Material The programs used to generate the empirical results in this paper and the documentation on how to use them can be found at the following links.
Software: `https://github.com/avras/cryptonote-analysis`
Documentation: `https://www.respectedsir.com/cna`

1 Introduction

1.1 CryptoNote Transactions

Coins in CryptoNote blockchains are associated with stealth addresses, which are also called one-time addresses or transaction outputs [21]. We will use the term output for brevity. Each output is uniquely identified by a public key, which is a point on an elliptic curve. To spend from an output, the spender needs to know the corresponding secret key.

In a transaction, the spender creates a *ring* of outputs which is a set containing the output being spent and some other outputs sampled from the CryptoNote blockchain (these are called decoy outputs or *mixins*). The spender generates a linkable ring signature [15] over the ring of outputs using the secret key of the output being spent. This signature only reveals that the signer knows the secret key corresponding to one of the ring outputs, without revealing the identity of the actual output being spent.

To prevent double spending from an output, the linkable ring signature reveals the *key image* of the output being spent. The key image of an output is a collision-resistant one-way function of the secret key. For example, in Monero the public key associated with an output is given by $P = xG$ where G is the base point of an elliptic curve group and x is the secret key. The key image I of the output associated with P is given by $xH_{\mathrm{p}}(P)$, where $H_{\mathrm{p}}(\cdot)$ is a cryptographic hash function that maps its input to a point on the elliptic curve. In the

rest of this paper, when we speak of *the key image of an output* we mean the result of the one-way function applied to the secret key associated with the output.

If the owner of the output corresponding to P tries to spend the coins associated with it more than once, then the key image I would appear again in the second transaction, identifying it as a double spending transaction. Such transactions will be rejected by miners, as blocks including them would be considered invalid by the network. In this sense, the key image acts as a *nullifier* of an output, ensuring that it is spent only once.

1.2 CryptoNote Transaction Graphs

Consider a CryptoNote transaction which spends from two existing outputs and creates three new outputs as illustrated in Figure 1. The new outputs are denoted by T_1, T_2, T_3. The transaction has two rings of outputs of size five each, $(P_1, P_2, \ldots, P_5)$ and $(Q_1, Q_2, \ldots, Q_5)$. Exactly one output from each ring is being spent in the transaction. The key images I_1 and I_2 of the outputs being spent are revealed in the transaction. Note that the two rings can have common outputs.

For the purpose of illustration, suppose that the two rings have two outputs in common. Let $Q_1 = P_4$ and $Q_2 = P_5$. The relationship between the ring outputs and the key images in this transaction can be represented by the bipartite graph in Figure 2. The union of the two ring output sets forms one vertex class and the two key images form the other vertex class. An edge between an output and a key image indicates that the latter could be the true key image of that output. Note that the new outputs T_1, T_2, T_3 play no role in the construction of the bipartite graph. We will refer to such output/key image bipartite graphs as *transaction graphs*.

As each key image must have been generated from a unique output, any pair of edges (P_i, I_1) and (Q_j, I_2) such that $P_i \neq Q_j$ is a plausible candidate for the true relationship between the outputs and key images. A *matching* on a graph is a subset of the edges such that no two edges in the subset share a vertex (see Section 5.2 for a precise definition). The pair of edges (P_i, I_1) and (Q_j, I_2) with $P_i \neq Q_j$ is a matching on the graph in Figure 2. In fact, it is a matching of maximum size as any three edges in this graph would have two which meet in either I_1 or I_2.

Let us now consider a similar bipartite graph induced by the set of all transactions which have appeared up to the block having height h. The key image vertex class $\mathcal{K}_h$ in this graph is the set of all key images which have appeared on the blockchain up to block height h.

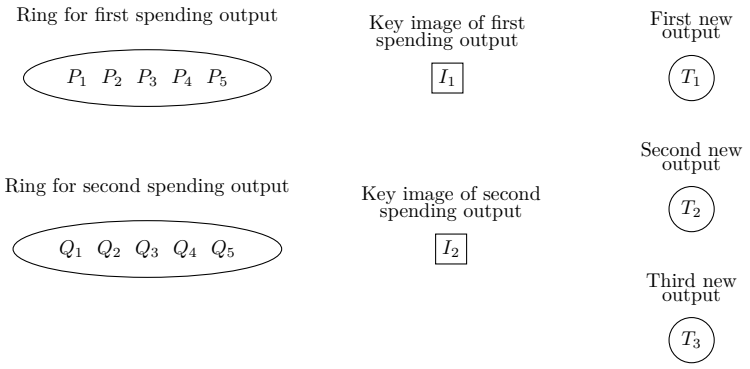

Figure 1 A CryptoNote transaction with two inputs and three outputs

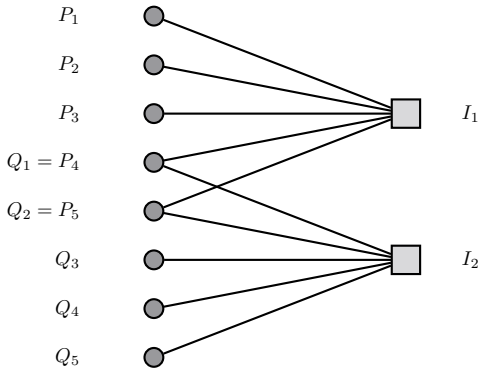

Figure 2 Transaction graph corresponding to the transaction in Figure 1

The output vertex class $\mathcal{O}_h$ is the set of all outputs which have *appeared in at least one transaction ring* in the blocks up to height h. Note that $\mathcal{O}_h$ is *not* the set of all outputs which have appeared on the blockchain in blocks up to height h. We represent the edge set of the transaction graph induced by the CryptoNote transaction rings as a subset E of $\mathcal{O}_h \times \mathcal{K}_h$. For $P \in \mathcal{O}_h$ and $I \in \mathcal{K}_h$, the edge (P, I) belongs to E if the output P appeared in the transaction ring used to create I (via the linkable ring signature).

Since each key image $I \in \mathcal{K}_h$ is generated from a unique output $P \in \mathcal{O}_h$, we have $|\mathcal{K}_h| \leq |\mathcal{O}_h|$. In a bipartite graph with vertex classes of cardinality m and n, the size of a maximum matching can be at most $\min(m, n)$. Since the edges corresponding to the *true association* between outputs and key images form a matching of size $|\mathcal{K}_h|$, the induced bipartite graph always has a maximum matching. In fact, we have the following principle which has been discussed by Monero Research Lab researchers [12] and others [25, 26].

▶ **Observation 1.** *Any maximum matching on a CryptoNote transaction graph is a plausible candidate for the ground truth, i.e. the true association between outputs and key images.*

1.3 Tracing CryptoNote Transactions

While a single linkable ring signature over a ring of public keys guarantees signer anonymity against computationally bounded adversaries [15], CryptoNote blockchains typically have signatures created using overlapping rings which can reveal the identity of the signing key. In this context, the *signing key* is the public key corresponding to the secret key which was used to generate the linkable ring signature.

▶ **Definition 2.** *A CryptoNote transaction ring is said to be* ***traceable*** *if the true signing key is correctly identified.*

To illustrate how CryptoNote transaction rings can be traced, consider three CryptoNote transaction rings having ring members $\{P_1\}$, $\{P_1, P_2\}$, and $\{P_1, P_2, P_3, P_4\}$ respectively. Let I_1, I_2, I_3 be the distinct key images created from these three transaction rings. The corresponding CryptoNote transaction graph is shown in Figure 3.

(a) The first transaction ring has only one member and is therefore trivially traceable. The key P_1 must be the signing key. Such transactions are called *zero-mixin transactions*.

(b) In isolation, either P_1 or P_2 could have been the signing key in the second transaction ring. But once P_1 has been identified as the signing key in the first transaction, we

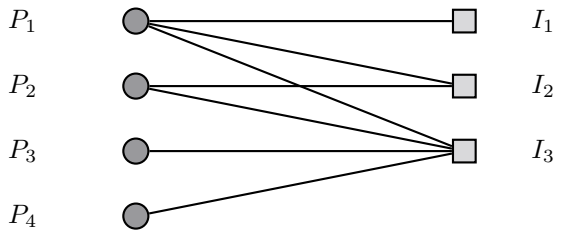

Figure 3 Transaction graph corresponding to three CryptoNote transactions

know that I_1 is the key image corresponding to P_1. Since the signing key in the second transaction ring has key image $I_2 \neq I_1$, it must be equal to P_2.

