A Traceability Analysis of Monero's Blockchain

Amrit Kumar(✉), Clément Fischer, Shruti Tople, and Prateek Saxena

National University of Singapore, Singapore, Singapore
{amrit,cfischer,shruti90,prateeks}@comp.nus.edu.sg

Abstract. Privacy and anonymity are important desiderata in the use of cryptocurrencies. Monero—a privacy centric cryptocurrency has rapidly gained popularity due to its unlinkability and untraceablity guarantees. It has a market capitalization of USD 290M. In this work, we quantify the efficacy of three attacks on Monero's *untraceability* guarantee, which promises to make it hard to trace the origin of a received fund, by analyzing its blockchain data. To this end, we develop three attack routines and evaluate them on the Monero blockchain. Our results show that in 88% of cases, the origin of the funds can be easily determined with certainty. Moreover, we have compelling evidence that two of the attack routines also extend to Monero RingCTs—the second generation Monero that even hides the transaction amount. We further observe that over 98% of the results can in fact be obtained by a simple temporal analysis. In light of our findings, we discuss mitigations to strengthen Monero against these attacks. We shared our findings with the Monero development team and the general community. This has resulted into several discussions and proposals for fixes.

Keywords: Monero · Cryptocurrency · Blockchain · Traceability · Anonymity

1 Introduction

Since the seminal work by Chaum [3], privacy and anonymity properties have become important desiderata for any e-cash system. Bitcoin, the most popular decentralized cryptocurrency fairs poorly in terms of privacy and anonymity as evidenced by several analyses in the past [6,13,19,21]. In light of the privacy issues in Bitcoin, a new cryptocurrency called *Monero* (XMR) was launched on April 18th 2014. Monero currently has a market capitalization of USD 290M [12] and has the most momentum of all the live privacy-enhancing cryptocurrency projects (*e.g.*, Zcash [1,24], Dash [4], Mimblewimble [9], *etc.*), see [2]. The value of Monero in terms of USD increased by 27 times in the year 2016 from its value in 2015, making it a valuable example of privacy-enhancing cryptocurrencies to study. Its rise to popularity is mainly due to strong privacy properties (compared to Bitcoin) and design simplicity (compared to Zcash).

Background on Monero. As in any other distributed cryptocurrency (such as Bitcoin), Monero coins are spent in the form of a *transaction*, where a sender transfers the coins to a recipient, both often called the *users* of the system.

S.N. Foley et al. (Eds.): ESORICS 2017, Part II, LNCS 10493, pp. 153–173, 2017.
DOI: 10.1007/978-3-319-66399-9_9

Each sender and the recipient has a public and private key pair. The public key uniquely defines the payment *address*. The transaction transfers funds held in the sender's public key to the recipient's public key. A transaction contains a set of *inputs* and *outputs*, wherein inputs consume coins from the sender and outputs transfer them to the recipient, while conserving the total balance. The transaction is digitally signed by the sender to authorize the transfer. It is then broadcast to the network that groups several transactions in the form of a *block* and validates it. Once accepted, the block gets appended to a public ledger called the *blockchain* and in the process a new coin is minted (as a *block reward*). The consensus on which blocks to append is determined by a *proof-of-work* puzzle. Later the recipient can redeem the funds by creating a new transaction that references the previous output as an input. The public blockchain also prevents the *double spend*—*i.e.*, the ability to spend a coin owned more than once.

An adversary can observe linkages between the addresses involved in transactions on the public blockchain, as evidenced by previous work on Bitcoin, allowing unintended inference of relationships between pseudonymous users [6,13,19,21]. Monero aims to address this privacy issue by requiring the currency to ensure the following two properties (see [22]), paraphrased here:

1. **Unlinkability:** For any two transactions, it should be impossible to deduce that they were sent to the same recipient.
2. **Untraceability:** Given a transaction input, the real output being redeemed in it should be anonymous among a set of other outputs.

In order to guarantee unlinkability, Monero by design introduces the notion of *one-time random addresses* (see Fig. 1). The idea is that each sender of a transaction generates a new one-time random address for the recipient *per output* in a way that only the recipient can spend it using a long-term secret key. If each address is generated using fresh randomness and is used only once, then it should be hard for an adversary to link two addresses. Monero enforces untraceability using a cryptographic primitive called *ring signatures* [7,20]. The primitive allows a sender (the signer)

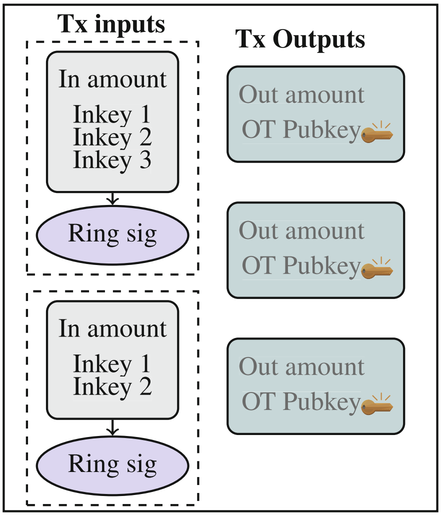


Fig. 1. A schematic representation of a simplified Monero non-RingCT transaction. It has two inputs and three outputs. The first input uses two mix-ins (hence three input keys), while the second one uses one. The sum of all output amounts must be equal to the sum of all input amounts. Ring signatures hide the real input key corresponding to the output being spent.

to anonymously sign the transaction (the message) on behalf of a "ring" or group of other users. As a result, the real output being redeemed remains anonymous amongst the chosen set of outputs (of the same amount) belonging to other users. The extra outputs used to create the ring signature are referred to as the *mix-ins*. As each mix-in is an output of a previous transaction, it is uniquely identified by the corresponding public key. Throughout this paper, we interchangeably use the term output and the public key. The guarantee that Monero aims to achieve through ring signatures is privacy *à la anonymity-set size* [22]. An input using m mix-ins has an anonymity-set size of $m + 1$.

Since January 2017, Monero has further strengthened its privacy guarantees by incorporating *ring confidential transactions* (RingCTs) [16] that also hide the transaction amount preventing several inference attacks. RingCTs are currently optional but are slated to become obligatory starting from September 2017.

Problem Statement. In this work, we focus on Monero's untraceability guarantee. While Monero does provide strong privacy features in theory, it is not known how well it does in real usage scenario, our work attempts to fill this gap. In other words, we aim to study the gap between the *desired anonymity-set size* (from user's perspective) and the *effective anonymity-set size* (from the attacker's perspective) in the presence of a passive adversary. To this end, we develop three attack routines and evaluate them on the Monero blockchain data. Monero designers and developers are aware of the theoretical possibility of the attacks [11,17] and have put in place some measures to mitigate the risks. In light of [11,17], our work has three goals:

1. Quantify through concrete attack strategies both past and existing risks.
2. Demonstrate that the existing measures often fall short in mitigating the risks.
3. Propose better mitigation strategies.

