An Empirical Analysis of Linkability in the Monero Blockchain

Andrew Miller *† ‡ Malte Möser § Kevin Lee* Arvind Narayanan §

Abstract

Monero is a privacy-centric cryptocurrency that allows users to obscure their transaction graph by including chaff coins, called "mixins," along with the actual coins they spend. In this report, we empirically evaluate two weaknesses in Monero's mixin sampling strategy. First, about 62% of transaction inputs with one or more mixins are vulnerable to "chain-reaction" analysis — that is, the real input can be deduced by elimination, e.g. because the mixins they include are spent by 0-mixin transactions. Second, Monero mixins are sampled in such a way that the mixins can be easily distinguished from the real coins by their age distribution; in short, the real input is usually the "newest" input. We estimate that this heuristic can be used to guess the real input with 80% accuracy over all transactions with 1 or more mixins. Our analysis uses only public blockchain data, in contrast to earlier attacks requiring active participation in the network [10, 7]. While the first weakness primarily affects Monero transactions made by older software versions (i.e., prior to RingCT), the second weakness is applicable to the newest versions as well. We propose and evaluate a countermeasure derived from blockchain data that can improve the privacy of future transactions.

Working paper disclaimer: This is a draft of work-in-progress. It has not yet been peer-reviewed, and contains preliminary results that may be subject to further revision.

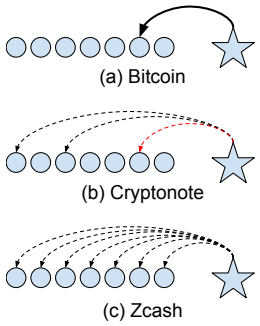
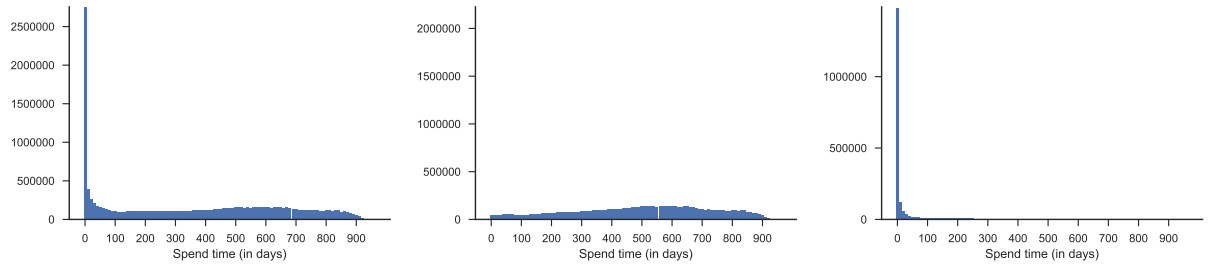


Figure 1: Transactions and linkage in different cryptocurrencies. Consider a new transaction (the star) which spends an available coin (the second circle from the right). In Bitcoin (a), each transaction input explicitly identifies the coin being spent, thus forming a linkage graph. In Zcash (c), each transaction (c) also spends a unique coin, however a zero-knowledge proof conceals any information about which coin is spent [1]. The Cryptonote protocol (b) offers a middle ground, where each transaction input identifies a set of coins, including the real coin along with several chaff coins called "mixins." However, due to a weaknesses in how mixins are sampled by client software, many mixins can ruled out by deduction (Section 3); furthermore, the real input is usually the "newest" one (Section 4).

1 Introduction

Monero is a leading privacy-centric cryptocurrency based on the Cryptonote protocol. As of April 2017 it is the sixth largest cryptocurrency by market capitalization. Whereas Bitcoin, the first and currently largest cryptocurrency, ex-

*University of Illinois at Urbana-Champaign
†Initiative for Cryptocurrencies and Contracts, `initc3.org`
‡Andrew Miller is a consultant to the Zerocoin Electric Coin Company and a board member of the Zcash Foundation.
§Princeton University



(a) Age distribution among all inputs (mixins and real) from blocks 0.9M-1.2M.

(b) Age distribution among all ruled-out mixins from blocks 0.9M-1.2M.

(c) Estimated age distribution of real inputs (recovered from deducible transactions, among blocks 0.9M-1.2M).

Figure 2: Age distributions of Monero mixins. In each graph, the Y-axis is the number of TXOs, and the X-axis represents the time difference between input and referenced output in days. The left graph (a) shows the distribution of all transaction inputs from blocks 0.9M to 1.2M where at least 1000 possible TXOs are available to choose from. Graph (b) shows the age distribution among mixins that can be ruled out. Graph (c) shows that real TXOs (based on deducible transactions, see Section 3) typically conform to a highly-skewed distribution. The disparity between these distributions makes it possible to guess the real input with an estimated 80% accuracy.

plicitly identifies which coin in the transaction graph is being spent, Cryptonote allows users to obscure the transaction graph by including chaff transaction inputs called "mixins."

