New Empirical Traceability Analysis of CryptoNote-Style Blockchains

Zuoxia Yu[1], Man Ho Au[1,⋆], Jiangshan Yu[2], Rupeng Yang[3,1], Qiuliang Xu[4], and Wang Fat Lau[1]

[1] Department of Computing, The Hong Kong Polytechnic University, Hong Kong
csallen@comp.polyu.edu.hk
[2] Monash University, Australia
[3] School of Computer Science and Technology, Shandong University, China
[4] School of Software, Shandong University, China

Abstract. The cascade effect attacks (PETS' 18) on the untraceability of Monero are circumvented by two approaches. The first one is to increase the minimum ring size of each input, from 3 (version 0.9.0) to 7 in the latest update (version 0.12.0). The second approach is introducing the ring confidential transactions with enhanced privacy guarantee. However, so far, no formal analysis has been conducted on the level of anonymity provided by the new countermeasures in Monero. In addition, since Monero is only an example of leading CryptoNote-style blockchains, the actual privacy guarantee provided by other similar blockchains in the wild remains unknown.

In this paper, we propose a more sophisticated statistical analysis on CryptoNote-style cryptocurrencies. In particular, we introduce a new attack on the transaction untraceability called *closed set* attack. We prove that our attack is optimal assuming that no additional information is given. In other words, in terms of the result, *closed set* attack is equivalent to brute-force attack, which exhausts all possible input choices and removes those that are impossible given the constraints imposed by the mixins of each transaction.

To verify the impact of our attack in reality, we conduct experiments on the top 3 CryptoNote-style cryptocurrencies, namely, Monero, Bytecoin and DigitalNote, according to their market capitalization. Since the computational cost of performing *closed set* attack is prohibitively expensive, we propose an efficient algorithm, called clustering algorithm, to (approximately) implement our attack. By combining our clustering method with the cascade attack, we are able to identify the real coin being spent in 70.52% Monero inputs, 74.25% Bytecoin inputs, and in 91.56% DigitalNote inputs.

In addition, we provide a theoretical analysis on the identified *closed set* attack, i.e., if every input in a CryptoNote-style blockchain has 3 mixins, and all mixins are sampled uniformly from all existing coins, the success rate of this attack is very small (about 2^{-19}). Given that *closed set* attack is equivalent to the best possible statistical attack, our findings provide two key insights. First, the current system configuration of Monero is

⋆ Corresponding author.

secure against statistical attacks, as the minimum number of mixin is 6. Second, we identify a new factor in improving anonymity, that is, the number of unspent keys. Our analysis indicates that the number of mixins in an input does not need to be very large, if the percentage of unspent keys is high.

1 Introduction

Since the introduction of Bitcoin in 2009 [10], numerous distributed cryptocurrencies have been proposed. Nonetheless, most of existing cryptocurrencies are not designed to provide strong privacy protection. For instance, several works [11, 7, 13] showed Bitcoin, currently the most popular and largest cryptocurrency, is vulnerable to de-anonymization attacks.

To address this problem, privacy-preserving cryptocurrencies with stronger privacy guarantees are attracting increasing attentions. Among them, CryptoNote-style cryptocurrencies are one of the noteworthy efforts. The CryptoNote protocol was first introduced in [14], with a focus on protecting the privacy and anonymity of the electronic cash. Since its introduction, many variations utilizing this protocol have been proposed, including Bytecoin, Boolberry, Dashcoin, DigitalNote, Monero, etc. Similar to many other distributed cryptocurrencies, CryptoNote also adopts the notion of transaction to represent the process of spending coins. Each transaction contains several inputs and outputs, where inputs consume coins from the sender, outputs transfer coins to the receiver. The total amount of coins consumed in the inputs and the total amount of coins transferred to the outputs should be equal. Besides, each transaction should be signed by the sender to authorize the transfer, by using the private key associated to the public-key (address)[5] of a to-be-spent coin. Moreover, a ring signature [12, 6] scheme is adopted to guarantee the privacy of the real-spend of each input, which is a cryptographic primitive that allows a user to anonymously sign a message on behalf of a group of users. Therefore, the identity of the real-spend is hidden. All other decoy coins in the input are called mixins.

However, in practice, CryptoNote-style cryptocurrencies fall short from realizing their claimed anonymity. Recently, two independent and concurrent works [8, 5][6] demonstrate that Monero transactions may be de-anonymized via statistical analysis. Specifically, they found that most inputs in Monero have very small number of mixins and more than half inputs are paid without having any mixin. Those inputs without mixins can be trivially de-anonymized. Even worse, once a coin payed without mixin is chosen as a mixin in another transaction, the input of this transaction also faces a danger of being de-anonymized. Based on this simple yet vital observation, these two works adopt similar strategies to conduct

[5] Throughout this paper, we interchangeably use the term coin, output and the public-key.