(c) Similarly, we can eliminate P_1 and P_2 from the set of possible signing keys of the third transaction ring. Any one of the remaining two ring members, P_3 and P_4, can the true signing key of this ring. There is not enough information to conclusively identify one of them as the true signing key.

In this example, the first two transaction rings are traceable and the third one is not. But the effective mixin size of the third transaction ring was reduced from 3 to 1. So the presence of traceable rings can affect the privacy of other untraceable rings.

1.4 Paper Organization

We present related work in Section 2 followed by a summary of our contributions in Section 3. The closed set attack and the clustering algorithm for approximating it are described in Section 4. In Section 5, we describe the Dulmage-Mendelsohn (DM) decomposition. In Section 6, we show that the DM decomposition finds all possible closed sets in a CryptoNote transaction graph. In Section 7, we describe the empirical results obtained by applying the DM decomposition to the Monero transaction graph with and without information from hard forks. Section 8 concludes the paper.

2 Related Work

The first traceability analyses on CryptoNote blockchains were performed by Kumar et al. [14] and Möser et al. [17]. They both studied the Monero blockchain history and found that zero-mixin transactions have a cascade effect of rendering other transactions traceable. This technique for achieving undeniable traceability is called the *cascade attack*.[1] They also considered heuristics for tracing transactions like the guess-newest heuristic and the output merging heuristic. But these methods only achieve plausible traceability and can lead to false positives.

In our survey of related work, we restrict our attention to methods for undeniable traceability. Readers interested in methods for plausible traceability can refer to [14, 17, 23, 18, 7]. A line of work describing the design of better ring samplers and sustainable ring-based anonymous systems can be found in [20, 11, 8].

The cascade attack proceeds in an iterative manner. First, it marks the outputs in zero-mixin transactions as spent. Then it marks these outputs as mixins in other (non-zero-mixin)

[1] The second transaction ring in the example of Section 1.3 was traced using the cascade attack.

transactions. If all outputs except one in a transaction ring are marked as mixins, then the remaining output is identified as the output being spent (and the transaction becomes undeniably traceable). The outputs which have been newly marked as spent in a ring are marked as mixins in other rings. The process continues until no new outputs can be marked as spent.

The initial implementation of Monero did not hide the transaction amounts. In January 2017, Monero introduced a new transaction type called *ring confidential transaction (RingCT)*, where transaction output amounts are hidden in Pedersen commitments. RingCT became mandatory in September 2017 [4].

While the cascade attack was able to trace a significant percentage of non-RingCT transactions, RingCT transactions remain immune to it. In our empirical evaluation, we found that the cascade attack could not trace any RingCT transactions up to block height 2,530,000. This was primarily because RingCT transactions did not allow zero-mixin rings.

Wijaya et al. [22] observed that a zero-mixin effect could be created in RingCT transactions by spending n times from a ring of size n. The n outputs in the ring can then be marked as mixins in other transaction rings. They name this type of spending behavior the *ring attack*. As a proof of concept, they created five outputs in Monero block 1,468,425 and then spent all of them using the other four as mixins in five transaction rings in block 1,468,439. This behavior does not arise naturally due to the mixin sampling strategy in Monero. Up to Monero block 2,530,000 (January 4, 2022), the ring of size 5 created by Wijaya et al. is the only RingCT ring which exhibits this behavior.

Yu et al. [26] defined a *closed set* to be a set of n outputs which can be represented as a union of n transaction rings. As each transaction ring must spend a unique output, the outputs in a closed set can be marked spent. They proved that the *closed set attack* (which finds all closed sets) is optimal by showing that its output is equivalent to the output of a brute-force attack. However, they observed that the naive method of finding closed sets by testing all subsets of the outputs has exponential time complexity.

As a workaround, they proposed an approximate algorithm to identify closed sets called the *clustering algorithm*. After executing the cascade attack [14, 17] on the transaction rings, the clustering algorithm attempts to find closed sets by combining transaction rings which are close to each other (see Section 4 for a more detailed description). They proved that the clustering algorithm can identify all closed sets up to size 5. While the performance of closed set attack is better than the cascade attack, they reported that no RingCT transactions were traced by their algorithm.

Several projects have forked the Monero blockchain resulting in multiple blockchains with large numbers of common outputs. When a common output is spent in two different forks, the same key image appears in both spending transactions. The real output is then contained in the intersection of the transaction rings of such transactions. Wijaya et al. [24] and Hinteregger et al. [13] used repeated key images which appeared in Monero and two hard forks, Monero Original [2] and MoneroV [3], to trace transactions in all three chains. While their methods were the first ones which successfully traced RingCT transactions in Monero, they reported that the overall impact of their techniques was small. Only a small percentage of the total set of transactions were rendered traceable. Wijaya et al. [24] also discuss strategies for mitigating the loss of anonymity due to key reuse in hard forks.

The Monero reference implementation includes a tool for identifying spent outputs using the techniques described above [6]. It implements the cascade attack, finds transactions which cause the ring attack characterized by Wijaya et al. [22], attempts to identify closed sets, and performs the cross-chain analysis proposed by Wijaya et al. [24] and Hinteregger

et al. [13]. It is included in every Monero release as an executable with the name monero-blockchain-mark-spent-outputs. It is informally called the "blackball tool" in the Monero community as the set of spent outputs represent a blacklist which should be avoided when sampling mixins. To perform cross-chain analysis, the tool takes the LMDB database files of all chains as input.

We noticed two issues with the Monero blackball tool with regard to cross-chain analysis. Firstly, the tool is not able to read the LMDB database of MoneroV [3] due to a discrepancy in the transaction formats in the main Monero code and the MoneroV code. This discrepancy did not affect our analysis as we used the JSON-RPC interface of the MoneroV client [3] to extract the transaction data. Secondly, and more seriously, the tool only uses an integer index to uniquely identify an output across chains and not the output public key.

Outputs in a single Monero chain are partitioned by their amounts (with RingCT outputs having dummy amount zero) and are assigned increasing indices in the order of their appearance on the chain. This means that outputs which appear in two different chains after a fork can have the same index even though they have different public keys. Consequently, the cross-chain analysis performed by the blackball tool has errors. To be fair, the tool outputs error messages during its execution when it encounters disjoint transaction rings for the same key image. The presence of code for generating these error messages suggests that the blackball tool developers are aware of this issue. To avoid such errors, we used the public keys of the outputs as their unique identifier in our cross-chain analysis.

3 Our Contributions

Our contributions are as follows.

1. Our main contribution is to show that the Dulmage-Mendelsohn (DM) decomposition of bipartite graphs gives an efficient implementation of the closed set attack, which is the optimal method for undeniable traceability in CryptoNote blockchains. Computing the DM decomposition involves finding a maximum matching, a depth-first search from all unmatched vertices, and a computation of strongly connected components, all on the CryptoNote transaction graph. All three algorithms have polynomial time complexity in the number of nodes and edges of the graph.
2. We implemented the DM decomposition, the cascade attack, and the closed set attack in Rust. The code is available at `https://github.com/avras/cryptonote-analysis` under an MIT license. While an open source implementation of the DM decomposition already existed in CSparse [9], it was much slower than the proprietary implementation in Matlab [1]. Our implementation of the DM decomposition has performance comparable to the Matlab implementation.
3. We compute the empirical performance of the DM decomposition method on Monero and show that it outperforms the clustering algorithm approximation to the closed set attack proposed by Yu et al. [26].
4. While previous traceability attacks have been effective against non-RingCT transactions in Monero, RingCT transactions have been mostly immune. Only cross-chain analysis which uses information from hard forks has been able to trace Monero RingCT transactions [13]. We compute the empirical performance of the DM decomposition method using information from four different hard forks: Monero Original, MoneroV, Monero v7, and Monero v9. Our results show that, even with hard fork information, Monero RingCT transactions are mostly immune to undeniable traceability via the DM decomposition method.

5. We released the scripts used to generate our empirical results in the code repository at `https://github.com/avras/cryptonote-analysis`. We prepared detailed instructions on how to reproduce our results and made them available at `https://www.respectedsir.com/cna`.