In this report, we evaluate the impact of two weaknesses of Monero's mixin sampling strategy, which substantially undermine its privacy guarantees. Neither of these weaknesses is entirely new, having been discussed (but not addressed) by Monero developers since as early as 2015. Our work provides the first quantitative assessment of the severity of these weaknesses.

Weakness 1. Many Monero transaction inputs contain deducible mixins, and can be linked to prior transactions via "chain-reaction" analysis.

The Monero software allows users to configure the default number of mixins to include in each transaction. Most Monero transaction inputs (66.09% of all transaction inputs so far) do not contain any mixins at all, but instead explicitly identify the prior TXO they spend, much like ordinary Bitcoin transactions.

Not only do 0-mixin transactions provide no privacy to the users that created them, but also, more worryingly, they present a privacy hazard if other users include these provably-spent transaction outputs as mixins in other transactions too. When the Monero client chooses mixins, it does not take into account whether the potential mixins have already been spent. We find that among Monero transaction inputs with one or more mixins, 62% of these are deducible, i.e. they can be incontrovertibly linked to the prior TXO they spend.

Weakness 2. Monero mixins are sampled from a distribution that does not resemble real transaction inputs, and thus the real inputs can usually be identified. When the Monero client spends a coin, it samples mixins to include by choosing randomly from a triangular distribution over the ordered set of available TXOs with the same denomination as the coin being spent. However, when users spend coins, the coins they spend are not chosen randomly from the blockchain, but instead appear (based on our empirical observations) as though drawn from a highly skewed (e.g., exponential) distribution.

A prescient 2015 post from Monero forum user `EhVedadoOAnonimato` describes this problem quite clearly [4]:

> Actual usage is not random: the age of the outputs change the likelihood of it being used. Newer outputs tend to be used more frequently. You can check that in Bitcoin and probably on other coins.

In Figure 2, we show data from the Monero blockchain

	Not deducible	Deducible	Total
Real input is not newest	15.07% [Est.]	286799 (4.60%)	19.67% [Est.]
Real input is newest	22.61% [Est.]	3597933 (57.72%)	80.33% [Est.]
Total	2349224 (37.68%)	3884732 (62.32%)	6233956 (100%)

Table 1: Linkability of Monero transaction inputs with 1+ mixins (up to block 1236195). Deducible transaction inputs can be linked with complete certainty to the transaction output they spend (see Section 3). Among deducible transaction inputs, the real input is usually the "newest" one, i.e., the one with the smallest offset (see Section 4). Entries marked [Est.] are estimated by extrapolating from deducible transaction inputs, under the assumption that the spend-time distribution of deducible transactions is representative of the distribution overall.

that supports `EhVedadoOAnonimato`'s concern. Figure 2(c) shows the real age of inputs in a representative subset of Monero transactions for which the real input is known (i.e., among deducible transaction inputs as described above). Figure 2(b) shows the age distribution of mixins for which we know that they are not real inputs. When looking at the overall distribution of all inputs, shown in Figure 2(a), the overall distribution can clearly be seen as a mixture of these two distributions. Among transactions for which we have ground truth (i.e., the deducible transaction shown in (c)), we find that *the real input is usually the "newest" input, 92.62% of the time*; based on simulation (Section 4), we estimate this holds for 80% of all transactions. Our results are summarized in Table 1.

Proposed countermeasures. We propose an improved mixin-sampling strategy that can mitigate this weakness for future transactions. Our solution is based on sampling mixins according to a model derived from the blockchain data. We provide evidence that the "spend-time" distribution of real transaction inputs is robust (i.e., changes little over time and across different software versions), and can be well approximated with a simple two-parameter model.

2 Background

Since the inception of Bitcoin in 2009 [9], a broad ecosystem of cryptocurrencies (including Monero) have grown in usage.

A cryptocurrency is a peer-to-peer network that keeps track of a shared append-only data structure, called a blockchain, which represents a ledger of user account balances (i.e., mappings between quantities of currency and public keys held by their current owner). To spend a portion of cryptocurrency, users broadcast digitally-signed messages called transactions, which are then validated and appended to the blockchain.

In slightly more detail, each cryptocurrency transaction contains some number of inputs and outputs; inputs consume coins, and outputs create new coins, conserving the total balance. Each input spends an unspent transaction output (TXO) created in a prior transaction. Together, these form a transaction graph.

The public nature of the blockchain data structure poses a potential privacy hazard to users. Since each transaction is publicly broadcast and widely replicated, any potentially-identifying information can be data-mined for even years after a transaction is committed. Several prior works have developed and evaluated techniques for transaction graph analysis in Bitcoin [11, 12, 8]. Our present work shows that the Monero blockchain also contains a significant amount of linkable data as well.

The function of the peer-to-peer network and consensus mechanism is not relevant to our current work, which focuses only on blockchain analysis; readers unfamiliar with cryptocurrencies can find a comprehensive overview in [3]. Network-based forensic attacks are also known to threaten privacy in Bitcoin [6, 2], but applying this is left for future work.