[6] An updated version [9] of [8] also appears recently, but both the method and the result for the traceability analysis are similar in these two works, thus we focus on the initial version.

empirical evaluations, which are based on the so-called “chain-reaction” analysis [8] or cascade effect [5]. Roughly speaking, the attacker first identifies all inputs with zero-mixin. As each located input is payed by merely one public-key, the public-key must be the real payer of the input. Since each public-key can only be used once in Monero, it is safe to delete these de-anonymized public-keys in mixins of the remaining inputs. This will lead to new zero-mixin inputs and the attack could be conducted repeatedly. According to the experiment results of [8, 5], by Feb 2017, nearly 65% of transaction inputs are with zero-mixin, and the cascade effect can render another 22% of inputs traceable, i.e., nearly 87% of all Monero inputs are insecure when considering users’ anonymity.

Having witnessed (and predicted) this type of attacks, Monero has proposed a few countermeasures. First, at version 0.9.0 (January 1, 2016), it releases a mandatory requirement that each transaction input should include at least 2 mixins. Subsequently, at version 0.10.0 (September 19, 2016), ring confidential transaction (RingCT), which aims at further enhancing privacy of users via hiding the transaction amount, is introduced. An added advantage of employing RingCTs is that all RingCT input must use outputs of RingCTs as its mixins, i.e., no public-key used before version 0.10.0 will be chosen as mixin for a RingCT input. Therefore, neither the chain-reaction attack nor the cascade attack works for RingCTs. Besides, after realizing the effect of the number of mixins, the minimum number of mixins is further increased from 2 to 6 in version 0.12.0 (March 29, 2018).

There is no doubt that known attacks are circumvented by Monero, but more fundamental problem remains, and it still threatens all CryptoNote-style currencies. That is, can anonymity of users be well-protected with the current ring size, i.e., the countermeasures for known attacks? A related question is how to theoretically analyze the security level achieved by those cryptocurrencies adopting ring signature for untraceability. Besides, how about the anonymity achieved in practice by other CryptoNote-style currencies?

Our Contributions. In this paper, we give answers to the above questions. First, we show that the current countermeasures to resist known attacks make Monero a good system to provide anonymity. However, on the negative side, we show other CryptoNote-style protocols are still suffering from the same type of attacks. In fact, our combined attacks are much more effective on Bytecoin and DigitalNote, as we can de-anonymize up to 91.56% transactions in the chain of DigitalNote.

We introduce a new attack on the untraceability of CryptoNote-style currencies called *closed set* attack. This attack is based on the fact that n transaction inputs will and must use n distinct public-keys as real-spend, since each public-key can only be redeemed once. A set of inputs is called a *closed set* if the number of inputs equals to the number of distinct public-keys included. Hence, we can deduce that all public-keys included in a *closed set* must be mixins in other inputs outside of this *closed set*. In this way, the searching for *closed set*s will be helpful to trace the real-spend of some other inputs. Different from cascade effect attack which relies on the “chain-reaction analysis” due to zero-mixin inputs, *closed set* attack conducts further traceability without relying on any previous traceable inputs.

The contributions of this work can be divided into the following aspects:

1. We introduce *closed set* attack on the untraceability of CryptoNote-style blockchains, and prove that *closed set* attack is *optimal*. In particular, it could get the minimal mixin for every input, i.e., it deletes all public-keys payed elsewhere in a mixin, identical to the results of brute-force attack.
2. We verify the impact of our attack via performing experiments on actual blockchain data, where we pick the top 3 CryptoNote-style currencies by market capitalization, i.e., Monero, Bytecoin and DigitalNote. As the proposed attack is too expensive to run due to its high complexity, we propose an efficient algorithm, namely, clustering algorithm, to (approximately) implement *closed set* attack. We give a lower bound of our clustering algorithm in implementing the *closed set* attack. Specifically, we prove that our algorithm can find all *closed set* of size less than or equal to 5. The experiment results of these three currencies are given in **Table** 1.

Table 1: Experiment Results. All inputs considered in this paper are non-coinbase transaction unless specific stated. The items under column "Cas."(resp. "Clu.") denote the total number and percentage of inputs traced by cascade attack (resp. clustering attack). "No. of C.S." denotes the total number of *closed set*s found by the clustering algorithm.

Coin	total Blocks	Total Inputs	Deducible (%)	Cas. (%)	Clu. (%)	No. of C.S.
Monero	1541236 (30 March 2018)	23164745	16334967 (70.516%)	16329215 (99.96%)	5752 (0.04%)	3017
Bytecoin	1586652 (3 August 2018)	45663011	33902808 (74.25 %)	33822593 (99.763%)	80215 (0.237%)	5912
DigitalNote	699748 (13 August 2018)	8110602	7426036 (91.56%)	7425987 (99.9993%)	49 (0.0007%)	38

3. In addition, we also provide a theoretical analysis on the existence of *closed set*. We find that if all inputs have 3 mixins and all mixins are uniformly distributed, with all but a very small probability (about 2^{-19}), there will not exist any *closed set*. Our analysis suggests that the usage rate of outputs is closely related to the anonymity of Monero. Moreover, if we can guarantee that the probability of choosing an unspent key as mixin is 25%, then the number of mixins of each input could be as small as 3 to render brute-force attack ineffective.