Contributions. We give below our contributions and main findings:

1. One of the most important findings is that over 65% of inputs have an anonymity-set size of one, thereby making them trivially traceable. Attack I leverages this finding to show that such inputs in fact lead to a *cascade effect*, where they affect the untraceability of other inputs with which they have a non-empty anonymity-set intersection. Our evaluation shows that the cascade effect renders another 22% of inputs traceable. The result here is conclusive as Attack I only yields true positives.
2. Attack II exploits the fact that several outputs from a previous transaction are often merged to aggregate funds when creating a new transaction. Such idioms of use leak information on the real outputs being redeemed. On non-RingCTs, Attack II has a true positive rate of 95% when compared to results from Attack I (that provides "ground truth"). We further observe merging of outputs among 1% of RingCTs. We believe that Attack II should extrapolate well even to RingCTs, but the ground truth is not available to scientifically deduce this.

3. Attack III considers an attack based on the temporal analysis of transaction outputs. It considers the most recent output (in terms of block height) in the anonymity-set as the real one being redeemed. Attack III has a true positive rate of 98.5% on non-RingCTs, where the comparison to the Attack I as ground truth is feasible. In light of the results, we propose a better sampling strategy to choose mix-ins. Our method takes into account the actual spending habit of users.

2 Monero Network and Usage Statistics

In this section, we discuss the network and usage statistics of Monero. Our goal is to gain insight into how Monero is used in practice and its ensuing privacy impact. The results obtained here also determine the efficacy of the traceability attacks (presented in Sect. 3). Below, we present the dataset for our statistical analysis. The same dataset will also be used to evaluate the impact of the attack routines.

2.1 Dataset Collection

We acquired the entire Monero history from the first transaction on April 18th 2014 up to and including the last transaction on February 6th 2017. The resulting dataset comprises of a total of 961,463 *non-coinbase transactions*[1] included in 418,910 blocks. The first non-coinbase transaction appears in the block with *height*[2] 110. Our dataset also includes 47,428 RingCTs. The first RingCT appears in the block with height 1,220,517 (on January 10th 2017). We include RingCTs so as to have a more representative dataset. Throughout our analysis, we maintain the raw data in the form of a blockchain and use Monero daemon (`monerod`) to access it [15].

In the remainder of the section, we present results of our statistical analysis on the dataset. Unless otherwise stated, we do not include *coinbase transactions*. Our analysis focuses on three important aspects: (1) available *liquidity*—the number of outputs that have the same amount. A Monero output can only be mixed with other outputs of the same amount. Hence, a low liquidity implies that it may not always be possible to find sufficient number of mix-ins for a given amount. As a consequence, a desired anonymity-set size may not be achieved. (2) the number of mix-ins used. Clearly, the larger it is the higher is the anonymity; (3) number of input and outputs in a transaction. Roughly speaking larger is the number of inputs and outputs, the higher is the probability that Attack II would work, hence we measure these.

[1] Transactions that do not create any new coin. The opposite of coinbase transactions that create new coins.

[2] Height is defined as the number of blocks preceding a particular block on the blockchain.

2.2 Available Liquidity

Since an output can only be mixed with other outputs having the same amount, it is important to have sufficiently large liquidity for each output amount. In order to ensure that sufficient liquidity is maintained, Monero incorporates the notion of denominations. A non-RingCT output amount is $A \times 10^B$, where $1 \leq A \leq 9$ and $B \geq -12$ [11].

There are 1,339,733 different output amounts in the dataset, the largest is 500,000 XMR. We also observe that approximately 85% of all outputs have an amount less than 0.01 XMR (≈0.13 USD [5]). We refer to them as *dust values*. This shows that lower denominations dominate the dataset, and users often transact with small amounts. We further observe that a total of 1,244,165 (93%) output amounts have a frequency of 1. This means that when these outputs have to be redeemed, they cannot be mixed with any other non-RingCT output. The only possible way to create an untraceable transaction redeeming these outputs is to create a RingCT which hides individual mix-in amounts cryptographically (each amount is '0', denoting unknown). These output amounts sum to 1,012,231.3 XMR. Moreover, 84.5% of these outputs correspond to dust values.

Another important observation is regarding the number of output amounts that do not respect the usual denomination format in Monero. We found that 99.98% of these are not denomination compliant. Moreover, 92.8% of these amounts appear only once in the dataset. The total monetary value of the outputs that appear only once and are non-denomination compliant is 12231.27 XMR. The large number of non-denomination compliant output amounts can be attributed to the fact that denomination was not enforced on coinbase transactions in the early years. Moreover, the block reward in Monero is often not denomination compliant by the way it evolves over time.

2.3 Number of Mix-ins

Generally speaking, larger is the number of mix-ins used, the better is the privacy achieved (due to the larger anonymity-set size). However, using a large number of mix-ins leads to a linear increase in the transaction size (in terms of bytes) and a larger transaction size means larger fees. Hence, there is an incentive for users to use low number of mix-ins at the cost of sacrificing privacy. Since March 23^{rd} 2016, Monero enforced a network-wide minimum mix-in of 2.

In fact, a total of 115 different number of mix-ins have been used in our dataset, the minimum and the maximum being 0 and 851 respectively. Table 1 presents the cumulative frequency of the number of mix-ins used in an input. One may observe that lower number of mix-ins, *i.e.*, 0, 1, 2, 3 and 4 correspond to roughly 96% of all mix-ins. Moreover, 65.9% of all inputs have zero mix-ins. Such inputs are traceable by default. In fact, as we show in Sect. 3.1, these inputs may render other inputs traceable as well even if the latter use higher number of mix-ins (through a *cascade effect*).

Table 1. Cumulative frequency of the number of mix-ins (only the first 11 are shown). The last column counts the number of instances where it was possible to choose a higher number of mix-ins.