Cryptonote: Non-interactive Mixing with Ring Signatures. The Cryptonote protocol [13] introduces a novel construction that enables users to obscure their transaction graph, in principle preventing transaction linkability. Instead of explicitly identifying the TXO being spent, a Cryptonote transaction input identifies a set of possible

TXOs, including both the real TXO along with several chaff TXOs, called mixins. Instead of an ordinary digital signature, each Cryptonote transaction comes with a ring signature (a form of zero-knowledge proof) that is valid for one of the indicated TXOs, but that does not reveal any information about which one is real. To prevent double-spending, every input must provide a key image that is unique to the output being spent, and the network must check whether this key image has ever been revealed before.

Several cryptocurrencies are based on the Cryptonote protocol, including Monero, Boolberry, Dashcoin, Bytecoin, and more. [1] We focus our empirical analysis on Monero, since it is currenty the largest and most popular, e.g. it has the fifth largest market cap of all cryptocurrencies, of over $200M. However, we believe our results are applicable to other Cryptonote-based currencies as well.

Choosing Mixin Values. The Cryptonote protocol does not provide an explicit recommendation on how the "mixins" should be chosen. However, the original Cryptonote reference implementation included a "uniform" selection policy, which has been adopted (at least initially) by most implementations, including Monero. Since all the TXOs referenced in a transaction input must have the same denomination, the client software maintains a database of available TXOs, indexed by denomination. Mixins are sampled from this ordered list of available TXOs, disregarding any temporal information except for their relative order in the blockchain.

In principle, it is up to an individual user to decide on a policy for how to choose the mixins that are included along with a transaction. Since it is not a "consensus rule," meaning that miners do not validate that any particular distribution is used, clients can individually tune their policies while using the same blockchain. The Monero command-line interface allows users to specify the number of mixins, with a default of 4.

Over the past several years, Monero's mixin selection policy has undergone several changes, the major ones of which we describe below (the block heights reported correspond to the cutoffs we use in our subsequent analysis):

- Prior to version 0.9.0. (January 1, 2016, Block 892866)
  Prior to the Monero version 0.9.0 release (Hydrogen Helix), mixins were selected uniformly from among the set of all prior TXOs having the same denomination as the coin being spent. As a consequence, earlier outputs would be chosen more often than newer ones.

- After version 0.9.0. (January 1, 2016, Block 892866)
  The Monero version 0.9.0 release (Hydrogen Helix) introduced a new policy for selecting mixins, based on a triangular distribution. In contrast with a uniform distribution, the triangular distribution selection is skewed so as to favor using newer coins rather than old coins as mixins. Based on forum posts from the developers [4], we believe this choice was made under the cognizance that the actual inputs are also likely to be new. However, as can be clearly seen from the blockchain data, the triangular distribution is not adequately skewed enough, significantly less than the real distribution.

  This version also introduced a mandatory minimum number of 2 mixins per transaction input, as recommended by MRL-0001 [10]. This mandatory minimum was enforced after a "hard fork" flag day, which occurred at block 1009827 on March 23, 2016.

- After version 0.10.0. (September 19, 2016)
  The Monero version 0.10.0 release (Wolfram Warptangent) introduced a new RingCT feature, which allows users to conceal the denomination of their coins, avoiding the need to partition the available coins into different denominations. RingCT transactions were not considered valid until after a "hardfork" flag day, which occurred at block 1220516 on January 5, 2017.

  The RingCT feature does not inherently address the linkability concern. However, since RingCT transactions can only include other RingCT transaction outputs as mixins, and since RingCT was deployed after the 2-mixin minimum was established (in verion 0.9.0), there are no 0-mixin RingCT inputs to cause a hazard.

- After version 0.10.1 (December 13, 2016, Block 1201528) The Monero version 0.10.1 release includes a change to the mixin selection policy: now, some mixins are chosen from among the "recent" TXOs (i.e., those created within the last 5 days, called the "recent zone"). Roughly, the policy is to ensure 25 % of the inputs in a transaction are sampled from the recent zone.

[1] See `https://cryptonote.org/coins` for a list of Crypotnote-based currencies.

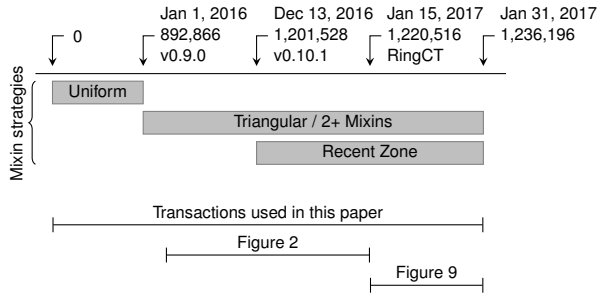


Figure 3: Data considered in our experiment.

In summary, the timeline of data we consider in our analysis is shown in Figure 3.