4 The Closed Set Attack and Clustering Algorithm

In a CryptoNote blockchain, let $R_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,n_i}\}$ be the ring of public keys in the ith transaction. Let $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$ be the multiset[2] of all transaction rings which have appeared on the blockchain up to height h. Since each ring has a unique key image associated with it, the set of key images $\mathcal{K}_h$ has size n.

As per the notation introduced in Section 1.2, the set of outputs that have appeared in at least one transaction ring is given by $\mathcal{O}_h = \cup_{i=1}^{n} R_i$. Let $m = |\mathcal{O}_h|$. As the number of key images cannot exceed the number of public keys, we have $m \geq n$.

4.1 Brute-Force Attack

Yu et al. [26] proposed the closed set attack and argued that it is optimal because of having identical traceability performance to the *brute-force attack*. We will use the notion of a *system of distinct representatives* [16] to describe the brute-force attack.

▶ **Definition 3.** *Let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be a multiset of subsets of a finite set S. A set of n distinct elements $\{s_1, s_2, \ldots, s_n\}$ which satisfies $s_i \in S_i$ is called a* ***system of distinct representatives (SDR)*** *for $\mathcal{S}$.*

In the context of CryptoNote blockchains, an SDR $\{P_1, P_2, \ldots, P_n\}$ for the multiset $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$ corresponds to a possible candidate for the sequence of signing keys for the rings in $\mathcal{R}_h$. This is because for each i the key P_i belongs to R_i and all the P_i's are distinct.

The brute-force attack for tracing CryptoNote transactions is shown in Algorithm 1. This attack prunes the rings in $\mathcal{R}_h$ to generate the multiset $\mathcal{R}'_h = \{R'_1, R'_2, \ldots, R'_n\}$ where each R'_i is a subset of R_i. It includes only those keys in R'_i which are potential signing keys for the ring R_i.

The time complexity of the brute-force attack is $O\left(\prod_{i=1}^{n} |R_i|\right)$ as each P_i can be independently chosen in $|R_i|$ ways. If the number of rings R_i in $\mathcal{R}_h$ with at least two keys is n_2, then $\prod_{i=1}^{n} |R_i| \geq 2^{n_2}$. Thus the brute-force attack becomes infeasible as the number of rings with at least one mixin increases.

4.2 Closed Set Attack

Yu et al. [26] define a closed set as follows.

▶ **Definition 4.** *Let $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$ be a multiset of CryptoNote transaction rings and $\mathcal{O}_h = \cup_{i=1}^{n} R_i$. A subset C of $\mathcal{O}_h$ having cardinality k is called a* ***closed set*** *of $\mathcal{R}_h$ if there exist k transaction rings $R_{i_1}, R_{i_2}, \ldots, R_{i_k}$ in $\mathcal{R}_h$ such that $C = \cup_{j=1}^{k} R_{i_j}$.*

[2] In a multiset, elements can occur more than once.

Algorithm 1 The brute-force attack for tracing CryptoNote transactions

Input : A multiset of transaction rings $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$
Output : A multiset of pruned transaction rings $\mathcal{R}'_h = \{R'_1, R'_2, \ldots, R'_n\}$ where $\emptyset \neq R'_i \subseteq R_i$ for all $i \in \{1, 2, \ldots, n\}$

// Initialize the rings in $\mathcal{R}'_h$ to the empty set;
for $i \leftarrow 1$ **to** n **do**
  $R'_i \leftarrow \emptyset$
end
// Iterate over elements of $R_1 \times R_2 \times \cdots \times R_n$ to find SDRs;
foreach $(P_1, P_2, \ldots, P_n)$ *in* $R_1 \times R_2 \times \cdots \times R_n$ **do**
  if $\{P_1, P_2, \ldots, P_n\}$ *is an SDR* **then**
    for $i \leftarrow 1$ **to** n **do**
      // Add P_i to R'_i;
      $R'_i \leftarrow R'_i \cup \{P_i\}$
    end
  end
end

▶ **Example 5.** To illustrate the definition and significance of a closed set, suppose that $\mathcal{R}_h$ contains four transaction rings of the following form.

$$R_1 = \{P_1, P_2, P_3\},$$
$$R_2 = \{P_2, P_3\},$$
$$R_3 = \{P_1, P_3\},$$
$$R_4 = \{P_1, P_2, P_3, P_4\}.$$

Exactly one public key from each ring is used to generate the linkable ring signature. If a public key was used twice, the key image would repeat and the later of the two transactions would be rejected by the miners. Since the union of the first three rings $R_1 \cup R_2 \cup R_3 = \{P_1, P_2, P_3\}$ has cardinality 3, the set $\{P_1, P_2, P_3\}$ is a closed set. Each of the keys P_1, P_2, P_3 must be the signing key in exactly one of the first three rings. This implies that none of them can be the signing key in the fourth ring R_4. So we deduce that P_4 must be the signing key of ring R_4, rendering the latter traceable. ⌟

The closed set attack for tracing CryptoNote transactions is shown in Algorithm 2. Yu et al. [26] proved that it is is optimal for undeniable traceability by showing that its output is identical to the output of the brute-force attack. We rephrase their Theorem 1 in our notation as follows.

▶ **Theorem 6** (Yu et al. [26]). *Given a multiset $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$ of CryptoNote transaction rings, let the multiset $\mathcal{R}_h^{closed}$ be the output of the closed set attack and the multiset $\mathcal{R}_h^{brute}$ be the output of the brute-force attack. Then $\mathcal{R}_h^{closed} = \mathcal{R}_h^{brute}$.*

The running time of the closed set attack is dominated by the time required to find all the closed sets in $\mathcal{R}'_h$. Note that $\mathcal{R}'_h$ is initially equal to $\mathcal{R}_h$. After a closed set is found, some of its rings may be pruned. This may cause new closed sets to become available. Hence the search needs to be performed again.

Algorithm 2 The closed set attack for tracing CryptoNote transactions

Input : A multiset of transaction rings $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$
Output : A multiset of pruned transaction rings $\mathcal{R}'_h = \{R'_1, R'_2, \ldots, R'_n\}$ where $\emptyset \neq R'_i \subseteq R_i$ for all $i \in \{1, 2, \ldots, n\}$

// Initialize the rings in $\mathcal{R}'_h$ to the rings in $\mathcal{R}_h$;
for $i \leftarrow 1$ **to** n **do**
  $R'_i \leftarrow R_i$
end
// Iterate over all the closed sets;
foreach *closed set* $C = \cup_{j=1}^{k} R'_{i_j}$ **do**
  for $i \leftarrow 1$ **to** n **do**
    // Check that the ring does not generate the closed set;
    if $i \notin \{i_1, i_2, \ldots, i_k\}$ **then**
      // Remove elements of C from R'_i;
      $R'_i \leftarrow R'_i \cap C^c$
    end
  end
end

The naive algorithm [26, Appendix A] for finding closed sets by considering all subsets of $\mathcal{R}'_h$ becomes infeasible as the size of $|\mathcal{R}_h|$ increases. Instead, Yu et al. [26] proposed the clustering algorithm as an approximation to the naive algorithm.

4.3 Clustering Algorithm

▶ **Definition 7.** *A subset of $\mathcal{R}_h$ is called a **cluster**.*

A cluster consists of a set of rings from $\mathcal{R}_h$. The distance of a ring R from a cluster is defined as follows.

▶ **Definition 8.** *Let $\mathcal{C} = \{R_{i_1}, R_{i_2}, \ldots\}$ be a cluster and let $pk(\mathcal{C}) = \cup_{R' \in \mathcal{C}} R'$ be the set of keys in it. The **distance** of a ring $R \in \mathcal{R}_h$ from a cluster $\mathcal{C}$ is defined as*

$$d(R, \mathcal{C}) = |R| - |pk(\mathcal{C}) \cap R|.$$

▶ **Example 9.** Consider the rings $R_1, \ldots, R_4$ from Example 5. Suppose we consider the cluster $\mathcal{C} = \{R_1, R_2\}$. Then we have $pk(\mathcal{C}) = \{P_1, P_2, P_3\}$, $d(R_3, \mathcal{C}) = 0$, and $d(R_4, \mathcal{C}) = 1$. ⌟

Yu et al. [26] use a cluster formation algorithm as a subroutine in their clustering algorithm. This algorithm forms a cluster by starting from any transaction ring in $\mathcal{R}_h$ and adding other rings which are at a distance of at most 1 from the running cluster (see Algorithm 3).