#Mix-ins	Freq	Cumul. freq. (in %)	Higher #mix-ins possib. (in %)
0	12148623	65.9	10434988 (85.9)
1	707788	69.7	701252 (99.1)
2	2908304	85.5	2902246 (99.8)
3	1313596	92.6	1313530 (99.9)
4	709686	96.5	709681 (99.9)
5	141800	97.3	141797 (99.9)
6	365720	99.2	365718 (99.9)
7	9616	99.3	9614 (99.9)
8	8593	99.3	8593 (100)
9	5369	99.4	5366 (99.9)
10	76524	99.8	76523 (99.9)

There are two possible explanations on why smaller mix-ins dominate the dataset. First, it could be possible that at the time a user creates a transaction, he may not find enough suitable outputs to mix with and hence is forced to choose a lower number of mix-ins. This may indeed happen since a majority of the output amounts are non-denomination compliant. Second, even though enough outputs are available at any given time, users deliberately choose a low number of mix-ins, for instance, to lower transaction fees. In order to distinguish the cause, we also provide in Table 1, the cumulative frequency of the anonymity-set size varying from 1 to 11 (*i.e.*, number of mix-ins between 0 and 10). The table also presents the percentage of cases when it was possible to choose a higher number of mix-ins. In order to compute these data, we include coinbase transactions. A coinbase transaction output cannot be used as a mix-in for 60 blocks due to system constraints. We observe that in the case of inputs with zero mix-ins, 85.9% of them could have been spent using a higher number of mix-ins. As for the rest, over 99% of the inputs could have been spent using a higher number of mix-ins. These results clearly show that users deliberately create a small anonymity-set (by choosing a small number of mix-ins), which could be to avoid paying a larger transaction fee.

In public communication, it has been mentioned by the Monero developers that Poloniex [18] (an exchange) used to pay to its clients using zero mix-ins (to save on transaction fees) and hence must have contributed to the total number of inputs with zero mix-ins. However, the exact percentage of transactions contributed by Poloniex is hard to obtain without an explicit disclosure from its part.

Lastly, we also observe that there are 9 different mix-in values (7.8%) that are unique in the sense that these many mix-ins are used in only one input (across the entire dataset). The corresponding transactions can be attributed to unique users. In other words, the number of mix-ins used may become an identifying trait of Monero users.

2.4 Number of Inputs and Outputs

The existence of denominations in Monero has a direct impact on the number of inputs and outputs that a transaction can have. To see this, consider a user Alice paying 11 XMR to another user Bob. Since 11 is not denomination compliant, Alice cannot pay 11 XMR in a single output. She has to create two outputs for 10 XMR and 1 XMR. Now, when Bob wishes to redeem the funds, he now needs to create a transaction with two inputs instead of one. Bob's transaction now merges the two previous outputs. Output merging in case of Bitcoin is well known to leak information on the owner of the funds [13]. We show through Attack II (Sect. 3.2) that Monero suffers from a similar privacy leakage. In fact, the efficacy of Attack II depends on the number of inputs and outputs that a transaction can have.

To this end, we present in Fig. 2a, the evolution of the number of inputs and outputs per transaction. We observe that the average number of inputs and outputs per transaction are 19 and 17 respectively. Smaller number of inputs and outputs at the tail can be attributed to RingCTs, which made denominations redundant. With RingCTs, the number of outputs in a transaction can be limited to two: one for the payment to the recipient, the other for the change. We now focus on the impact of RingCTs on the number of inputs and outputs. On an average, a RingCT has 3.7 inputs and only 1.2 output. Figure 2b shows the evolution of these values over time. Clearly, with the advent of RingCTs, the number of outputs per transaction was consistently around 2. The number of inputs however does not show a stable and consistent pattern. The peaks in the curve do however tend to lower. It could be due to the existence of denominations from non-RingCTs. As a result, users were still forced to merge a large number of inputs to reach a desired transaction amount.

To summarize the statistical analysis of this section, we observed that over 65% of inputs use zero mix-ins and that the average number of inputs/outputs per transaction is large for non-RingCTs but considerably small for RingCTs. Attack I and Attack II discussed in the following section leverage these findings.

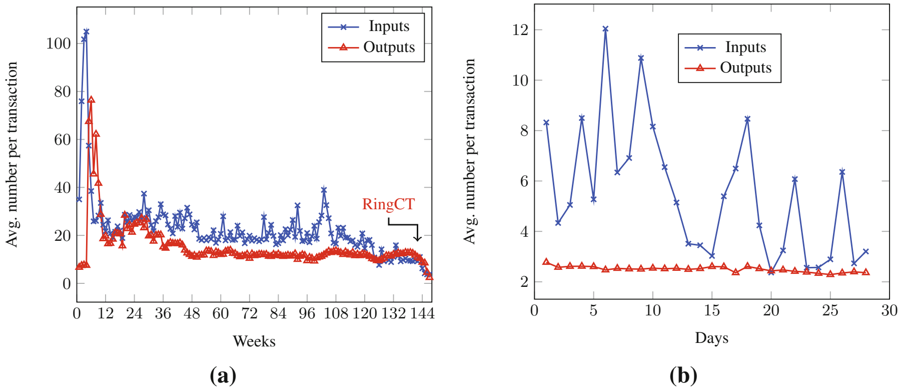


Fig. 2. (a) Average number of inputs and outputs in a transaction. The x-axis represents the number of weeks after the launch of Monero. (b) Average number of inputs and outputs in a RingCT. The x-axis represents the number of days after the launch of RingCTs.

3 Traceability Attacks and Evaluation

In this section, we present three attack strategies on existing Monero transactions. The attacks only assume access to the entire public blockchain. The transactions on the public blockchain are accessible to anyone and hence it justifies our model. We do not assume any active adversary that actively participates in the protocol to undermine the privacy of users. Albeit weak, we show this passive adversary can trace 88% of inputs.

3.1 Attack I: Leveraging Zero Mix-ins

Attack I exploits the presence of inputs spent using zero mix-ins (over 65% in our dataset). Consider the case, where, a user Alice creates a transaction Tx-a with an input that she spends without using any mix-in. Alice's input key can clearly be identified as spent. Now, consider a later transaction Tx-b created by another user Bob who uses Alice's input key as a mix-in for his input. If the number of mix-ins used by Bob is one. Then, any adversary can identify the real input key being spent in Bob's transaction and his input becomes traceable too. More generally, if the number of mix-ins used by Bob is $m > 1$, then the *effective anonymity-set size* now becomes m (instead of the desired size of $m + 1$). A closer look reveals that use of zero mix-in leads to a *cascade effect* where the traceability of an input affects the traceability of another input in a later transaction. Figure 3 schematically presents this cascade effect triggered by Tx-a that uses no mix-in. The cascade effect may not always make an input fully traceable, if for instance, the effective anonymity-set size reduces to a value greater than one.

Our attack routine due to memory restrictions, runs a number of iterations denoted by η. In each iteration, the algorithm makes a pass over the entire blockchain data and finds a set of traceable inputs. In the next iteration, it uses the set of previously found traceable inputs to obtain new set of traceable inputs.