Communication with the community. We have fully disclosed our results to the related research communities, including CryptoNote, Monero, Bytecoin, and DigitalNote. We learnt that Monero researchers have concurrently and independently observed similar attacks [3], and the blackball tool developed by Monero is able to identify a part of the closed sets we identified in this work. Considering Monero, no RingCT transactions are affected under their blackball tool and our current analysis. We will release our code as an open-source repository, and work with Monero to help improving its tool set to enhance user privacy.

2 Preliminary

CryptoNote Protocol. CryptoNote protocol [14] aims at providing a privacy enhanced cryptocurrency, with the following two properties:

- Untraceability: for any transaction, the real-spend should be anonymous among all the sets of outputs in an input;
- Unlinkability: for any two transactions, it is impossible to prove that they were sent to a same user.

To guarantee unlinkability, for each output in a transaction, CryptoNote uses a one-time random public-key as the destination, which is derived from receiver's public-key and sender's random data. In this way, only the receiver who holds the permanent secret key can redeem that output. For the untraceability, CryptoNote adopts ring signature, which is a primitive that allows a user to anonymously sign a transaction on behalf of a group of users, which is usually referred as a ring. Therefore, the real-spend will be hidden via the help of other outputs, which are called mixins. Obviously, for an input with n public-keys, the number of mixin is n-1.

Notation. We use $[m]$ to denote the integer set $\{1, 2, \dots, m\}$. For any set S, we use $|S|$ to denote its size. For a transaction tx, we use $tx.in$ to denote an input of this transaction, which is a set of public-keys $\{pk_1, pk_2, \dots, pk_\ell\}$ used to create a ring signature. We also interchangeably call each input $tx.in$ of a transaction as a ring R throughout this paper. Specifically, we use $R = \{pk_1, pk_2, \dots, pk_n\}$ to denote the transaction input including public-keys $pk_1, pk_2, \dots, pk_n$.

We also need the Chernoff bound in our analysis. There are various forms of the Chernoff bound, here we use the one from [4].

Lemma 1 (Chernoff Bounds). *Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}(X) = \sum_{i=1}^{n} p_i$. Then*

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{2+\delta}\mu} \text{ for all } \delta > 0;$$

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2}{2}\mu} \text{ for all } 0 < \delta < 1.$$

3 Closed Set Attack

In this section, we introduce our *closed set* attack. All attacks considered in this section only assume access to the transactions in the blockchain of a CryptoNote-style currency, without any further active ability. This assumption is valid since all transactions on the blockchains are publicly accessible.

We prove that our proposed *closed set* attack is optimal, i.e., brute-force attack. Looking ahead, brute-force attack will traverse all possible assignments of payers of all inputs and delete those with conflict data. Both our attack and brute-force attack return the minimum set of candidates for the real payer of each input.

3.1 Brute-Force Attack

Brute-force attack is an attack that tries all possible sequence of distinct public-keys to test whether it is valid for the assignments of the real-spends for all transaction inputs. While a sequence of public-keys is valid if it satisfies requirements: 1) the size of the sequence equals to the number of total transaction inputs in the dataset; 2) all public-keys included in the sequence are distinct; 3) for all index i of that sequence, the i-th public-key in the sequence belongs to the corresponding i-th input in the dataset. In other words, brute-force attack is the process of searching for all valid sequences among the permutations of all public-keys with specific length according to the above requirements. We call all elements included in index i ($i \leq$ no. of all inputs) of all valid sequences as the candidates for the real-spend of the i-th transaction input. Therefore, the resulting valid sequences are the combinations of the possible real-spend of each transaction input. Besides, if a transaction input only has one candidate for the real payer, then the candidate must be its payer.

It is not hard to see that brute-force attack is a perfect attack which can find out all possible real-spends for each transaction input. Assume that there are n distinct keys and m transaction inputs in our dataset, and without loss of generality, n is larger than m. Let $\mathbb{A}_n^m$ denote the number of permutations of m elements among n elements. The number of valid sequences after the execution of brute-force attack is $(\mathbb{A}_n^m - |\mathsf{Conflicts}|)$, where $\mathsf{Conflicts}$ denotes the set of deleted permutations which fail to the above requirements.

3.2 Our Attack

Although the aforementioned brute-force attack is perfect, the complexity is prohibitively high in practice, which is $\mathcal{O}(n!)$. Considering the inefficiency and impracticability of brute-force attack, we propose a new attack called *closed set*, which is more efficient while providing the same result.

The proposed *closed set* attack is based on the observation that if the number of distinct public-keys included in a set of transaction inputs equals to the number of the inputs of the set, then we can deduce that each public-key included must be a real-spend of a certain input in this set, and be mixin in other outside inputs. In this way, the finding of a *closed set* has at least two significant impacts. Firstly, it will render other inputs become traceable after removing public-keys of a *closed set*. Secondly, the average size of the inputs will decrease, which is helpful for further operation.