The clustering algorithm for finding closed sets is shown in Algorithm 4 where `CascadeAttack` is a procedure that implements the cascade attack of [14, 17]. `CascadeAttack` takes a multiset of transactions rings as input and outputs a pruned multiset after removing keys from each ring that have been identified as mixins by the cascade attack.

The clustering algorithm may fail to find certain closed sets because a ring that is needed to form a closed set may be at a distance of 2 or more from the current cluster in Algorithm 3. We observed this in our empirical analysis of the Monero transaction graph where the clustering algorithm failed to find some closed sets that were found by the DM decomposition.

Algorithm 3 ClusterForm: An algorithm for constructing a cluster

Input : A multiset of transaction rings $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$ and a specific ring $R \in \mathcal{R}_h$
Output : A cluster $\mathcal{C}$ containing R

// Initialize $\mathcal{C}$ to the set containing R;
$\mathcal{C} \leftarrow \{R\}$

// Iterate over all the rings in $\mathcal{R}_h$ not equal to R;
foreach $R' \in \mathcal{R}_h \setminus R$ **do**
  // Check if R' is within a distance of 1 from $\mathcal{C}$;
  if $d(R', \mathcal{C}) \leq 1$ **then**
    // Add R' to $\mathcal{C}$;
    $\mathcal{C} \leftarrow \mathcal{C} \cup R'$
  end
end

5 The Dulmage-Mendelsohn Decomposition

Consistent with notation used by Dulmage and Mendelsohn [10], we define an undirected bipartite graph K as a triple (S, T, E) where S and T are non-empty sets representing vertex classes and $E \subseteq S \times T$ represents the edge set. So an edge in K is given by an ordered pair (s, t) where $s \in S$ and $t \in T$. The ordering of the vertices in the edge (s, t) is simply a consequence of putting S before T in the triple (S, T, E), and does not imply directivity. We say that an edge (s, t) belongs to the graph K, written as $(s, t) \in K$, to mean that $(s, t) \in E$. We only consider bipartite graphs K where both S and T are finite sets.

5.1 Minimum Covers of Bipartite Graphs

▶ **Definition 10.** *Let $K = (S, T, E)$ be a bipartite graph. Let A and B be subsets of S and T respectively. A pair of such sets (A, B) is called a **vertex cover** for a bipartite graph K if for each edge $(s, t) \in K$, either $s \in A$ or $t \in B$ (both conditions can also hold).*

▶ **Definition 11.** *The **size** of a vertex cover (A, B) is defined as $|A| + |B|$ where $|X|$ denotes the cardinality of a set X.*

Since S and T are assumed to be finite sets, every vertex cover of K will have a finite size.

▶ **Definition 12.** *The **cover number** of a bipartite graph K is the minimum of $|A| + |B|$ over all vertex covers (A, B) of K.*

▶ **Definition 13.** *A vertex cover (A, B) of a bipartite graph K whose size equals the cover number of K is called a **minimum cover**.*

The following two results were proved by Dulmage and Mendelsohn [10].

▶ **Lemma 14.** *If (A_1, B_1) and (A_2, B_2) are minimum covers of a bipartite graph K having finite cover number, then $(A_1 \cap A_2, B_1 \cup B_2)$ and $(A_1 \cup A_2, B_1 \cap B_2)$ are both minimum covers of K.*

▶ **Lemma 15.** *Let (A_1, B_1) and (A_2, B_2) be minimum covers of a bipartite graph K having finite cover number. If $A_1 \subseteq A_2$, then $B_1 \supseteq B_2$.*

Algorithm 4 The clustering algorithm

Input : A multiset of transaction rings $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$
Output : A multiset of pruned transaction rings $\mathcal{R}'_h = \{R'_1, R'_2, \ldots, R'_n\}$ where $\emptyset \neq R'_i \subseteq R_i$ for all $i \in \{1, 2, \ldots, n\}$

// Run the cascade attack on $\mathcal{R}_h$;
$\mathcal{R}'_h \leftarrow$ `CascadeAttack(`$\mathcal{R}_h$`)`;
// Set flag to true;
flag $\leftarrow$ true;

while flag *is true* **do**
  flag $\leftarrow$ false;
  // Iterate over all the rings in $\mathcal{R}'_h$;
  foreach $R' \in \mathcal{R}'_h$ **do**
    // Form a cluster starting from R' using Algorithm 3;
    $\mathcal{C}' = \{R'_{i_1}, R'_{i_2}, \ldots, R'_{i_k}\} \leftarrow$ `ClusterForm(`R'`)`;
    if $C = \cup_{j=1}^{k} R'_{i_j}$ *is a closed set* **then**
      for $i \leftarrow 1$ **to** n **do**
        // Check that the ring does not generate the closed set C;
        if $i \notin \{i_1, i_2, \ldots, i_k\}$ **then**
          // Check if C and R'_i have elements in common;
          if $C \cap R'_i \neq \emptyset$ **then**
            // Remove elements of C from R'_i;
            $R'_i \leftarrow R'_i \cap C^c$;
            // Set the flag to indicate modification of transaction rings;
            flag $\leftarrow$ true;
          end
        end
      end
      if flag *is true* **then**
        // Run the cascade attack on $\mathcal{R}'_h$;
        $\mathcal{R}'_h \leftarrow$ `CascadeAttack(`$\mathcal{R}'_h$`)`;
        // Run the cascade attack on the cluster $\mathcal{C}'$;
        $\mathcal{C}'' = \{R''_{i_1}, R''_{i_2}, \ldots, R''_{i_k}\} \leftarrow$ `CascadeAttack(`$\mathcal{R}'_h$`)`;
        // Replace the rings in $\mathcal{R}'_h$ with intra-cluster cascade attack results;
        for $j \leftarrow 1$ **to** k **do**
          $R'_{i_j} \leftarrow R''_{i_j}$;
        end
      end
    end
  end
end

Setting $A_1 = A_2$ in the above lemma gives us the following corollary.

▶ **Corollary 16.** *If (A, B_1) and (A, B_2) are both minimum covers of a bipartite graph K having finite cover number, then $B_1 = B_2$.*

For a bipartite graph K, let $\mathcal{C}$ be the set of all minimum covers. Let us define the following sets obtained by taking intersections and unions of the components of the minimum covers.

$$A_* = \bigcap_{(A,B)\in\mathcal{C}} A, \qquad A^* = \bigcup_{(A,B)\in\mathcal{C}} A, \tag{1}$$

$$B_* = \bigcap_{(A,B)\in\mathcal{C}} B, \qquad B^* = \bigcup_{(A,B)\in\mathcal{C}} B. \tag{2}$$

By Lemma 14, if K has a finite cover number then the pairs (A_*, B^*) and (A^*, B_*) are both minimum covers of K.

▶ **Example 17.** Reconsider the bipartite graph shown in Figure 3 with vertex classes $S = \{P_1, P_2, P_3, P_4\}$ and $T = \{I_1, I_2, I_3\}$. Since $(\emptyset, T)$ is a minimum cover the graph, $A_* = \emptyset$ and $B^* = T$. As $(\{P_1\}, \{I_2, I_3\})$ and $(\{P_1, P_2\}, \{I_3\})$ are the only other minimum covers, $A^* = \{P_1, P_2\}$ and $B_* = \{I_3\}$.

5.2 Maximum Matchings on Bipartite Graphs

In a graph, we say that edges (s, t) and (s', t') share a vertex if either $s = s'$ or $t = t'$.

▶ **Definition 18.** *A **matching** on a bipartite graph $K = (S, T, E)$ is a subset M of the edge set E such that no two edges in M share a vertex. The cardinality $|M|$ is called the **order** of the matching M.*

▶ **Definition 19.** *A **maximum matching** on a bipartite graph K is a matching on K of maximum order.*

The following definition classifies edges according to their membership in maximum matchings on K.

▶ **Definition 20.** *An edge (s, t) of a bipartite graph K is said to be **admissible** if there exists a maximum matching M on K such that $(s, t) \in M$. An edge which is not admissible is said to be **inadmissible**.*

The following result by König [16] says that maximum matchings have the same size as minimum covers in bipartite graphs. We will need it in the proof of Theorem 26.