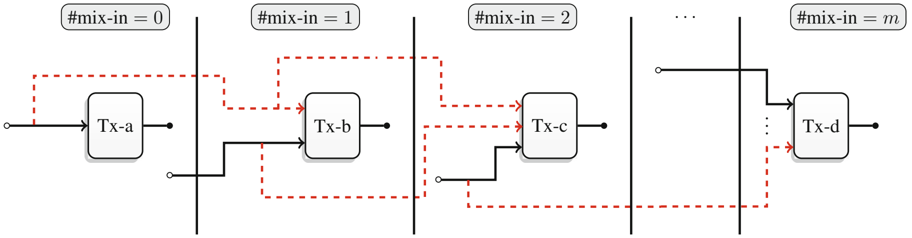


Fig. 3. Attack I: cascade effect due to zero mix-ins. Each transaction has only one input (left of the transaction) and one output (on the right). The number of mix-ins used increases from left to right. Dashed lines represent the input keys identified as a mix-in. Lines in bold are the real input keys being spent.

Results. The impact of Attack I on the traceability of inputs is shown in Figs. 4, 5 and 6. The success of Attack I is due to the fact that 65.9% of inputs do not use any mix-ins and are trivially traceable. This cascades to traceability of another 22% of the inputs, leading to a total of 87.9% of traceable inputs.

In Fig. 4, we present the percentage of traceable inputs for number of mix-ins less than or equal to 10. These together cover 99.8% of all inputs in our dataset. We present histograms for three values of η (1, 3 and 5). With $\eta = 5$, we observe that the set of traceable inputs almost reaches a fixed point. Just after the first iteration ($\eta = 1$), the number of traceable inputs using one mix-in reaches as high as 81%. For $\eta = 5$, this percentage of traceable inputs using one mix-in becomes 87%. The plot also shows the cascade effect as inputs using a high number of mix-ins such as 10 also have a considerable percentage of traceable inputs (27% for $\eta = 5$).

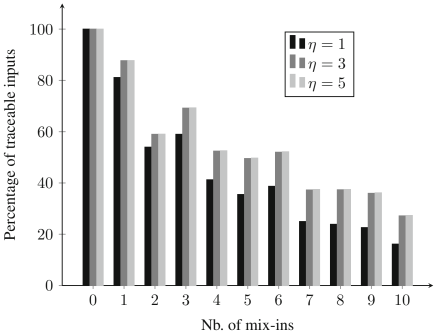
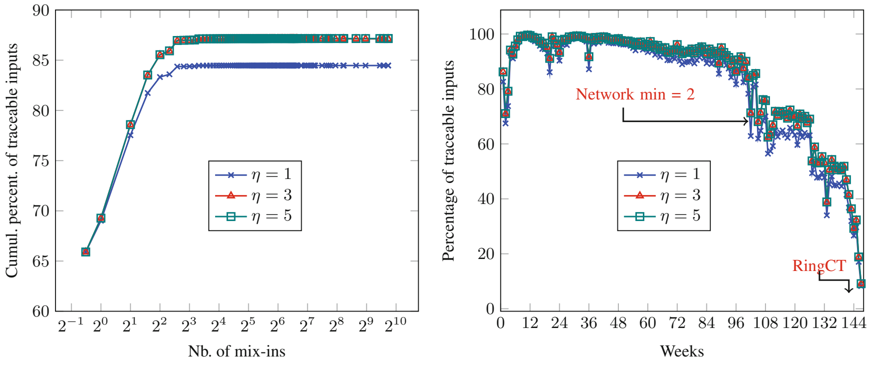


Fig. 4. Results on Attack I: the percentage of traceable inputs as a function of the number of mix-ins.

Figure 5a shows how deep (in terms of number of mix-ins) does the cascade effect propagate. It presents the cumulative percentage of traceable inputs as a function of the number of mix-ins. Clearly, as the number of mix-ins increases, the cascade effect deteriorates and roughly 88% of all inputs become traceable. It is interesting to note that the cascade effect leads to one traceable input that uses 153 mix-ins. This is the largest number of mix-ins that gets affected by the cascade effect.

Figure 5b shows how long (in terms of days) does the cascade effect propagate. It presents the evolution of the percentage of traceable inputs over time grouped by week. Since, the initial transactions did not use any mix-in, a large majority (over 95%) of the inputs were traceable. The maximum percentage of traceable inputs per week is 98.9% (in the 10^{th} week). In fact, the percentage dropped to roughly 62% in the 105^{th} week when the network-wide minimum mix-in of 2 could come into effect. Since then, the percentage of traceable inputs has seen a consistent decline. For the last week, only 8% of all inputs were found to be traceable. Attack I did not find results on RingCTs.

We found several instances where Attack I could not identify a traceable input, but it did nevertheless succeeded in reducing the effective anonymity-set size. Figure 6 presents these findings. For the sake of completeness, it also includes the results presented in Fig. 4. We observe that for $\eta = 5$, roughly 24% of inputs that use 10 mix-ins have an effective anonymity-set size of two. This shows how close Attack I can be in identifying the real input. Moreover, the plot shows that as the number of mix-in increases, the percentage of inputs on



(a) (b)

Fig. 5. Results on traceability using Attack I. (a) The cumulative percentage of traceable inputs as a function of the number of mix-ins. (b) The evolution of the percentage of traceable inputs over time. In the x-axis, we have the number of weeks after the launch of Monero.

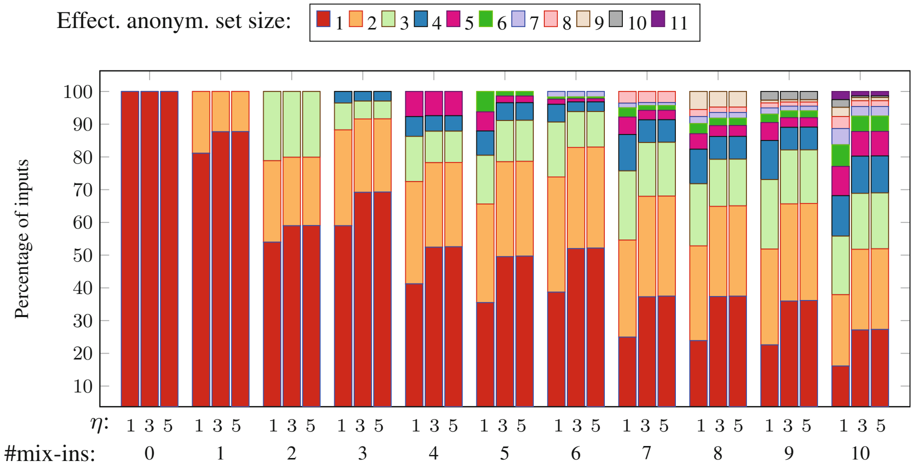


Fig. 6. Results on effective anonymity-set size using Attack I. In x-axis, we have the number of mix-ins from 0 to 10. For each of these mix-ins we plot three stacked bars corresponding to $\eta = 1, 3, 5$. Each stacked bar for a fixed η represents the percentage of inputs that have an effective anonymity-set size between 1 to 11. For instance, for inputs using only 2 mix-ins, and for $\eta = 5$, the percentage of inputs with effective anonymity-set size of 1 is 59%, the percentage of inputs with effective anonymity-set size of 2 is 21% and the percentage of inputs with effective anonymity-set size of 3 is 20%.

which Attack I does not work at all tends to decrease. In fact, for inputs using 10 mix-ins, Attack I does not affect the effective anonymity-set size for only 0.9% of all such inputs.