The *closed set* attack is an iteration process that finds out all possible *closed set*s from the transaction inputs, removes public-keys included, and finds those traceable inputs. Compared with the previous cascade effect attack presented by [5] or "chain-reaction" analysis by [8], the *closed set* attack can render more inputs traceable. More precisely, cascade effect attack utilizes the fact that the zero-mixins inputs will affect the traceability of other inputs who pick those public-keys of them as mixins. In other words, this attack bases on the set of previous traceable inputs to track the remaining anonymous ones, while our attack can start from any anonymous input.

To better explain our attack, we give a brief example below. Here we consider four inputs included in transactions $\{tx_i\}_{i\in[4]}$ and assume that there are four distinct public-keys $\{pk_j\}_{j\in[4]}$ included in the input sets of them, i.e.,

$$\begin{aligned} &tx_1.in = \{pk_1, pk_2, pk_3\};\\ &tx_2.in = \{pk_2, pk_3\};\\ &tx_3.in = \{pk_1, pk_3\};\\ &tx_4.in = \{pk_1, pk_2, pk_3, pk_4\}. \end{aligned}$$

Note that, there must exist no other transaction input who is only composed of public-keys among $\{pk_j\}_{j\in[4]}$. Otherwise, the design principle of Monero that one output can only be redeemed once will be broken. While the original cascade attack [8, 5] does not work here, since there exists no 0-mixin input.

Although we can not make all aforementioned inputs traceable, but we can trace the real-spend of one of them. Specifically, consider the set $S = \{tx_i.in\}_{i\in[3]}$. Among that, the union set of all distinct public-keys included is $\{pk_1, pk_2, pk_3\}$. Clearly, the size of S equals to the number of distinct public-keys included in it such that it is a *closed set*. Since each output can be spent once only, then the output pk_j $(j \in [3])$ must be a real-spend in a certain $tx_j (j \in [3])$. In this way, we can deduce that the real-spend of tx_4 must be pk_4.

A Naive Implementation. A naive method to find all *closed set*s is to visit all possible subsets of transaction inputs. For each visited subset, we check whether it is a *closed set* by comparing the number of inputs and the number of distinct public-keys included in it. If yes, we further conduct the removing and tracing operations triggered by this *closed set*. Otherwise, continue the process until all subsets have been visited. Due to space limitation, we give this algorithm in **Appendix** A.

Theoretically, this algorithm can find all *closed set* included in all transaction inputs. However, it is expensive to implement in reality, since the complexity of traversing all subsets of inputs is $\theta(2^m)$, where m is the total number of all transaction inputs included in the blockchain. For instance, up to block 1541236 of Monero, the number of untraceable inputs remained is 6835530 after the execution of the cascade attack. Starting from these inputs, at the step of searching subsets of size 5, the complexity of the algorithm will become $\mathcal{O}(2^{100})$.

Analysis of Closed Set Attack. We prove that our *closed set* attack is optimal. In other words, our *closed set* attack is equivalent to brute-force attack.

Specifically, we prove that after the execution of our *closed set* attack, each transaction input is the set of candidates of the real-spend of it found by brute-force attack. The analysis is concluded by the following theorem.

Theorem 1. *The aforementioned closed set attack is equivalent to brute-force attack. In other words, for any set of transactions, the impact of our attack on it is identical to the impact of brute-force attack.*

Due to space limitation, we refer readers to the full version of this paper for the proof of this theorem.

3.3 On The Existence of Closed Set: A Theoretical Perspective

As mentioned before, the *closed set* attack is optimal. This is to say, we can conclude that anonymity of inputs cannot be reduced if no *closed set* exists. In this section, we estimate the probability that there exists at least one *closed set* in an ideal scenario, namely, all inputs have a (small) constant number of mixins and all mixins are selected uniformly from all keys.

More concretely, we consider a scenario that

- There are $6 \cdot 2^{20}$ inputs, with $6 \cdot 2^{20}$ real-spend public-keys;
- There are also additional 25% (i.e. $2 \cdot 2^{20}$) unspent public-keys;
- Each input has 3 mixins;
- Each mixin is sampled uniformly from all $8 \cdot 2^{20}$ keys;

where the first two conditions come from the real data of Monero after cascade attack, and the third condition is based on the fact that the average ring size after the cascade attack is 4.62.

Lemma 2. *With all but a small probability 2^{-19}, there does not exist any closed set in the above dataset if all inputs have 3 mixins and all mixins are sampled uniformly from all keys.*

The proof of this lemma is given in the full version of this paper.

4 Our Clustering Algorithm

Considering the impracticability of subset-based algorithm mentioned above, here we introduce an approximate but efficient algorithm for searching *closed sets*, which is named as clustering algorithm. Looking ahead, although the clustering algorithm is just an approximate algorithm, we show that the lower bound of the size of *closed set* found by it is 5. In other words, all *closed set* with size less than or equal to 5 can be found. Besides, we conduct experiments and find that our clustering algorithm achieves a better result during the actual execution.