Transaction Notation. We briefly introduce some notation for describing transaction graphs. For a transaction $\mathtt{tx}$, $\mathtt{tx}.in$ denotes the vector of $\mathtt{tx}$'s transaction inputs, and $\mathtt{tx}.out$ denotes the vector of $\mathtt{tx}$'s transaction outputs. We use subscripts to indicate the elements of input/output vectors, e.g. $\mathtt{tx}.in_1$ denotes the first input of $\mathtt{tx}$. Each Cryptonote transaction input contains a reference to one or more prior transaction outputs. We use array notation to denote the individual references of an input. We used a dashed arrow, $\dashleftarrow$, to denote this relationship, e.g. $\mathtt{tx}_A.out_i \dashleftarrow \mathtt{tx}_B.in_j[m]$ means that m'th reference of the j'th input of transaction $\mathtt{tx}_B$ is a reference to the i'th output of $\mathtt{tx}_B$. Although a Cryptonote transaction input may contain more than one reference, only one input is the *real* reference (known only to the sender), indicated by a solid arrow. Thus $\mathtt{tx}_A \leftarrow \mathtt{tx}_B$ indicates that $\mathtt{tx}_A$ contain an output that is spent by one of the inputs in $\mathtt{tx}_B$.

Transactions included in the blockchain are processed in sequential order; we use $\mathtt{tx}_A < \mathtt{tx}_B$ to indicate that $\mathtt{tx}_A$ occurs before $\mathtt{tx}_B$. Other properties of a transaction are defined as functions, and introduced as needed. For example, $\mathsf{time}(\mathtt{tx})$ refers to the timestamp of the block in which $\mathtt{tx}$ is committed.

Blockchain Analysis Infrastructure. We extracted relevant information from the Monero blockchain, up to block 1236196 (January 31, 2017) and stored it for further analysis in a graph database (9.7GB of data in total).

More specifically, we use the RPC interface of a Monero node (running v0.10.2.1) to extract all relevant data into CSV files, and then pass these files a Neo4j batch importer to import the blockchain data into a Neo4j graph database. We chose Neo4j for its stability and expressive SQL-like query language Cypher. In the near future we will provide a reproducibility kit that includes the source code we used for data collection and analysis.

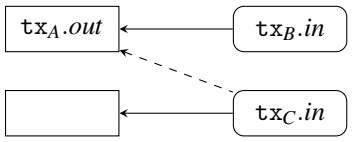


Figure 4: 0-mixins effectively reduce the unlinkability of other transactions: the dashed reference can be ruled out since $\mathtt{tx}_A.out$ must have been spent in $\mathtt{tx}_B.in$.

3 Deducible Monero Transactions

A significant number of Monero transactions do not contain any mixins at all, but instead explicitly identify the real TXO being spent (i.e., resembling Bitcoin transactions). Critically, at the beginning of Monero's history, users were allowed to create zero-mixin transactions that do not contain any mixins at all. Figure 5 shows the fraction of transactions containing zero-mixin inputs over time. As of block 1236196, a total of 12148622 transaction inputs do not contain any mixins, accounting for 66.09% of all transaction inputs overall.

One might think at first that 0-mixin transactions are benign. Transactions with fewer mixins are smaller, and hence cost less in fees; they thus represent an economical choice for an individual who does not explicitly desire privacy for a particular transaction. However, it turns out that the presence of 0-mixin transactions is a hazard that reduces the unlinkability of other transactions, even those that include one or more mixins. For example, suppose a transaction output $\mathtt{tx}_A.out_i$ is spent by a 0-mixin transaction input $\mathtt{tx}_B.in_i$ (i.e. $\mathtt{tx}_A.out_i \dashleftarrow \mathtt{tx}_B.in_j$ where $|\mathtt{tx}_B.in_j| = 1$, from which we can conclude $\mathtt{tx}_A.out_i \leftarrow \mathtt{tx}_B.in_j$). Now, suppose $\mathtt{tx}_A.out_i$ is also included as a mixin in a second transaction with one mixin, $\mathtt{tx}_A.out_i \dashleftarrow \mathtt{tx}_C.in_k[m]$, where $|\mathtt{tx}_C.in_k| = 2$. Since we know that the given output was actually spent in $\mathtt{tx}_B$, we can deduce it is not spent by $\mathtt{tx}_C$ (i.e., $\mathtt{tx}_A.out_i \not\leftarrow \mathtt{tx}_C$),

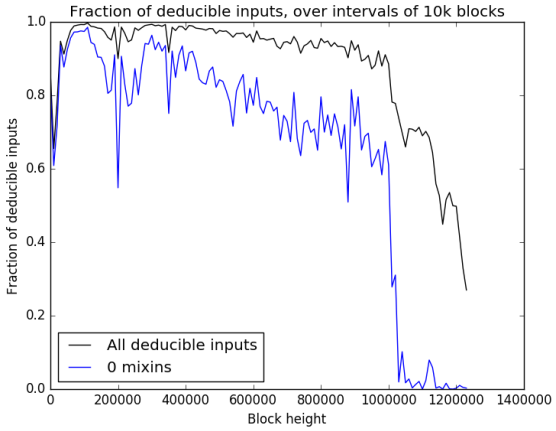


Figure 5: Fraction of transaction inputs that can be deduced over time (averaged over intervals of 10,000 blocks)

```
MATCH (i:Input)-[:REFERENCES]->(o:UnknownSpend)
WITH i, COUNT(o) as cnt WHERE cnt = 1
MATCH (i)-[:REFERENCES]->(o:UnknownSpend)
REMOVE o:UnknownSpend
CREATE (i)-[:SPENDS]->(o)
```

Figure 6: A Neo4J Cypher Query for deducing Monero transaction inputs.

and hence the remaining input of $\mathtt{tx}_C.in_j$ is the real one (cf. Figure 4).