3.2 Attack II: Leveraging Output Merging

Consider a scenario where a user creates a transaction Tx-a having one input and two outputs O_1, O_2 (Cf. Fig. 7). Without loss of generality, let us assume that only 1 mix-in is used. At a later time, another user creates a transaction Tx-b with two inputs I_1, I_2 and one output. Let us suppose that both the inputs use one mix-in each. The first input I_1 uses one of the outputs O_1 of Tx-a as an input key. Similarly, the second input I_2 uses the other output O_2 of Tx-a as an input key. Attack II then identifies O_1 and O_2 as the real input keys being spent in Tx-b.

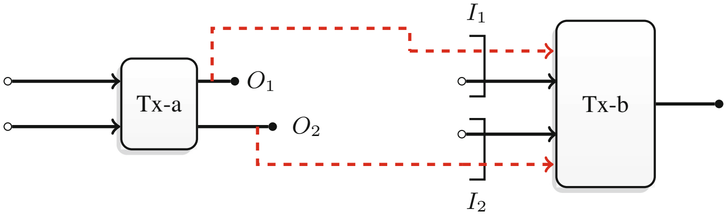


Fig. 7. Attack II. Tx-a is a transaction with one input that uses one mix-in. It has two outputs O_1 and O_2. Tx-b is another transaction that has two inputs I_1 and I_2. Each input again has one mix-in. Both I_1 and I_2 include outputs of Tx-a. According to Attack II, the input keys O_1 and O_2 represented using the dashed line are the real keys being spent in Tx-b.

Attack II functions under the assumption that while creating a transaction, it is unlikely to choose several mix-ins that are outputs of a single previous transaction. Hence, if a transaction includes keys (possibly across several inputs) that are outputs of a single previous transaction, then they are likely to be the real ones being spent. Intuitively, this is true for non-RingCTs, where, due to denominations, several outputs in a transaction may belong to the same recipient. As a result, those outputs will get merged to aggregate funds in a later transaction. Since, mix-ins are randomly chosen, the probability that Attack II gives incorrect result should be small. For RingCTs, denominations are redundant and hence output merging should be less prevalent.

Due to the randomness involved in choosing mix-ins, results obtained from Attack II however cannot always be conclusive and may admit false positives. In this sense, Attack II is weaker than Attack I as the latter does not admit any false positives. As our evaluation later shows, Attack II has a true positive rate of 87.3% on non-RingCTs.

In order to simplify the discussion of Attack II, we refer to transactions of type Tx-a (as in Fig. 7) as a *source* transaction, while, those of type Tx-b as a *destination* transaction. Hence, a destination transaction uses two or more outputs of a source transaction across its inputs. A minimum number of two or more outputs is needed so as to capture the merging of outputs. Attack II may encounter the following scenarios:

- **S1:** It may not find any destination for a given source.
- **S2:** It may find several destinations for a given source.
- **S3:** It may find one (or more) destination for a given source, where the same source output appears in more than one destination input.
- **S4:** It may find one (or more) destination for a given source, where more than one source outputs appear in a single destination input.

S1 means that Attack II failed to yield any result on the given source instance. While, S2 means that the attack has false positives at the transaction level, hence it is hard to ascertain the real destination for a given source. S3 presents the worst case for the attack. It means that the attack has false positives even at the input level. Hence, it is hard to even ascertain the input where the source output was indeed spent. Lastly, S4 yields a set of candidate keys, one of which is probably being spent in the input.

In the following, we conduct experiments to estimate how well Attack II performs. We first measure how frequently the above scenarios occur and then estimate the false positive rate by relying on the results obtained from Attack I as a comparison point. Note that because Attack I yields results only on non-RingCTs, this comparison is limited to non-RingCTs, but the inference extends to RingCTs as well.

Results. Attack II found results on 410,237 different source transactions, which is roughly 43% of all transactions in our dataset. These source transactions also include 636 RingCTs, which is 1% of all RingCTs in the dataset. The low fraction of RingCTs is due to the fact that the average number of inputs and outputs per RingCT is only 3.7 and 1.2 respectively. Recall that Attack II exploits the use of outputs of source transactions in a destination transaction. Hence, a low number of inputs and outputs directly affects the applicability of the attack. Even though, Attack II affects only 1% of RingCTs, it is a serious concern for the Monero developers as exchanges and mining pools are now actively merging RingCT outputs (See Sect. 5 for their feedback).

In Fig. 8a, we present the results obtained on 409,601 non-RingCT sources. Around 60% of all source transactions have only 1 matching destination. The maximum number of destinations found for a source was 146. However, the percentage of source drops exponentially as the number of destinations increases. Similarly, Fig. 8b presents the results obtained on 636 RingCT sources. We observe that a source has at most 3 destinations and a majority of the source transactions (95.1%) have only one destination. The small number of destinations in case of RingCT sources is due to the low number of inputs in RingCTs.