Intuition of Our Clustering Algorithm. Recall that, the main feature of a *closed set* is that it embraces the same number of transaction inputs and distinct public-keys. Hence, our target should be finding a set of inputs with the above characteristics. To do so, one intuitive way is forming a set from a certain input, then absorb other input which is helpful to achieve a *closed set*. A key challenge is how to select other rings?

We observe that since the ultimate target is to make two numbers about this set equal, it is possible to select rings based on the consequence of adding an input into a set. For instance, assuming the set being considered now is called S, which is initialized by input R. Whenever an input R' is added into S, the possible consequences can be divided into the following three cases:

- Case 1. If all public-keys included in R' are a subset of all public-keys contained in S, then for set S, the number of included transaction inputs is increased by one, and the number of distinct public-keys remains the same. Thus, the insertion of R' will certainly increase the possibility of S becoming a *closed set*. We call such kind of input as useful input.
- Case 2. If the insertion of R' will only introduce one distinct public-key to S, then the insertion of this input will not change the current relationship between the number of distinct public-keys and the number of inputs included in S. This kind of input extends the public-key set of S, which maybe helpful for absorbing other inputs. We call such kind of input as uncertain input.
- Case 3. If the insertion of R' will introduce two or more distinct public-keys to S, then the number of inputs will only be increased by one, but the number of public-keys will be increased by 2 or more. As this does not help our analysis at all, we call such kind of input as bad input.

Above all, if we only pick the relatively useful and uncertain inputs to a set, then we can find a *closed set* faster with high probability.

Definition of Cluster. A cluster $Clus$ is defined as a set of inputs, namely, $Clus = \{R_1, R_2, \ldots, R_n\}$. Each cluster represents a set PK_Clus, which is defined as $PK\_Clus = \bigcup_{R \in Clus} R$. In other words, PK_Clus is the set used to collect all distinct public-keys included in the inputs of $Clus$.

The *distance* from an input to a cluster is defined as the number of public-keys included in the input but not in the cluster. The formal definition of it is given below:

$$Dist(R, Clus) = Dist(R, PK\_Clus) = |R| - |PK\_Clus \cap R|,$$

where R is the input considered to be added, and $Clus$ is a cluster with public-keys set PK_Clus. Notably, this definition is not symmetric. According to our definition, the distance from an input to a cluster, i.e., $Dist(R, Clus)$, is different with the distance from a cluster to an input, i.e., $Dist(Clus, R)$.

For instance, consider the cluster $Clus$ and the input R composed as follows:

$$\begin{aligned} &Clus = \{\{pk_1, pk_2\}, \{pk_1, pk_3\}, \{pk_2, pk_4\}\}, \\ &R = \{pk_1, pk_3, pk_5\}. \end{aligned}$$

Obviously, the public-key set of $Clus$ is $PK\_Clus = \{pk_1, pk_2, pk_3, pk_4\}$. The size of R is 3, and number of common public-keys are 2. Hence, according to our definition, the distance from R to $Clus$ is $Dist(R, Clus) = Dist(R, PK\_Clus) = 3 - 2 = 1$. So, if we add R into $Clus$, then only one new public-key, i.e., pk_5, will be introduced in $Clus$.

Starting from a specific input, the construction of a cluster is a dynamic process of searching for other qualified inputs. To clarify which kind of inputs can be absorbed into a cluster, we associate each cluster with a distance. More precisely, we say a cluster $Clus$ with distance 1, if only those inputs satisfying $Dist(R, Clus) \leq 1$ can be added into it. As the insertion operation may cause changes to a cluster, we should always adopt the present cluster to calculate the distance from an input to it. The construction algorithm of a cluster from a certain input is given in **Algorithm** 1.

Algorithm 1 $Cluster\_Form$(R)

```
1: Start with an input $R$, and define the cluster as $Clus = \{R\}$
2: Let $DataSet$ be all transaction inputs in the blockchain
3: for each $R'(\neq R) \in DataSet$ do
4:     if $Dist(R', Clus) \leq 1$ then
5:         $Clus = Clus \cup \{R'\}$
6: return $Clus$
```

For each cluster, we use two additional parameters to check whether it is a *closed set*. One is the number of inputs included in it, the other one is the number of distinct public-keys included. Formally, if the number of inputs equals to the number of distinct public-keys included in a cluster, we say that this cluster is a *closed set*. Besides, in some cases, a *closed set* may contain other sub-*closed set*. To find all *closed set*s, whenever we get a *closed set* via this algorithm, we further conduct a sub-*closed set* searching operation. An important observation is that if a public-key only appears once in a *closed set*, then it must be the real spend of the input including it. For simplicity, we utilize this method to test whether there exists sub-*closed set* inside a *closed set*, since the complexity of brute-forcing all subsets of this *closed set* is quite large.