▶ **Proposition 21.** *The cover number of a finite bipartite graph equals the order of maximum matchings on the graph.*

5.3 Definition of the DM Decomposition

With the above definitions in place, we are ready to describe the DM decomposition.

▶ **Definition 22.** *Let $K = (S, T, E)$ be a bipartite graph having a finite cover number. The **Dulmage-Mendelsohn decomposition** of K is a partition of $S \times T$ into three disjoint sets R_1, R_2, R_3 which satisfy the following properties:*

1. *The set of admissible edges in K equals $E \cap R_1$.*
2. *The set of inadmissible edges in K equals $E \cap R_2$.*
3. *$E \cap R_3 = \emptyset$.*

The fine structure of the sets R_1, R_2, R_3 depends on the minimum covers of K. Let us consider two cases.

5.3.1 Case 1: $A_* = A^*$.

If $A_* = A^*$, then the graph K has only one minimum cover given by $(A_*, B^*) = (A^*, B_*)$. In this case, the following result holds.

▶ **Proposition 23.** *Let $K = (S, T, E)$ be a bipartite graph having a finite cover number. If K has only one minimum cover given by (A_*, B^*), then the Dulmage-Mendelsohn decomposition of K is the partition of $S \times T$ into the sets R_1, R_2, R_3 given by*

$$\begin{aligned} R_1 &= (A_* \times (B^*)^c) \bigcup ((A_*)^c \times B^*), \\ R_2 &= A_* \times B^*, \\ R_3 &= (A_*)^c \times (B^*)^c. \end{aligned} \tag{3}$$

5.3.2 Case 2: $A_* \neq A^*$.

Now suppose $A_* \neq A^*$. By definition, $A_* \subseteq A^*$. So A_* must be a proper subset of A^*. Then there exists at least one non-empty set $X \subset S$ such that $A_* \cap X = \emptyset$ and $(A_* \cup X, Y)$ is a minimum cover of K for some $Y \subset T$. The existence of such a set follows from the fact $A^* \setminus A_*$ is a candidate for X. Let S_1 be a set of smallest cardinality among all candidates for X. There may be many possibilities for S_1, all having the same smallest cardinality. We can pick any one of them.

Let (A_1, B_1) be a minimum cover with $A_1 = A_* \cup S_1$. By Corollary 16, B_1 is uniquely determined by A_1. As $A_* \subseteq A_1$, Theorem 15 tells us that $B_1 \subseteq B^*$. As all minimum covers of K have the same size, we have $|A_*|+|B^*| = |A_1|+|B_1|$. Since $|A_1| > |A_*|$, we have $|B_1| < |B^*|$. Thus B_1 is a proper subset of B^*. Let $T_1 = B^* \setminus B_1$. Since $|A_1| - |A_*| = |B^*| - |B_1|$, we have $|S_1| = |T_1|$.

If $A_1 = A^*$, the process stops. Otherwise, there exists at least one non-empty set $X \subset S$ such that $A_1 \cap X = \emptyset$ and $A_1 \cup X$ is the first component of a minimum cover of K. Let S_2 be a set of smallest cardinality among all candidates for X. Let (A_2, B_2) be a minimum cover with $A_2 = A_1 \cup S_2 = A_* \cup S_1 \cup S_2$. As before, B_2 is uniquely determined by A_2 and $B_2 \subset B_1$. Let $T_2 = B_1 \setminus B_2$. Since $|A_2| - |A_1| = |B_1| - |B_2|$, we have $|S_2| = |T_2|$. Since $B^* = T_1 \cup B_1$ and $T_2 = B_1 \setminus B_2$, we have $B^* = T_1 \cup T_2 \cup B_2$.

If we proceed in this manner, the process will stop for some k where

$$A_* \cup S_1 \cup S_2 \ldots \cup S_k = A^*. \tag{4}$$

At this point, (A^*, B_*) will be the resulting minimum cover. Furthermore, the T_i's satisfy

$$B^* = T_1 \cup T_2 \cup \ldots T_k \cup B_*. \tag{5}$$

In the intermediate stages of this process, (A_i, B_i) is a minimum cover for K for each $i \in \{1, 2, \ldots, k\}$ where

$$A_i = A_* \cup S_1 \cup S_2 \cup \ldots \cup S_i, \tag{6}$$

$$B_i = T_{i+1} \cup T_{i+2} \cup \ldots \cup T_k \cup B_*. \tag{7}$$

Equations (4) and (5) give the following decompositions of the vertex classes S and T.

$$S = A^* \bigcup (A^*)^c = A_* \cup S_1 \cup S_2 \ldots \cup S_k \bigcup (A^*)^c, \tag{8}$$

$$T = (B^*)^c \bigcup B^* = (B^*)^c \bigcup T_1 \cup T_2 \ldots \cup T_k \cup B_*. \tag{9}$$

The $k+2$ sets in the unions on the extreme right of both the above equations form a partition of S and T respectively. These partitions are unique except for a permutation of the S_i's having same cardinality, with the T_i's appropriately permuted.

With these definitions in place, we have the following result from [10].

▶ **Proposition 24.** *Let $K = (S, T, E)$ be a bipartite graph having a finite cover number. Then the Dulmage-Mendelsohn decomposition of K is given by the partition of $S \times T$ into the sets R_1, R_2, R_3 given by*

$$R_1 = (A_* \times (B^*)^c) \bigcup (S_1 \times T_1) \bigcup \ldots \bigcup (S_k \times T_k) \bigcup ((A^*)^c \times B_*)\,, \tag{10}$$

$$R_2 = (A_* \times B^*) \bigcup (A^* \times B_*) \bigcup_{i<j} (S_i \times T_j)\,, \tag{11}$$

$$R_3 = ((A_*)^c \times (B^*)^c) \bigcup ((A^*)^c \times (B_*)^c) \bigcup_{i>j} (S_i \times T_j)\,. \tag{12}$$

▶ **Example 25.** Consider the bipartite graph in Figure 3 with vertex classes $S = \{P_1, P_2, P_3, P_4\}$ and $T = \{I_1, I_2, I_3\}$.

(i) As we noted in Example 17, $A_* = \emptyset, B^* = T$ and $A^* = \{P_1, P_2\}, B_* = \{I_3\}$.

(ii) As $(\{P_1\}, \{I_2, I_3\})$ is the only candidate for (A_1, B_1), we have $S_1 = \{P_1\}$ and $T_1 = \{I_1\}$.

(iii) As $(\{P_1, P_2\}, \{I_3\})$ is the only candidate for (A_2, B_2), we have $S_2 = \{P_2\}$ and $T_2 = \{I_2\}$.

The DM decomposition is given by

$$R_1 = \{(P_1, I_1), (P_2, I_2), (P_3, I_3), (P_4, I_3)\}\,,$$
$$R_2 = \{(P_1, I_3), (P_2, I_3), (P_1, I_2)\}\,,$$
$$R_3 = \{(P_3, I_1), (P_3, I_2), (P_4, I_1), (P_4, I_2), (P_2, I_1)\}$$

The graph has no edges in R_3. The edges in R_2 cannot appear in any maximum matching, as P_1 must be matched to I_1 and P_2 must be matched to I_2. The edges in R_1 appear in at least one maximum matching on the graph. ⌟

To visualize the DM decomposition, suppose that the vertices in S are ordered according to the partition in Equation (8), i.e. the vertices in A_* appear first, followed by vertices in $S_1, S_2, \ldots, S_k$, and $(A^*)^c$. Similarly, suppose that the vertices in T are ordered according to the partition in Equation (9). Then the DM decomposition can be represented by Figure 4, where the rows correspond to vertices in T and the columns correspond to vertices in S. The admissible edges lie in blocks along the diagonal, the inadmissible edges lie above these blocks, and there are no edges below these blocks.

5.4 Computing the DM Decomposition

The DM decomposition of a bipartite graph K can be computed by finding a maximum matching M on K, then finding subsets of vertex classes unreachable from M via alternating paths, and finally by finding strongly connected components of the subgraph induced by the unreachable vertices (see [19] for details). Surprisingly, the DM decomposition is independent of the particular maximum matching chosen in the first step [19]. The component algorithms of the DM decomposition computation have worst-case running times which are polynomial in the number of graph vertices and edges.