We estimate the accuracy of Attack II by comparing its results with those obtained by Attack I (that yields "ground truth"). Since Attack I does not yield any result on RingCTs, we cannot determine the corresponding accuracy. However, if it performs well on non-RingCTs, then we may extrapolate this result over RingCTs and expect similar accuracy. In order to compare the two attacks, we use the following terms:

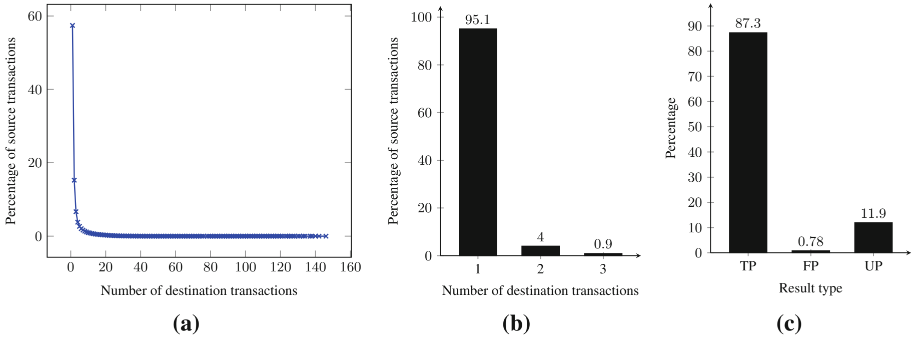


Fig. 8. (a) Result of employing Attack II on non-RingCTs. The x-axis presents all the observed number of destination transactions for a given source transaction. In y-axis, we show the number of source transactions that admit a given number of destination transactions. The number is given as a fraction of 409,601 non-RingCT source transactions. (b) Result of employing Attack II on RingCTs. The x-axis presents all the observed number of destination transactions for a given source transaction. In y-axis, we show the number of source transactions that admit a given number of destination transactions. The number is given as a fraction of 636 RingCT source transactions. (c) Overall observed percentage of TP, FP and UP.

- **True positive (TP):** An input creates a *true positive* if: (a) Attack II identifies a unique key as the one being spent in the input and, (b) the key is the same as the one identified by Attack I. The two attacks are hence in agreement with the conclusion.
- **False positive (FP):** An input creates a *false positive* if all the keys identified as being spent by Attack II were actually found to be spent in a different input by Attack I. In other words, none of the probable keys identified by Attack II was actually the real key being spent in the input. Hence, the two attacks disagree on the real key being spent.
- **Unknown positive (UP):** An input creates an *unknown positive* if at least one of the keys identified by Attack II could not be identified as being spent in any input (of any transaction) by Attack I. The uncertainty comes from Attack I as it does not give ground truth for all inputs.

Now that the terms TP, FP and UP are established, we are ready to present the results on the accuracy of Attack II. The overall accuracy of Attack II computed over all non-RingCT inputs for which it returns a result is given in Fig. 8c. The result shows that Attack II has an overall true positive rate of 87%, while the false positive rate is as low as 0.78%, while the result is inconclusive for around 12% of inputs. The high true positive rate on non-RingCTs clearly demonstrates that it should do equally well even on RingCTs. However, due to the lack of ground truth it is impossible to verify this.

A breakdown of TP, FP and UP as a function of number of mix-ins is given in Fig. 9. The plot shows that as the number of mix-in increases, the percentage of TP decreases, while the percentage of UP increases. Moreover, irrespective of the number of mix-ins used, the number of FP remains very close to 0. All of this is due to the fact that the result of Attack I deteriorates as the number of mix-ins increases and hence it becomes hard to verify the result of Attack II due to a lack of ground truth.

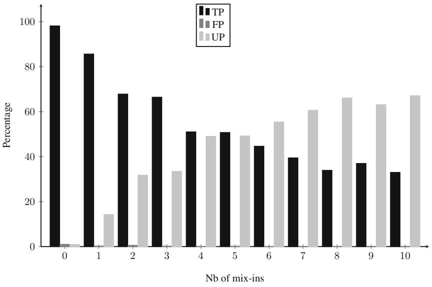


Fig. 9. Observed percentage of TP, FP and UP as a function of number of mix-ins. The result corresponds to non-RingCTs.

Remark 1. Attack II can also be used to break the unlinkability guarantee of Monero. To see this, refer to Fig. 7: If O_1 and O_2 have been determined to be the real input keys being spent in I_1 and I_2, then, they must belong to the same Monero user, hence breaking unlinkability. Since, the focus of this work is on traceability, we do not develop this any further and leave it as a future work.

3.3 Attack III: Temporal Analysis

The third attack leverages the fact that an output does not remain unspent for an infinite time. In general, its probability of being spent should increase with time (eventually becoming 1). Indeed, an output that has been on the blockchain for 100,000 blocks is much more likely to have already been spent than an output that has been on the blockchain for only 100 blocks. In light of this, the Attack III strategy is defined in the following manner: *Given a set of input keys used to create a ring signature, the real key being spent is the one with the highest block height,* where it previously appeared as an output. The strategy applies on both non-RingCTs and RingCTs.

Results. We employed Attack III on our dataset and compared its results with the ones obtained from Attack I ($\eta = 5$). Globally, we observed that Attack III has a true positive rate of 98.1%. This clearly shows that Attack III is very accurate and very often the most recent output is the real one being spent.

Table 2. TP: True Positive and FP: False Positive. Breakdown of traceable inputs obtained using Attack I. In the first row, we show the total number of traceable inputs that employ uniform distribution and the variant of triangular distribution (hence *). The second and third row show the true and false positive rate observed on Attack III.

#Inputs	Uniform dist. (until April 4, 2015)	Triangular dist.* (since April 5, 2015)
	9885810	6174801
TP	99.5%	96%
FP	0.5%	4%

In theory, Attack III can be prevented by mixing with only those outputs that are yet to be redeemed. This would however require the ability to distinguish a spent output from an unspent one. Monero's main aim is to make this hard using a ring signature. In order to circumvent this problem, Monero developers have decided since April 5th 2015 to sample mix-ins from a variant of triangular distribution [23]. The distribution gives higher probability to newer outputs than to ones that are old and hence can potentially mitigate the attack. Prior to April 5th 2015, mix-ins were sampled from a uniform distribution, *i.e.*, each output had the same probability of being a mix-in for any input at any given time. We evaluate how well the current sampling strategy mitigates Attack III. The results are shown in Table 2. While, using the current sampling strategy does help in reducing the number of true positives, the gain over uniform distribution is however marginal, *i.e.*, only 3.5%. Our results clearly show that the existing sampling strategy drastically fails in mitigating Attack III—that is, user spending patterns do not follow the expected (variant of) triangular distribution.

4 Mitigation Strategies and Recommendations

As argued in the Monero Research Lab (MRL) report MRL-004 [11], without a non-interactive zero-knowledge (NIZK) approach, traceability is inevitable. Moreover, it is argued that since NIZK based techniques are computationally intensive, privacy issues should be addressed without employing those techniques. Under these constraints, we propose a mitigation strategy for Attack III that performs better than the variant of the triangular distribution currently employed in Monero. We also propose recommendations to reduce the potential risks associated with Attack I and Attack II.

4.1 Recommendations for Attack I and Attack II

Since Attack I could not find any traceable input on RingCTs, we recommend the Monero development team to make RingCTs obligatory as soon as possible. As long as RingCTs are optional, the possibility of a user creating a non-RingCT cannot be ruled out. Hence, the cascade effect may continue to propagate.