Next we introduce the clustering algorithm for all clusters with distance 1. The main idea is that we repeatedly pass over all the transaction inputs via numerous iterations. In each iteration, the algorithm picks an input and uses it to initialize a cluster $Clus$. Then we run the constructing cluster algorithm (Algorithm 1) to add proper inputs into $Clus$. We continue the next iteration if the resulted cluster is not a *closed set*. Otherwise, before starting with the next iteration, the algorithm should finish the following operations. Remove all public-keys contained in this cluster from the remaining inputs, and find the traceable ones. Afterwards, we check whether the current *closed set* includes a public-key such that it only appears in one input. If yes, we further de-anonymize inputs inside a *closed set*.

The algorithm of searching for all clusters with distance 1 from all transaction inputs in the blockchain is given in **Algorithm** 2. Notably, all rings considered in our algorithm are anonymous. Once finding the real-spend of an input, we will not do any operation on that input. Besides, our algorithm concentrates on resulting data after the execution of cascade effect attack. Hence, in **Algorithm** 2, we abuse the concept, where a cascade effect algorithm is first invoked.

Algorithm 2 Clustering Algorithm

```
 1: Let $DataSet$ be all transaction inputs in the blockchain.
 2: Cascade-Effect(Dataset)
 3: Flag = true
 4: while Flag == true do
 5:     Flag = false
 6:     for each $R \in DataSet$ do
 7:         $Clus\_Form(R) \to Clus$
 8:         if $Clus$ is a closed set then
 9:             Remove$(Clus) \to Flag$
10:             if Flag == true then
11:                 find traceable inputs
12:                 check whether rings inside $Clus$ are traceable
```

Analysis of accuracy. The accuracy of the clustering algorithm is analyzed through the following theorem, which gives a lower bound of the clustering algorithm. This is to say that all *closed set*s with size less than or equal to 5 can be found after the execution of the clustering algorithm.

Theorem 2. *After the execution of our clustering algorithm with searching distance 1, all indivisible closed sets with size less than or equal to 5 can be returned by our algorithm.*

We refer readers to the full version of this paper for the proof of this theorem.

Analysis of complexity. Assume the total number of transaction inputs included in the blockchain is N. The number of iterations in our algorithm is $\theta(N)$. Suppose the average length of an input is ℓ. While in each iteration, in the worst case, we calculate $\mathcal{O}(\ell N)$ times distance between all inputs and the current clusters. Therefore, in the worst case, the complexity is $\theta(\ell N^2)$.

5 Experiment Result

To evaluate the level of anonymity achieved by the CryptoNote-style currencies, as well as the estimation of the probability of the existence of *closed set*s in reality, we implement our clustering algorithm in C++, and the program is executed on a computer with 3.1 GHz Intel Core i5 Processor, 16 GB RAM and 256GB

SSD storage disk. Notably, here we only analyze the top three CryptoNote-style currencies according to their market capitalizations [1], i.e., Monero, Bytecoin and DigitalNote. For all these three currencies, we export all related data directly from the corresponding blockchain database via modifying its source codes.

5.1 Analysis of Monero

As there are two pioneering works [8, 5] considering cascade effect attacks on the untraceability of Monero transactions, we mainly concentrate on the analysis of the anonymity of those data after the known attacks.

Dataset Collection. We collect all blocks in the Monero blockchain from the first block (18th April 2014) up to block 1541236 (30th March 2018). Additionally, all related data is directly exported from the blockchain database via modifying the source code of Monero [2]. Our dataset in total contains 4153307 transactions. Among them, 2612070 are non-coinbase transactions, which are composed of 23164745 transaction inputs in total, and 25126033 distinct public-keys are involved. Notably, throughout this paper, we only consider those non-coinbase transactions unless otherwise stated.

Experiment Results. In **Table** 2, we give the result of the clustering algorithm on the aforementioned dataset. As it turns out, a total of 16334967 inputs become traceable. Specifically, 16329215 inputs are traceable due to the cascade effect attack, and the remaining inputs, i.e., 5752 in total, are traced by the finding of *closed set*. Total of 70.52% of Monero transaction inputs are traceable. While for the dataset after the cascade effect attack, only 0.084% inputs can be further traced.

Table 2: The traceability of Monero.

No. of mixins	Total	Deducible	Cascade Effect	Clustering Algorithm	(%)
0	12209675	12209675	12209675	0	100
1	707786	625641	625264	377	88.39
2	4496490	1779134	1776192	2942	39.57
3	1486593	952855	951984	871	64.10
4	3242625	451959	451230	729	13.94
5	319352	74186	73980	206	23.23
6	432875	202360	202100	260	46.75
7	21528	4296	4282	14	19.96
8	30067	3506	3490	16	11.66
9	17724	2178	2162	16	12.29
$\geq$10	200030	29177	28856	321	14.59
Total	23164745	16334967	16329215	5752	70.52

Besides, a total of 6829778 transaction inputs are still untraceable. For all these remaining inputs, we give the frequency of number of mixins before and after the execution of clustering algorithm in **Figure** 1.