We implemented the DM decomposition in Rust. Our code is available at `https://github.com/avras/cryptonote-analysis` under an MIT license. While an open source

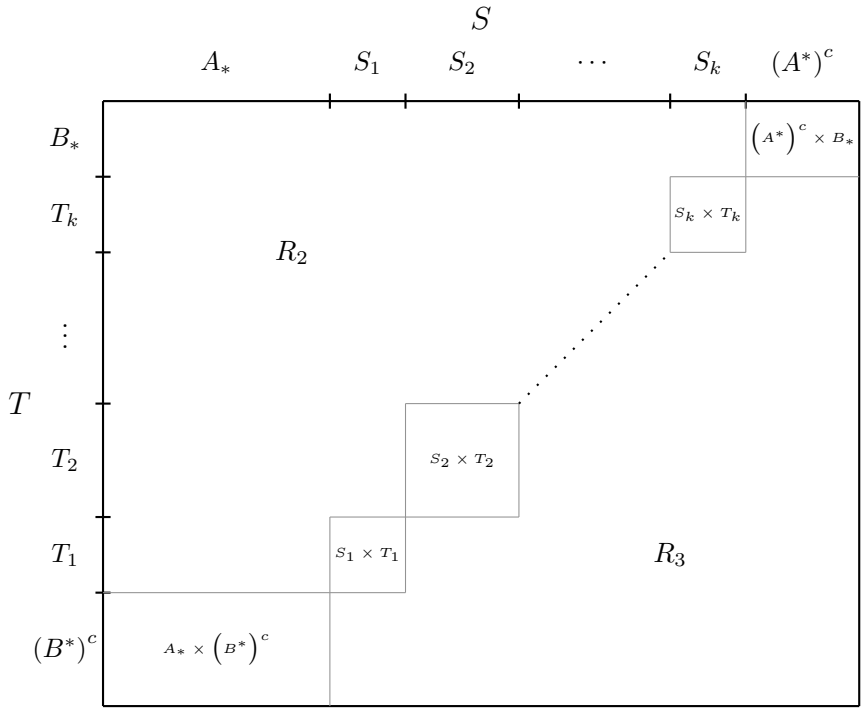

Figure 4 Visualization of the DM decomposition of a bipartite graph

implementation of the DM decomposition already existed in CSparse [9], it was much slower than the proprietary implementation in Matlab [1]. Our implementation of the DM decomposition has performance comparable to the Matlab implementation. Instructions on preparing the input data for our implementation are available at `https://www.respectedsir.com/cna`.

6 The DM Decomposition Finds All Closed Sets

By Theorem 6, the closed set attack is an optimal method for performing undeniable traceability analysis on CryptoNote transaction graphs. However, the naive method of finding closed sets by checking all subsets of the transaction rings [26, Appendix A] is not computationally feasible. The clustering algorithm for finding closed sets is guaranteed to find all closed sets of size 5 or less [26, Theorem 2]. While it does find closed sets with size more than 5, it is not guaranteed to find all closed sets.

In this section, we show that the DM decomposition finds all the closed sets in a CryptoNote transaction graph. As the DM decomposition can be computed in polynomial time, we obtain an efficient method to achieve the best possible undeniable traceability performance.

Let us establish/recall the following notation.

(i) Let $\mathcal{R}_h = \{R_1, R_2, \ldots, R_n\}$ be the multiset of all transaction rings which have appeared on the blockchain up to height h.

(ii) Let I_i be the key image corresponding to ring R_i.

(iii) Let $\mathcal{K}_h = \{I_1, I_2, \ldots, I_n\}$ be the set of all key images that have appeared on the blockchain up to height h.

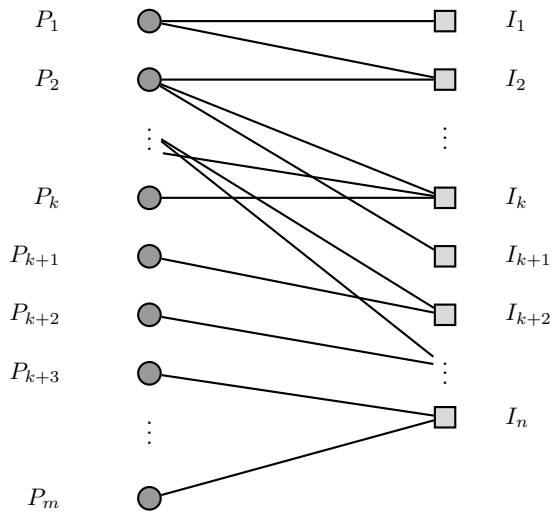

Figure 5 Transaction graph used in the proof of Theorem 26

(iv) Let $\mathcal{O}_h = \cup_{i=1}^{n} R_i = \{P_1, P_2, \ldots, P_m\}$ be the set of all keys (outputs) that have appeared in at least one transaction ring.

(v) Let $E \subseteq \mathcal{O}_h \times \mathcal{K}_h$ be the edge set of the transaction graph induced by the rings in $\mathcal{R}_h$. For $P \in \mathcal{O}_h$ and $I \in \mathcal{K}_h$, the edge (P, I) belongs to E if the output P appeared in the transaction ring used to create I.

(vi) Let C be a closed set of $\mathcal{R}_h$ having cardinality k. By Definition 4, there exist k transaction rings $R_{i_1}, R_{i_2}, \ldots, R_{i_k}$ in $\mathcal{R}_h$ such that $C = \cup_{j=1}^{k} R_{i_j}$. To describe this scenario briefly, we will say that the rings $R_{i_1}, R_{i_2}, \ldots, R_{i_k}$ *constitute* the closed set C. Furthermore, there exist k keys $P_{i_1}, \ldots, P_{i_k}$ in $\mathcal{O}_h$ such that $C = \{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\}$.

(vii) Let $I_C = \{I_{i_1}, I_{i_2}, \ldots, I_{i_k}\}$ be the set of key images corresponding to the rings $R_{i_1}, R_{i_2}, \ldots, R_{i_k}$ that constitute C.

▶ **Theorem 26.** *Let $\mathcal{G}_h = (\mathcal{O}_h, \mathcal{K}_h, E)$ be the CryptoNote transaction graph induced by the rings in $\mathcal{R}_h$ and their corresponding key images. Let C be a closed set of $\mathcal{R}_h$ and let I_C be the set of key images corresponding to the rings that constitute C. Then $(C, \mathcal{K}_h \setminus I_C)$ is a minimum cover of $\mathcal{G}_h$. Conversely, if (A, B) is a minimum cover of $\mathcal{G}_h$ where $A \neq \emptyset$, then A is a closed set.*

Proof. Without loss of generality, we can assume that $R_1, R_2, \ldots, R_k$ are the rings that constitute C and $I_1, I_2, \ldots, I_k$ are the key images corresponding to these rings. Furthermore, let $C = \cup_{i=1}^{k} R_i = \{P_1, P_2, \ldots, P_k\}$.[3] By definition, $I_C = \{I_1, I_2, \ldots, I_k\}$.

Suppose we draw the bipartite graph induced by the blockchain history by listing $P_1, \ldots, P_k$ and $I_1, \ldots, I_k$ before the other vertices on each side. Let $P_{k+1}, \ldots, P_m$ be the other outputs in $\mathcal{O}_h$. Let $I_{k+1}, \ldots, I_n$ be the other key images in $\mathcal{K}_h$. Figure 5 illustrates this bipartite graph.

Since each key image in $I_1, I_2, \ldots, I_n$ corresponds to a unique true output on the left hand side, there exists a maximum matching of order n on this graph. By Proposition 21,

[3] In general, the rings that constitute C and the corresponding keys and key images may not have indices $1, 2, \ldots, k$. But we can always relabel the elements in these sets to satisfy our assumption.

minimum covers of this graph will also have size n. Note that every edge in the graph is incident on some element in $\{I_1, I_2, \ldots, I_n\}$. Thus $(\emptyset, \mathcal{K}_h) = (\emptyset, \{I_1, \ldots, I_n\})$ is a minimum cover of the graph.