One may argue that enforcing RingCTs must be subject to its acceptance by the community. To this end, we present in Fig. 10, the fraction of RingCTs after its launch. In the fourth day after the launch, the percentage of RingCTs rose as high as 70%. But, in the following two weeks, it remained less than 62%. Looking at the last few days, it appears that RingCTs got well accepted by the Monero users. Indeed, the last day saw a record value of 98%. This is an encouraging result showing the acceptance of RingCTs and hence consolidates our argument that it can be made obligatory.

As for Attack II, it is hard to prevent it completely at the protocol level due to the one-time output addresses. One way to reduce output merging is to warn users when they attempt to do so. Warning can be displayed either when a user creates 2 or more outputs paying the same recipient or when he merges 2 or more outputs from the same transaction. This wallet level solution is certainly not the ideal fix. As a modified version of Attack II can subvert this mitigation measure. To see this, imagine a more general version of Attack II that also considers merging of outputs stemming from different transactions within the same block. We reiterate that information leakage is very hard (if not impossible) to be absolutely prevented without a NIZK based approach.

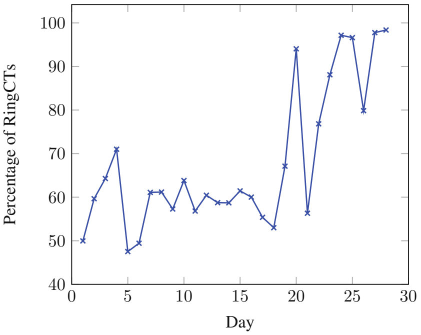


Fig. 10. The percentage of RingCTs among all transactions submitted in a day. The plot only shows data for the days after the launch of RingCT on January 10^{th} 2017.

4.2 Mitigating Threat from Attack III

As we have seen the current sampling strategy for mix-ins fails drastically in preventing temporal analysis. There are two possible strategies towards mitigating the ensuing risks: (a) mimic users' spending behavior or, (b) force mix-ins to be picked according to some "unknown" distribution. Solution (b) can be achieved by employing primitives such as ORAM [8] that will provide cryptographic guarantees on the impossibility to learn information from a passive analysis of the blockchain. However, their applicability to distributed cryptocurrencies seems to require either a trusted party [10] or NIZK proof. We consider them to be important future work. Here, we will focus on (a).

We use the results of Attack I to extract information on when an output (in terms of block height) is created and when it gets spent. We then compute the difference of the two block heights. Since a coinbase output is locked for 60 blocks, its block difference can only be greater than or equal to 60. There is no such restriction on non-coinbase outputs though. Results from Attack I also include a considerable fraction of coinbase outputs (28.9%). This further motivates studying them independently of non-coinbase outputs.

Table 3. Percentage of outputs that remain unspent with a given block interval. The percentage on coinbase is given over coinbase outputs only. Similarly for non-coinbase.

Block difference	Percentage of outputs		
	Coinbase	Non-coinbase	Overall
[0, 10]	0%	0.2%	0.17%
]10, 100]	11.1%	8.4%	9.2%
]100, 1000]	42.6%	22.6%	28.4%
>1000	46.3	68.8%	62.2%

We first observe that users' spending habits can be roughly grouped into four distinct categories characterized by the difference in the block heights (Cf. Table 3). As we see, the percentage of outputs that are spent within the first 1000 blocks is larger among coinbase outputs than among non-coinbase outputs. As a general conclusion, coinbase outputs have a larger probability of being spent quickly once unlocked.

The actual frequency distribution on the entire population is shown in Fig. 11a. The curve clearly shows three distinct clusters identified by the block differences: $[0, 10]$, $]10, 100]$ and >100. Figure 11b presents a breakdown according to coinbase and non-coinbase. We observe that both the curves have a long tail starting from a difference of 1000 blocks.

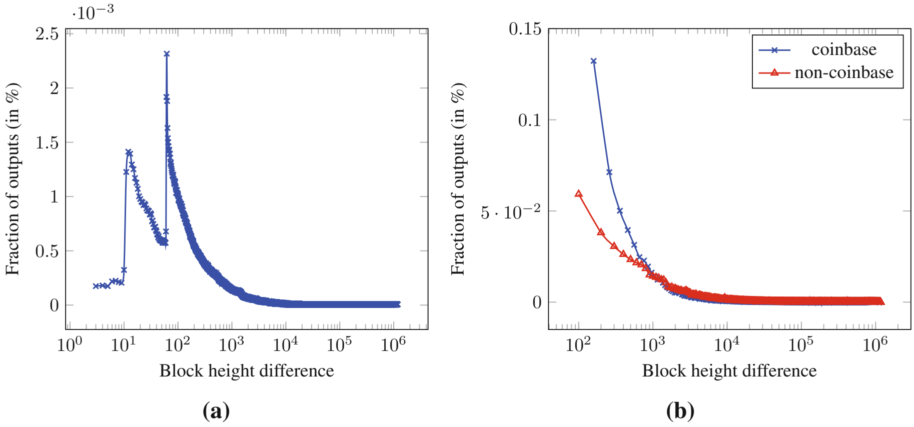


Fig. 11. Spending habit of Monero users. In x-axis, we have the difference between the block height where the output was created and the block height where the output was spent. It measures the duration for which the output remained unspent. In y-axis, we plot the fraction of outputs that share the same difference. Only 1 out of every 100 data points are shown. (a) Combined result on the entire dataset including both coinbase and non-coinbase outputs. (b) Separate results.

In order to mitigate the risks of temporal analysis, we propose replacing the existing sampling method to choose mix-ins. In fact, a better sampling strategy would consist in sampling from the distribution identified by the probability density functions of Fig. 11b. The sampling distribution should moreover depend on whether the output being spent is coinbase or non-coinbase. We estimate the probability density function (PDF) using `descdist()` function of the `R` statistical tool. For each of the dataset, the underlying heuristic based on the Cullen and Frey graph suggests that the PDF is closest to the gamma distribution ($\Gamma(\alpha, \beta)$) with different parameters α and β. For coinbase outputs, the estimated parameters are $\alpha = 0.24, \beta = 7.97 \times 10^{-6}$, while for non coinbase outputs, the estimated parameters are $\alpha = 0.27, \beta = 5.15 \times 10^{-6}$.

Hence, we hereby recommend Monero developers to employ the corresponding PDFs. However, it should be noted that choosing any static distribution renders the system insensitive to social and economic factors that may influence users' spending behavior.