Notice that in this example, it does not matter if the real spend $\mathtt{tx}_B$ occurs after the 0-mixin transaction $\mathtt{tx}_C$ that renders it deducible, i.e., if $\mathtt{tx}_C < \mathtt{tx}_B$. Thus at the time $\mathtt{tx}_C$ is created, it is impossible to know whether a future transaction will render that mixin useless. However, the problem has been exacerbated by the behavior of the Monero client software, which does not keep track of whether a potential mixin has already been clearly spent, and naïvely includes degenerate mixins anyway.

We apply the insight above to build an iterative algorithm, where in each iteration we mark all of the mixin references that cannot be the real spend since we have already deduced that the corresponding output has already been spent in a different transaction. With each iteration, we further deduce the real inputs among additional transaction input sets.

Implementation in Neo4j. In Figure 6 we show a Neo4J Cypher query that implements this algorithm. Initially, we import the Monero blockchain as a graph comprising two types of nodes, `Inputs` and `Outputs`, with directed edges of type `REFERENCES` between them. To each `Output` node we initially add an `UnknownSpend` label, indicating that we have not (yet) deduced whether this has been spent. The query proceeds by identifying each `Input` node i that references only a single `UnknownSpend` `Output` node o; for each such output, we create a `SPENDS` relation between i and o, and remove the `UnknownSpend` label for o. We then iteratively repeat this query until no new `SPEND` edges are created. [2]

Results. In Table 2 we show the results from applying the query described above to Monero blockchain data. As it turns out, approximately 62% of Monero transaction inputs (with 1+ mixins) so far can be linked in this way.

In Figure 7, we show how the vulnerability of Monero transactions to deduction analysis varies with the number of mixins chosen, and in Figure 5 we show how this has evolved over time. We make two observations: first, transactions with more mixins are (as one would hope) significantly less likely to be deducible; and second, even among transactions with the same number of mixins, transactions made with later versions of the software are less vulnerable. This is because at later dates, the 0-mixin transaction outputs accounted for a smaller number of the available mixins to choose from. Surprisingly, we founds tens of thousands of transactions with a large number (10+) of mixins (presumably indicating a high level of desired privacy) that are vulnerable under this analysis.

Comparison with related work on Monero linkability. We note that earlier reports from Monero Research Labs (MRL-0001 [10] and MRL-0004 [7]) have previously discussed concerns about such deduction, called a "chain-reaction," based on similar insights as described above. However, our results paint a strikingly different picture than these.

First, the MRL reports suggested that the vulnerability would require the participation of an attacker, who must

[2] In our actual implementation, we use an optimized (but equivalent) variation of this query that runs in batches and takes advantage of multiple CPU cores.

Table 2: Monero transaction inputs (with 1 or more mixins) where the real input can be deduced.

	Prior to 0.9.0			After 0.9.0, prior to 0.10.1			After 0.10.1, prior to Feb 1, 2017		
	Total	Deducible	(%)	Total	Deducible	(%)	Total	Deducible	(%)
1 mixins	662011	566800	(85.62)	45777	39979	(87.33)	0	–	–
2 mixins	231855	189438	(81.71)	1925778	1216993	(63.19)	713616	294702	(41.30)
3 mixins	487121	368010	(75.55)	714907	484706	(67.80)	107695	52942	(49.16)
4 mixins	191667	138555	(72.29)	403259	209658	(51.99)	106829	24980	(23.38)
5 mixins	61068	25688	(42.06)	76588	44833	(58.54)	3883	817	(21.04)
6 mixins	53820	32245	(59.91)	285833	152044	(53.19)	24695	6942	(28.11)
7 mixins	3191	1529	(47.92)	5010	2190	(43.71)	1351	171	(12.66)
8 mixins	2001	948	(47.38)	5314	2253	(42.40)	1206	204	(16.92)
9 mixins	1388	662	(47.69)	3728	1284	(34.44)	246	43	(17.48)
10+ mixins	54126	10251	(18.94)	52332	14463	(27.64)	7661	1402	(18.30)
Total	1748248	1334126	(76.31)	3518526	2168403	(61.63)	967182	382203	(39.52)
Overall	(62.32)								

at least must have owned some coins used in previous transactions (in their terms, even a "passive" attacker is one that owns prior coins and creates subsequent transactions):

> ... a malicious party with a large number of transaction outputs can cause a chain reaction in traceability by sacrificing their own anonymity.
> (Mackenzie et al. [7])

Our results show this vulnerability is not hypothetical and does not require an active attack, but in fact leads to the linkability of most existing transactions to date.