We claim that there are no edges between the key images $I_1, \ldots, I_k$ and the outputs $P_{k+1}, P_{k+2}, \ldots, P_m$. To see this, suppose there is an edge from I_j to P_l for some $j \in \{1, 2, \ldots, k\}$ and $l \in \{k+1, \ldots, m\}$. Then P_l must belong to the ring R_j as it is the only ring which contributes edges incident on I_j. This would mean P_l belongs to $\cup_{i=1}^{k} R_i = \{P_1, P_2, \ldots, P_k\}$, which is a contradiction as $l \geq k+1$. So all the edges incident on $I_1, \ldots, I_k$ must have an output from $P_1, \ldots, P_k$ on the other end.

The above argument shows that $(C, \mathcal{K}_h \setminus I_C) = (\{P_1, \ldots, P_k\}, \{I_{k+1}, \ldots, I_n\})$ is a minimum cover of the graph. Thus every closed set of $\mathcal{R}_h$ is the first member of a minimum cover of the transaction graph.

To prove the other direction, suppose that (A, B) is a minimum cover of the transaction graph where $A \neq \emptyset$. Let $B^c = \mathcal{K}_h \setminus B$ be the set of key images not in B. Suppose $B^c = \{I_{i_1}, I_{i_2}, \ldots, I_{i_l}\}$.

Since $(\emptyset, \mathcal{K}_h)$ is a minimum cover of the graph, every minimum cover must have size n. This implies that $|A| + |B| = n$. As $l = |B^c| = n - |B|$, the set A must have l outputs.

Since (A, B) is a cover of the transaction graph, every edge incident on key images in B^c must be covered by an output in A (as B can only cover edges incident on the key images in it). Each key image I_{i_j} in B^c is associated with a unique transaction ring R_{i_j} which contains the true output corresponding to it. The ring R_{i_j} is the set of outputs adjacent to I_{i_j} in the graph. We claim that R_{i_j} is a subset of A.

We prove our claim by contradiction. If there is a key P in R_{i_j} that is not contained in A, then the edge (P, I_{i_j}) exists but $P \notin A$ and $I_{i_j} \notin B$. This contradicts our assumption that (A, B) is a vertex cover.

Since the transaction ring R_{i_j} is a subset of A for every $I_{i_j} \in B^c$, we have $\cup_{j=1}^{l} R_{i_j} \subseteq A$. Furthermore, $\left|\cup_{j=1}^{l} R_{i_j}\right| \geq l$ because each of the l key images $I_{i_1}, I_{i_2}, \ldots, I_{i_l}$ has a unique true output in $\cup_{j=1}^{l} R_{i_j}$. Putting all this together, we have

$$l \leq \left| \bigcup_{j=1}^{l} R_{i_j} \right| \leq |A| = l. \tag{13}$$

We conclude that $A = \cup_{j=1}^{l} R_{i_j}$ and that $\left|\cup_{j=1}^{l} R_{i_j}\right| = l$, which proves that A is a closed set of $\mathcal{R}_h$. ◀

The above theorem says that finding all closed sets of a CryptoNote transaction graph is equivalent to finding all minimum covers of it. When the graph has only one minimum cover (i.e. when $A_* = A^*$), there is only one possible closed set. For the case when $A_* \neq A^*$, Dulmage and Mendelsohn proved the following theorem [10, Theorem 10] which shows that all the possible minimum covers of a bipartite graph can be calculated from the DM decomposition.

▶ **Theorem 27.** *Let $K = (S, T, E)$ be a bipartite graph having a finite cover number and more than one minimum cover. Let $A_*, B_*, S_1, S_2, \ldots, S_k, T_1, T_2, \ldots, T_k$ be as defined in Section 5.3.2. Let (A, B) be a minimum cover of K. Then exists a subset Λ of the index set $\{1, 2, \ldots, k\}$ such that*

$$A = \left(\bigcup_{i \in \Lambda} S_i \right) \cup A_*, \quad B = \left(\bigcup_{i \in \Lambda^c} T_i \right) \cup B_*.$$

This theorem implies that every minimum cover of the transaction graph can be recovered from the DM decomposition of the graph. This in turn implies that every closed set in the transaction graph can be found from the DM decomposition.

7 DM Decomposition of the Monero Transaction Graph

We implemented the DM decomposition, the cascade attack, and the closed set attack in Rust. Our code is available at `https://github.com/avras/cryptonote-analysis` under an MIT license.

7.1 Empirical Analysis without Hard Fork Information

To evaluate the effectiveness of the DM decomposition in tracing transaction rings, we used the results obtained by the clustering algorithm of Yu et al. [26] on Monero as the benchmark. The latter results are the best results on Monero undeniable traceability which do not use information from hard forks.

Yu et al. considered Monero transactions contained in blocks with height up to 1,541,236 (March 30, 2018). This data set contains 23,164,745 transaction rings (each one contributing a key image) and 25,126,033 outputs. The corresponding bipartite graph has 58,791,856 edges. Out of the 23,164,745 transaction rings in the data set, 4,330,234 were RingCT rings and the remaining 18,834,511 were pre-RingCT rings.

Previous work [14], [17], [26], on Monero traceability has shown that RingCT transactions in Monero are immune to undeniable traceability attacks. The same observation holds for the DM decomposition approach. None of the 4,330,234 RingCT rings could be traced by the DM decomposition (when information from hard forks is not used). Table 1 compares the number of pre-RingCT transaction rings traced by the clustering algorithm and the DM decomposition. Each row in the table gives results for transaction rings which have a certain number of mixin outputs. The results for all transaction rings with 10 or more mixin outputs are combined in the row with label "≥ 10".

All the 16,335,308 rings traced by the DM decomposition are associated with a set S_i with $|S_i| = 1$. The singleton set T_i corresponding to S_i has the key image of the output in S_i. As seen from the last row, the DM decomposition identifies 341 more traceable rings than the clustering algorithm. These new rings are only among the transaction rings having 2, 3, or 4 mixins.

Yu et al. report finding 3017 closed sets with sizes in the range 2 to 55. The DM decomposition is able to find 3045 closed sets with 3041 of them having sizes in the range 2 to 55. The remaining four closed sets have sizes 103, 106, 119, and 122. This discrepancy is due to the approximate nature of the clustering algorithm used by Yu et al. to find closed sets.

To check if the transactions which have appeared after block 1,541,236 have affected the traceability of RingCT rings, we computed the DM decomposition of the subgraph induced exclusively by RingCT transaction rings in all blocks up to height 2,530,000 (January 4, 2022).[4] This subgraph has 40,351,733 key images and 45,805,726 outputs with 409,626,277 edges between them. Let $\mathcal{K}$ be the set of all the key images in this subgraph. Its DM decomposition

[4] We could not try later block heights as the resulting transaction graphs were too large to fit in the memory of our test machine.

No. of mixins	No. of pre-RingCT rings	Traced by clustering algorithm	Traced by DM decomposition
0	12,209,675	12,209,675	12,209,675
1	707,786	625,641	625,641
2	2,941,525	1,779,134	1,779,446
3	1,345,574	952,855	952,862
4	972,457	451,959	451,981
5	143,793	74,186	74,186
6	366,894	202,360	202,360
7	12,361	4,296	4,296
8	9,148	3,506	3,506
9	6,396	2,178	2,178
≥ 10	118,902	29,177	29,177
Total	18,834,511	16,334,967	16,335,308

Table 1 Monero traceability of pre-RingCT rings by the clustering algorithm vs DM decomposition (up to block 1,541,236)

revealed only two minimum covers, $(\emptyset, \mathcal{K})$ and $(S_1, \mathcal{K} \setminus T_1)$ where $|S_1| = |T_1| = 5$. The set S_1 consists of RingCT outputs with indices 3890287, 3890288, 3890289, 3890290, and 3890291.

These five outputs were created by Wijaya et al. [22] in block 1,468,425. All of them were spent using the other four as mixins in five transaction rings in block 1,468,439 (Dec 17, 2017), to demonstrate that a set of outputs can be considered spent without relying on zero-mixin transactions. These five outputs are also marked as spent by the Monero blackball tool [6]. Thus, the DM decomposition of the Monero RingCT subgraph (using only main chain data) does not identify any new outputs as spent.