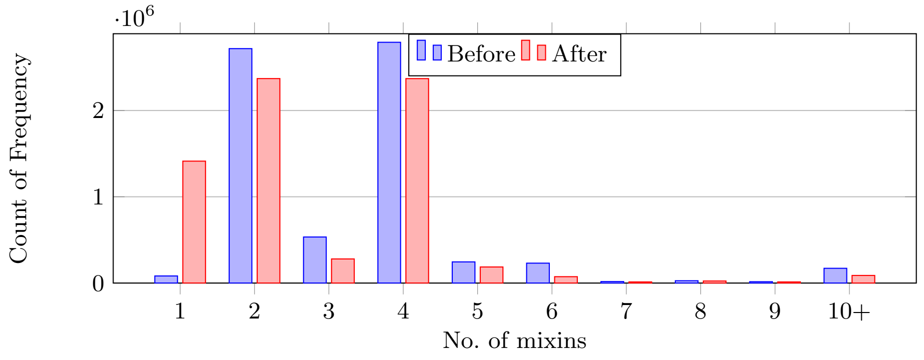

Fig. 1: Frequency of number of mixins of those anonymous inputs before and after the execution of clustering algorithm.

The clustering algorithm also finds 3017 distinct *closed set*s, whose size vary from 2 to 55, and include a total of 7478 distinct public-keys. As we mentioned before, these 7478 public-keys must be the real-spend of a certain input contained in these *closed set*. In other words, we can deduce that they are spent although we do not know which concrete transaction they are used. However, it is useless for the anonymity if any other new input picking public-keys from them.

One may wonder there is a discrepancy between probability of 2^{-19} for finding *closed set* and the existence of 3017 *closed set*s found during the experiment. This is due to the fact that our analysis assumes mixins are chosen uniformly and that each input has 3 mixins. However, in practice, sampling distributions and number of mixins of all inputs are not uniform. This will increase slightly the probability of finding *closed set*.

5.2 Analysis of Bytecoin

We provide analysis on the untraceability of Bytecoin via cascade effect attack and clustering attack.

Dataset Collection. We collect all blocks in the Bytecoin blockchain from block 1 (4 July 2012) to block 1586652 (3 August 2018). A total of 3782566 non-coin based transactions is contained in this dataset, and there are altogether 45663011 transaction inputs included. Additionally, a total of 48613764 distinct public-keys are involved.

Experiment Results. The experiment result on Bytecoin dataset is summarized in **Table** 3. More specifically, a total of 33902808 Bytecoin transaction inputs become traceable, counting for 74.25% of all inputs considered in our dataset. Among them, 28591486 inputs are zero-mixin inputs, and the cascade effect caused by them further makes 5231107 inputs become traceable, which covers 99.763% of the total traceable inputs. Besides, our clustering algorithm traces another 80215 transaction inputs from the remaining ones, which count to 0.68% of those untraceable inputs after the cascade effect attacks. There are a total of 5912 *closed set*s found, whose size vary from 2 to 55.

Table 3: The traceability of Bytecoin.

No. of mixins	Total	Deducible	Cascade Effect	Clustering Algorithm	(%)
0	28591486	28591486	28591486	0	100
1	5751268	3281500	3240142	41358	57.06
2	2840745	1133602	1112648	20954	39.91
3	1442133	261197	260298	899	18.11
4	2516851	276237	275172	1065	10.98
5	617041	59922	59493	429	9.71
6	3145092	270355	255156	15199	8.60
7	388759	26434	26160	274	6.80
8	81504	1231	1220	11	1.51
9	65379	397	389	8	0.61
≥10	222753	447	429	18	0.2
Total	45663011	33902808	33822593	80215	74.25

5.3 Analysis of DigitalNote

We also provide the first work on analyzing the untraceability of DigitalNote.

Dataset Collection. We collect all 633548 non-coin based transactions included in the block 1 (31 May 2014) up to block 699748 (13 August 2018) in the DigitalNote blockchain. A total of 8110602 inputs are included in the aforementioned transactions, and 8396472 distinct public-keys are involved.

Experiment Results. The experiment result of DigitalNote is given in **Table** 4. Specifically, 91.56% of all transaction inputs in our dataset is traceable, while 60.39% of them is without any mixin. Besides, the cascade attack further contributes 39.60% of those traceable inputs. Our clustering algorithm makes 49 additional inputs traceable, which covers 0.007% of the untraceable inputs after the cascade effect attacks, with the help of 38 *closed set*s.

6 Observations and Recommendations

In this section, we give our observations and recommendations according to the experiment results.

- **Observation 1**: *The usage rate of outputs is an important factor for the anonymity of CryptoNote-style currencies.* The usage rate of outputs refers to the percentage of public-keys that have been spent, which can be easily calculated by using the total amount of inputs in the dataset over the total

Table 4: The traceability of DigitalNote .