Second, MRL-0001 [7] provided a simulation analysis predicting that the mandatory 2-mixin minimum (implemented in version 0.9) would "allow the system to recover from a passive attack quite quickly." Our results (Figure 5) show that indeed the fraction of deducible inputs drops steadily after instituting the 2-mixin minimum (from 95% in March 2016 down to 20% in January 2017). However, a significant fraction still remain vulnerable.

In summary, while the MRL reports model a hypothetical "attack" scenario, our results show that this scenario has in fact occurred and we quantify the vulnerability of actual transactions. The earlier MRL reports did not contain an analysis of the blockchain data, and as a result their simulation results are based on unrealistically conservative estimates. For example, while MRL-0001 considered scenarios where the attacker owns up to 50% of the transactions, even at the time of its publication (September 2014, around block 250000) it could have been seen that around 80% of transaction inputs had zero-mixins.

Applicability to current and future transactions using RingCT. The weakness studied in this section is primarily a concern for transactions made in the past, as transactions using the new RingCT transaction option are generally immune. RingCT has been available to users since February 2017, and at the time of writing has already been widely deployed. RingCT transactions are now used by default for the vast majority of new transactions. The reason why these new transactions are immune is not beause of the RingCT mechanism itself, but rather because RingCT was only deployed after the mandatory 2-mixin minimum was enforced. Therefore, RingCT transaction outputs cannot be spent by 0-mixin inputs.

4 Linking With Temporal Analysis

In the previous section, we showed that a majority of Monero transactions inputs can be linked with certainty through logical deduction. In this section, we investigate an entirely unrelated complementary weakness that allows to be linked statically.

4.1 The Guess-Newest Heuristic

Among all the prior outputs referenced by a Monero transaction input, the real one is usually the newest one (the

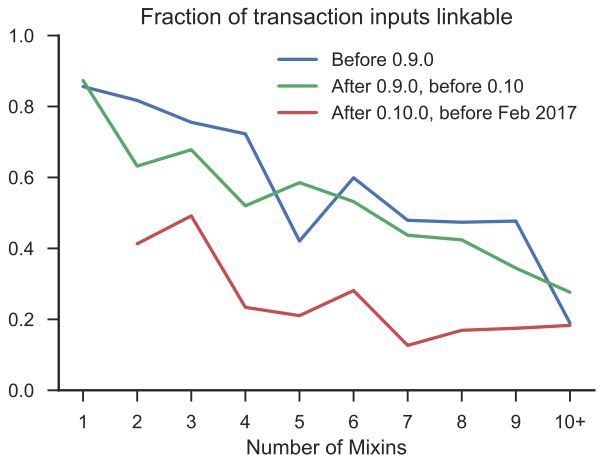


Figure 7: Transaction inputs are less likely to be deducible if they have more mixins and if they are found among later blocks in the Monero blockchain.

one that was committed most recently in the blockchain).

Mixin Offset Distribution. In Cryptonote-based currencies, each transaction input reference is represented as an "offset" into the history of available transaction outputs. Each transaction input can only refer to transaction outputs with a matching denomination (i.e., a 0.01 XMR input can only refer to an 0.01 XMR output). The set of available mixins for a transaction input, $\mathsf{avail}(\mathtt{tx}_B.in_j)$ is defined as the set of outputs with match denomination occurring earlier in the chain.

We define the normalized mixin offset of an input reference as a fraction of the available outputs,

$$\mathsf{norm}(\mathtt{tx}.in_j[m]) = \frac{\mathtt{tx}.in_j[m]}{|\mathsf{avail}(\mathtt{tx}.in_j)|}$$

Figure 2(c) shows the distribution of (normalized) mixin offsets for deducible transaction inputs (i.e., zero-mixin inputs and inputs for which the real TXO can be deduced using the technique from Section 3). As can be seen, this distribution is highly left-skewed; in general users spend coins soon after receiving them (e.g., a user might withdraw a coin from an exchange, and then spend it an hour later). In contrast, the distribution from which (most) mixins are sampled (either a uniform distribution or a triangular distribution, for the most part) includes much older coins with much greater probability.

Cross-validation with Deducible Transactions. In order to quantify the effect of these mismatched distributions, we examine the rank order of the transactions with 1 or more mixins for which we have ground truth (i.e., the deducible transactions from Section 3). Table 3 shows the percentages of deducible transaction inputs for which the real (deduced) reference is also the "newest" reference. It turns out that overall, 92% of the deducible inputs could also be guessed correctly this way.

We note that transactions with more mixins are only slightly less vulnerable to this analysis: even among transaction inputs with 4 mixins (which is currently the default, and planned to become the required minimum in September 2017) the Guess-Newest heuristic still applies to 81% of these transactions.

4.2 Monte Carlo Simulation

The weakness in Monero that leads to input deduction (Section 3) is not directly related to the weakness leading to the Guess-Newest heuristic. However, the effects are partially correlated, i.e., transactions where the real input is the "newest" are also more likely to be "deducible."