5 Responsible Disclosure and Feedback

As a part of responsible disclosure, we shared our findings both with the Monero development team and the general community. We received varied feedback through diverse channels including but not limited to e-mail, Twitter and Reddit. Monero development team found our result on Attack II insightful:

> "*We hadn't had it [Attack II] accompanied by a model before, i.e., we've been approaching the problem more generally. It's NOT trivially solvable. Because most users will receive additional outputs and need to combine them; the combination betrays a small amount of correlation data.*"

Our findings on Attack II further resulted into the creation of a Github issue on Monero's project repository. The issue was initially termed as a bad user behavior, where, a user merges several outputs together falsely believing it to bring more privacy. The proposed fix was to develop a wallet-based warning system to dissuade users from doing so. The limitations of this mitigation strategy, in particular, the possibility of eventual modifications of the wallet software were also discussed. Later, the issue was identified as non-trivial to solve as output merging is apparently done deliberately by pools and exchanges who want to break up their outputs so payments can be made with locking less change.

Our work has also led to discussions on several Reddit threads accumulating over 50 comments from the Monero community including the development team. Our results on Attack III has also catalyzed the ongoing work on developing better mitigation strategies. Several mitigation measures have been discussed: (1) Sampling mix-ins using our proposed approach of taking into account actual spending behavior, (2) Sampling mix-ins using Zipf distribution, (3) Developing a dynamic sampling procedure, (4) Wallet-specific sampling procedure, *etc.* The ensuing limitations have also been discussed at length. For instance, a dynamic sampling procedure being costly; wallet-specific sampling leading to potential fingerprinting, *etc.*

6 Related Work

Our work is motivated by two prior unpublished works on the privacy analysis of Monero: MRL-001 [17] and MRL-004 [11]. Both of these have been authored by

Monero researchers and developers. The two prior works report on the theoretical possibility of mounting the attacks studied in this paper. However, the impact of the attacks in real usage scenario was unknown. Our work fills this gap by quantifying the existing and the past threat on the blockchain data. It provides a data backed analysis that shows (1) The risks of using no mix-in in practice, (2) How often the risks may arise? (3) How far the cascade effect can propagate? and (4) How the impact has evolved over time? For instance, our work quantifies the severity of cascade effect: it propagates up to 153 mix-ins over a span of three years.

Since the prior works were theoretical and were not backed by real usage behavior, the developers decided on employing a triangular distribution to sample the mix-ins expecting it to mitigate Attack III. Our work shows that a triangular distribution fails drastically in mitigating the attack. This further highlights the difference between the prior work and the new results put forth by our work. Moreover, our work studies the spending habit of users and empirically provides the desired probability distribution function. Our results can be used to improve the current sampling method.

Finally, we are aware of a concurrent work with similar results [14]. The work studies Attack I and Attack III and a countermeasure of Attack III based on real spending behavior. Our work is different from [14] in three contributions: (1) Attack II is not studied in [14] (2) Mitigation for Attack III does not consider coinbase and non-coinbase outputs separately. This may result in a bias as coinbase and non-coinbase outputs have different spending behavior (3) Our network and usage statistics reveal new potential privacy issues in Monero (unknown in [14]) that 7.8% of users are unique in the way they choose their anonymity-set size.

7 Conclusion

This work performs a passive blockchain analysis of Monero and evaluates the efficacy of several attacks on its untraceability guarantees. These attacks are effective as around 88% of inputs are rendered traceable. We also found some traceability results on RingCTs and finally discuss a better strategy (than the one currently employed) to mitigate temporal analysis. Our results hereby reaffirm the weaknesses of anonymity-set size as a privacy metric when implemented in practice. As a future work, we aim to study traceability under active attacks on Monero, where the adversary can take part in the protocol to undermine users' privacy. We are further investigating the use of cryptographic primitives such as zero knowledge proofs and ORAM to strengthen the traceability guarantees beyond the current solutions.

Acknowledgements. Authors would like to thank the anonymous reviewers for their feedback. Amrit Kumar was supported by the research grants R-252-000-560-112 and R-252-000-565-720 from MOE Singapore.

References

1. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, 18–21 May 2014, pp. 459–474 (2014)
2. Meet the Best Performing Digital Currency of 2016: Monero. http://bit.ly/2pVnaJb. Accessed 22 Apr 2017
3. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology: Proceedings of CRYPTO 1982, Santa Barbara, California, USA, 23–25 August 1982, pp. 199–203. Plenum Press, New York (1982)
4. Dash (2017). https://www.dash.org/. Accessed 7 Apr 2017
5. https://www.cryptonator.com/rates/XMR-USD. Accessed 23 Feb 2017
6. Fleder, M., Kester, M.S., Pillai, S.: Bitcoin transaction graph analysis (2015). CoRR abs/1502.01657
7. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 181–200. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71677-8_13
8. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM **43**(3), 431–473 (1996)
9. Jedusor, T.E.: Mimblewimble (2016). Accessed 7 Apr 2017. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt
10. Jia, Y., Moataz, T., Tople, S., Saxena, P.: OblivP2P: an oblivious peer-to-peer content sharing system. In: 25th USENIX Security Symposium, USENIX Security 2016, Austin, TX, USA, 10–12 August 2016, pp. 945–962 (2016)
11. Mackenzie, A., Noether, S., Monero Core Team: Improving obfuscation in the CryptoNote protocol. Research Bulletin MRL-0004, Monero Research Lab, January 2015
12. https://coinmarketcap.com/currencies/monero/. Accessed 22 Apr 2017
13. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, 23–25 October 2013, pp. 127–140. ACM (2013)
14. Miller, A., Moeser, M., Lee, K., Narayanan, A.: An Empirical Analysis of Linkability in the Monero Blockchain (2017). https://arxiv.org/abs/1704.04299
15. https://getmonero.org/knowledge-base/developer-guides/wallet-rpc. Accessed 22 Apr 2017
16. Noether, S., Mackenzie, A.: Monero research lab: ring confidential transactions. Ledger **1**, 1–18 (2016)
17. Noether, S., Noether, S., Mackenzie, A.: A note on chain reactions in traceability in CryptoNote 2.0. Research Bulletin MRL-0001, Monero Research Lab, September 2014
18. Poloniex. https://poloniex.com. Accessed 22 Apr 2017
19. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), PASSAT/SocialCom 2011, Boston, MA, USA, 9–11 October 2011, pp. 1318–1326. IEEE (2011)
20. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). doi:10.1007/3-540-45682-1_32

21. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 6–24. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39884-1_2
22. van Saberhagen, N.: CryptoNote v2.0. Technical report, CryptoNote, October 2013
23. https://github.com/monero-project/monero/commit/f2e8348be0c91c903e68ef582cee687c52411722. Accessed 14 Apr 2017
24. Zerocoin Electric Coin Company: Zcash (2017). https://z.cash/. Accessed 7 Apr 2017