No. of mixins	Total	Deducible	Cascade Effect	Clustering Algorithm	(%)
0	4484726	4484726	4484726	0	100
1	2087295	1847151	1847132	19	88.49
2	1194410	895480	895472	8	74.97
3	129700	101872	101872	0	78.54
4	6225	4362	4358	4	70.07
5	193669	85941	85939	2	44.38
6	3071	1840	1837	3	59.92
7	844	442	440	2	52.38
8	1686	856	853	3	50.77
9	1288	682	681	1	52.95
$\geq$10	7688	2684	2677	7	34.91
Total	8110602	7426036	7425987	49	91.56

number of distinct outputs (i.e., public-keys), as each output can only be redeemed once. As mentioned in **Section** 3.3, those unspent public-keys play an important role in preventing the formation of a *closed set*. Hence, it is fair to say that, to some degree, decreasing the usage rate will improve anonymity.

- **Observation 2**: *Closed sets are closely related to the anonymity of inputs.* In this work, we have shown that finding *closed set*s could help identify real-spends or decrease the ring size (so the level of anonymity) of those inputs. Although the probability of the existence of a *closed set* is not high, but *closed set*s do exist and threaten the anonymity of inputs.
- **Recommendation 1:** *Decreasing the usage rate of outputs by generating more outputs.* Recall that a lower usage rate of outputs is beneficial to the anonymity of Monero inputs. Hence, to decrease the usage rate of outputs, we recommend users to additionally generate some outputs with 0 amount, which can make the unspent output set larger.
- **Recommendation 2:** *Do not pick the useless mixin.* Take the Monero as an example, our clustering algorithm has found 3017 distinct *closed set*s, which contain 7478 distinct public-keys. These 7478 public-keys must be the real-spend of a certain input contained in these *closed set*s. Hence, for any newly generated input, picking these keys as mixin will not improve anonymity. So, we recommend users not to pick these useless mixins. However, for an ordinary user, it is difficult to determine whether an output is contained in *closed set*s or not. Thus, we will release our code that implements the attack.

Acknowledgement. We appreciate the anonymous reviewers for their valuable suggestions. Part of this work was supported by the National Natural Science Foundation of China (Grant No. 61602396, U1636205, 61572294, 61632020), the MonashU-PolyU-Collinstar Capital Joint Lab on Blockchain and Cryptocurrency Technologies, from the Research Grants Council of Hong Kong (Grant No. 25206317), and the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128.

References

1. Cryptocurrencies market capacity. URL:`https://coinmarketcap.com/all/views/all/`, Online, accessed 16 April 2018.
2. Monero source code. URL:`https://github.com/monero-project/monero`,Online accessed 30 March 2018.
3. Sets of spent outputs. URL:`https://ww.getmonero.org/resources/research-lab/pubs/MRL-0007.pdf` ,Online, accessed 26 November 2018.
4. M. Goemans. Lecture notes in chernoff bounds, and some applications, February 2015. URL:`http://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf`.
5. A. Kumar, C. Fischer, S. Tople, and P. Saxena. A traceability analysis of monero's blockchain. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2017.
6. J. K. Liu, W. Susilo, and D. S. Wong. Ring signature with designated linkability. In *International Workshop on Security*, pages 104–119. Springer, 2006.
7. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.
8. A. Miller, M. Möser, K. Lee, and A. Narayanan. An empirical analysis of linkability in the monero blockchain. *arXiv preprint arXiv:1704.04299*, 2017.
9. 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*, 3:143–163, 2018.
10. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
11. F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 1318–1326. IEEE, 2011.
12. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.
13. D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.
14. N. Van Saberhagen. Cryptonote v 2. 0, 2013.

A Subset-Based Algorithm

Here we give a naive algorithm to search for all *closed set*s through finding all subsets of transaction inputs. Looking head, we use Cascade-Effect(inputs) to denote the function which implements the cascade effect attack. Assume Remove $(closed\ set\ CS) \rightarrow flag$ is a function will remove all public-keys contained *closed set* CS from other inputs outside CS, and outputs a variable $flag = true$ if any removing operation happens. The algorithm is given in **Algorithm** 3 below.

Algorithm 3 Subset-Searching Algorithm

```
 1: Let DataSet be the set of all transaction inputs in the blockchain.
 2: Let $\ell$ be the size of current subset, and $\ell \geq 2$.
 3: Cascade-Effect(Dataset).
 4: while $\ell \leq |DataSet|$ do
 5:     Let $Set_\ell \subseteq DataSet$ be the set of all inputs, s.t., the size of each input is equal
        or smaller than $\ell$.
 6:     Let $\{Subset_{\ell,j}\}$ be all subsets of $Set_\ell$ with size $\ell$, where $j \in \mathcal{C}^{\ell}_{|Set_\ell|}$, and each
        $Subset_{\ell,j} = \{R_1, R_2, \ldots, R_\ell | \ \forall i \in [\ell], R_i \in Set_\ell\}$
 7:     for $j = 1$ to $\mathcal{C}^{\ell}_{|Set_\ell|}$ do
 8:         if $Subset_{\ell,j}$ is a closed set then
 9:             Remove($Subset_{\ell,j}$) $\rightarrow flag$
10:             if flag == true then
11:                 find traceable inputs
12:             goto while with $\ell$++
```