To extrapolate from our deducible transactions dataset, we developed a Monte Carlo simulation based on the Monero client code and the blockchain dataset. For each iteration, we first sample a real output (as described below), and then follow the mixin sampling procedure.

Initially, we create a dataset of "spend time" distributions, i.e. the block timestamp difference between when a transaction output is created and when it is spent, i.e.

$$\mathsf{spendtime}(\mathtt{tx}_B.in_j) = \mathsf{time}(\mathtt{tx}_B) - \mathsf{time}(\mathtt{tx}_A) \text{ where } \mathtt{tx}_A \leftarrow \mathtt{tx}_B.$$

This dataset consists of all spend times of 0-mixin transaction inputs as well as deducible inputs.

For each trial in the simulation, we begin by sampling a spend time at random, weighted according to the relative frequency of denominations. Next, for the given denomination, we choose a spend time at random from the dataset, and then choose the offset. Note that while the spend time dataset is based on historic transactions, the resulting "offset" is relative to the head of the chain at our time of

Table 3: Percentage of deducible transaction inputs where the real input is the "newest" input.

	Prior to 0.9.0			After 0.9.0, prior to 0.10.1			After 0.10.1, prior to Feb 1, 2017		
	Deducible	Newest	(%)	Deducible	Newest	(%)	Deducible	Newest	(%)
1 mixins	566800	550145	(97.06)	39979	33999	(85.04)	0	NA	(NA)
2 mixins	189438	177995	(93.96)	1216993	1131586	(92.98)	294702	283978	(96.36)
3 mixins	368010	358330	(97.37)	484706	451565	(93.16)	52942	49668	(93.82)
4 mixins	138555	123905	(89.43)	209658	165006	(78.70)	24980	16136	(64.60)
5 mixins	25688	22881	(89.07)	44833	42687	(95.21)	817	442	(54.10)
6 mixins	32245	28635	(88.80)	152044	124862	(82.12)	6942	5842	(84.15)
7 mixins	1529	1375	(89.93)	2190	1417	(64.70)	171	96	(56.14)
8 mixins	948	850	(89.66)	2253	1593	(70.71)	204	154	(75.49)
9 mixins	662	598	(90.33)	1284	495	(38.55)	43	37	(86.05)
10 mixins	10251	9395	(91.65)	14463	12957	(89.59)	1402	1304	(93.01)
Total	1334126	1274109	(95.50)	2168403	1966167	(90.67)	382203	357657	(93.58)
Overall	(92.62)								

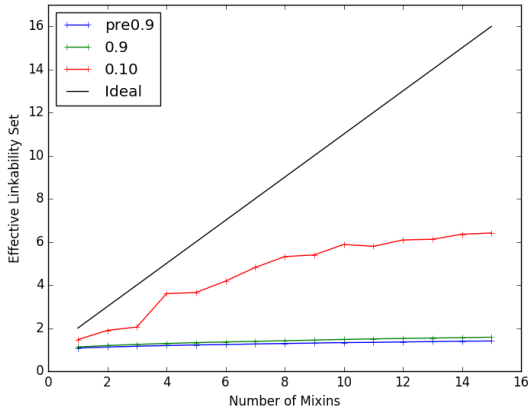


Figure 8: Estimated vulnerability to the Guess-First heuristic for varying sampling policies in Monero, based on Monte Carlo simulations (100k trials each).

writing. After sampling an output to be "spent," we then simulate the Monero mixin-choosing procedure. In the Appendix, we provide the pseudocode for the sampling procedures used in our simulation. We note that our sampling procedures are simplified models, and in particular elide the handling of edge cases such as avoiding "locked" coins that have been recently mined, i.e. "coinbase outputs."

In Figure 8 we show the results of simulating transactions using the sampling rules from the three main Monero versions (pre 0.9, 0.9, and 0.10.1), as applied to the blockchain at height 1236196 (Jan 31, 2017). We report our results in the form of the "effective linkability set," which we define as the inverse of the probability of guessing the correct input reference. In the ideal case (i.e., when all input references are equally likely to be the real one), the linkability set for an input with M mixins is $M+1$.

For versions 0.9.0 and prior, even up to 4 mixins, our simulation suggests that the newest input can be guessed correctly 75% of the time. Across all versions, including more mixins does reduce the effectiveness of this heuristic, although the benefit of each additional mixin is significantly less than the ideal.