There were 37,038,237 RingCT transaction rings in the blocks with heights from 1,468,426 to 2,530,000. The five spent RingCT outputs were chosen as mixins in only 25 of these RingCT rings. Each of the 25 rings had at least 4 mixins and had their effective number of mixins reduced by one. Thus, the RingCT rings are mostly unaffected by the DM decomposition analysis.

The clustering algorithm was also able to identify the size 5 closed set. But it took 64 hours to finish running on our test machine while the DM decomposition could be computed in 4 hours.

7.2 Empirical Analysis using Hard Fork Information

To check the immunity of Monero RingCT transactions against the DM decomposition technique which incorporates hard fork information, we constructed a transaction graph using four different hard forks: Monero Original, MoneroV, Monero v7, and Monero v9. Table 2 gives the information regarding these forks where the last column contains the number of RingCT key images which appeared both in fork chain and the Monero main chain up to block height 2,530,000. While Monero Original and MoneroV were intentional hard forks created by developers who preferred a different design, the blocks on the Monero v7 and Monero v9 forks were unintentionally created by miners who were late in upgrading to the latest version of the main Monero client. This is the reason for the small number of blocks in the Monero v7 and Monero v9 forks.

We computed the performance of the DM decomposition technique on this graph up to

Fork Name	Fork block	Number of blocks in fork	Number of common key images	Number of common RingCT key images
Monero Original	1,546,000	238,682	86,685	64,189
MoneroV	1,564,966	146,325	9,387	6,609
Monero v7	1,685,555	29	1,061	1,027
Monero v9	1,788,000	73	1,581	1,581

Table 2 Information about the four Monero hard forks

Monero block height 2,530,000 (January 4, 2022). We found that 63,060 RingCT transaction rings out of 40,351,733 are undeniably traceable, i.e. only 0.15% of the RingCT rings are undeniably traceable. Note that the number of traceable RingCT rings is less than the total number of common RingCT key images shown in Table 2. This is because the appearance of key image in both the main chain and the fork chain does not imply traceability. If the transaction rings in both cases have more than one output in common, the true output being spent may not be identified.

The clustering algorithm was also able to trace the same 63,060 RingCT transaction rings. But it took 64 hours to finish while the DM decomposition took 4 hours. In fact, the cascade attack, which is the first step in the clustering algorithm (see Algorithm 4), was able to trace all these rings. The subsequent closed set search was fruitless as there were no closed sets in the transaction graph, except for the size 5 closed set induced by Wijaya et al. [22].

Readers interested in the effect of the DM decomposition analysis (using hard forks) on non-RingCT rings up to block height 2,530,000 can read the section at `https://www.respectedsir.com/cna/hardfork-nonringct.html` in our documentation.

8 Conclusion

We showed that the classical notion of the Dulmage-Mendelsohn decomposition of bipartite graphs gives an efficient implementation of the closed set attack, which is the optimal method for undeniable traceability in CryptoNote blockchains. Combining the DM decomposition with previously proposed methods for plausible traceability is an interesting direction for future work. We have released open source implementations of the DM decomposition, cascade attack, and clustering algorithm. We have also released the scripts used to generate all the empirical results in this paper along with detailed instructions on how to use them. We hope that these tools will be useful to other researchers, especially those working on methods for plausible traceability.

Acknowledgments

We thank Justin Ehrenhofer for sharing the blockchain databases of the (no longer operational) Monero Original, MoneroV, Monero v7, and Monero v9 forks [5]. We also thank him for his feedback on an earlier version of this paper, which helped improve the presentation of the empirical results. We thank Zuoxia Yu for sharing the full version of their FC 2019 paper. Finally, we thank the anonymous reviewers of PoPETs 2022 (where an earlier version of this paper was eventually rejected) and of the current conference for their comments. We prepared the instructions for reproducing our empirical results on the suggestion of a PoPETs 2022 reviewer.

References

1 dmperm: MATLAB function for Dulmage-Mendelsohn decomposition. URL: https://in.mathworks.com/help/matlab/ref/dmperm.html.

2 Monero Original GitHub Repository, 2018. URL: https://github.com/XmanXU/monero-original.

3 MoneroV GitHub Repository, 2019. URL: https://github.com/monerov/monerov.

4 Monero Scheduled Software Upgrades, 2020. Last Accessed: August 25, 2021. URL: https://github.com/monero-project/monero/#scheduled-software-upgrades.

5 Monero Blackball Databases, 2021. Last Accessed: August 25, 2021. URL: https://github.com/monero-blackball/monero-blackball-site.

6 Monero Blackball Tool Code, 2023. Last Accessed: June 13, 2023. URL: https://github.com/monero-project/monero/blob/master/src/blockchain_utilities/blockchain_blackball.cpp.

7 Sina Aeeneh, João Otávio Chervinski, Jiangshan Yu, and Nikola Zlatanov. New attacks on the untraceability of transactions in CryptoNote-style blockchains. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5, 2021. doi:10.1109/ICBC51069.2021.9461130.

8 Sherman S. M. Chow, Christoph Egger, Russell W. F. Lai, Ivy K. Y. Woo, and Viktoria Ronge. On sustainable ring-based anonymous systems. In *36th IEEE Computer Security Foundations Symposium*, 2023.

9 Timothy A. Davis. CSparse: A concise sparse matrix package. URL: https://people.engr.tamu.edu/davis/suitesparse.html.

10 A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 10:517–534, 1958. doi:10.4153/CJM-1958-052-0.

11 Christoph Egger, Russell W. F. Lai, Viktoria Ronge, Ivy K. Y. Woo, and Hoover H.F. Yin. On defeating graph analysis of anonymous transactions. In Michelle Kerschbaum, Florian; Mazurek, editor, *Proceedings on Privacy Enhancing Technologies*, volume 2022 (3), page 538–557, Warschau (Polen), 2022. Sciendo. URL: https://petsymposium.org/popets/2022/popets-2022-0085.pdf, doi:10.56553/popets-2022-0085.

12 Brandon Goodell. Perfect privacy or strong deniability? In *Monero Konferenco*, 2019. URL: https://youtu.be/xicn4rdUj_Q.

13 Abraham Hinteregger and Bernhard Haslhofer. An empirical analysis of Monero cross-chain traceability. In *Financial Cryptography and Data Security*, pages 150–157, 2019.

14 Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of Monero's blockchain. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2017.

15 Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy*, pages 325–335. Springer, 2004.

16 L. Lovász and M.D. Plummer. *Matching Theory*. North-Holland, 1986.

17 Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the Monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3):143–163, 2018. doi:10.1515/popets-2018-0025.

18 João Otávio Chervinski, Diego Kreutz, and Jiangshan Yu. Analysis of transaction flooding attacks against Monero. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8, 2021. doi:10.1109/ICBC51069.2021.9461084.

19 Alex Pothen and Chin-Ju Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.*, 16(4):303–324, December 1990. doi:10.1145/98267.98287.

20 Viktoria Ronge, Christoph Egger, Russell W. F. Lai, Dominique Schröder, and Hoover H.F. Yin. Foundations of ring sampling. *Proceedings on Privacy Enhancing Technologies*, 2021:265–288, 2021. doi:10.2478/popets-2021-0047.

21 Nicolas van Saberhagen. CryptoNote v 2.0. White paper, 2013. URL: https://cryptonote.org/whitepaper.pdf.

22 Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, and Dongxi Liu. Monero ring attack: Recreating zero mixin transaction effect. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1196–1201, 2018. doi:10.1109/TrustCom/BigDataSE.2018.00165.

23 Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, Dongxi Liu, and Tsz Hon Yuen. Anonymity reduction attacks to Monero. In Fuchun Guo, Xinyi Huang, and Moti Yung, editors, *Information Security and Cryptology*, pages 86–100, Cham, 2019. Springer International Publishing.

24 Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, and Jiangshan Yu. On the unforkability of Monero. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 621–632, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3321705.3329823.

25 J. Yu, M. H. A. Au, and P. Esteves-Verissimo. Re-thinking untraceability in the CryptoNote-style blockchain. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 94–106, 2019. doi:10.1109/CSF.2019.00014.

26 Zuoxia Yu, Man Ho Au, Jiangshan Yu, Rupeng Yang, Qiuliang Xu, and Wang Fat Lau. New empirical traceability analysis of CryptoNote-style blockchains. In *Financial Cryptography and Data Security*, pages 133–149, 2019.