Each subsequent update to Monero's mixin sampling procedure has improved the resistance of transactions to the Guess-First heuristic. We note that from developer discussions [5] it appears that this concern has indeed been the motivation for such changes. In particular, the triangular distribution was adopted in place of the uniform distribution in Monero version 0.9 specifically because it chooses new mixins with higher probability than "old" mixins, and version 0.10.1 explicitly introduces a policy that often includes additional "recent" (within 5 days) mixins. However, we believe the magnitude of the problem has so far been underestimated, and even the most recent version only partially mitigates the problem. Under the current default behavior, i.e. 4 mixins and using the 0.10.1 sampling procedure, we estimate that the correct input ref-

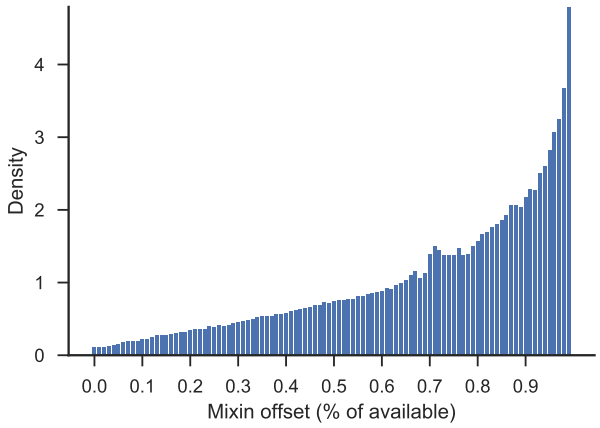


Figure 9: Mixin offset distribution among RingCT transactions (Blocks 1220516 to 1236195).

erence can be guessed with 45% probability (rather than the 20% ideal if all input references were equally likely).

Excluding all of the "deducible" inputs, the average number of mixins among the remaining inputs is 3.53. We therefore estimate that as a lower bound, based on the "0.10.1" line in Figure 8 that at least 60%. Extrapolating from this figure, we estimate that in total the Guess-Newest heurisic correctly identifies 80% of all Monero inputs (Table 1).

4.2.1 Extrapolation to Future Transactions (using RingCT)

As mentioned in Section 2, a new transaction format, called RingCT, has been available since January 5 2017. RingCT transaction inputs can only reference other RingCT transaction outputs. In Figure 9 we show the offset-distribution of all RingCT transactions inputs to date (up to block 1236195, approximately Jan 31, 2017). The characteristic pattern of a triangular distribution mixed with a more heavily left-skewed distribution is plainly visible, though not as pronounced.

Unlike the deducibility weakness, which appears to improve over time (see Figure 7), we hypothesize that the problem of sampling from the wrong distribution becomes *worse* over time, since the set of "old" mixins to choose from grows larger and larger as time goes on.

To validate this hypothesis, we used our simulation strategy to extrapolate the behavior of Monero version 0.10 with RingCT, out to six and twelve months. Our extrapolation is based on an assumption that the transaction rate of Monero remains constant; since RingCT's activation, there have been an average of 8.29 RingCT transaction outputs per block. The results of our experiment (along with a comparison of the countermeasures proposed in the next section) are shown in Figure 11.

5 Improved Mixin Sampling

In this section we discuss a way to improve the sampling procedure. At a high level, the idea is to estimate the actual spend-time distribution, and then sample mixins according to this distribution.

Estimating the spend-time distribution. Monero core developer `smooth` justified the choice of the triangular distribution by the heuristic that this is the "lack of knowledge" distribution [4]. As we are now equipped with data from the Monero blockchain, we thus set about fitting a model to this data.

In Figure 10, we show the spend-time distributions from the Monero blockchain as well as the Bitcoin blockchain. For the Monero data, we split the data by 0-mixin transaction inputs as well as the 1+ mixin inputs, and divide between for which we can deduce the real input.

From this graph, we make the following qualitative observations:

- Observation 1: The spend-time distribution appears invariant with respect to time.
- Observation 2: The spend-time distribution appears the same for 0-mixin as well as 1+ mixin transactions.
- Observation 3: The spend-time distribution has a similar shape in Monero and in Bitcoin, although the spend time is different.

We heuristically determined that the spend time distributions, when plot on a log scale, are closely matched by the shape of a gamma distribution. We used R's `fitdistr` function to fit a gamma distribution to the combined Monero data from deducible transaction inputs (in log seconds). The resulting best-fit distribution has shape parameter 19.28, and rate parameter 1.61. By inspection

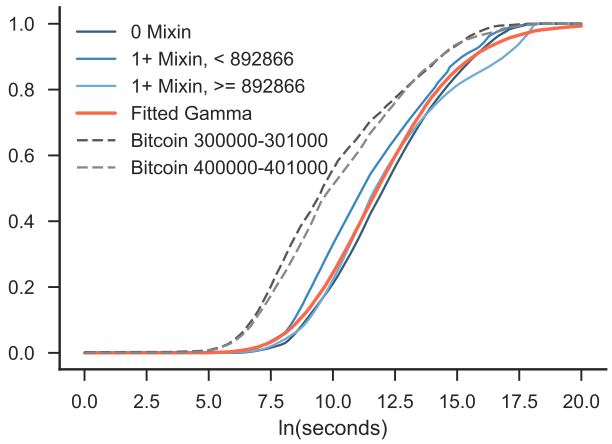


Figure 10: Spend-time distributions in Bitcoin and in Monero (deducible transaction inputs), over multiple time intervals. A gamma distribution (black line) is fitted to the combined Monero data (shape=19.28, rate=1.61).

Algorithm 1: Our proposed mixin sampling scheme.

SAMPLEMIXINS(*RealOut*, *NumMixins*)

```
MixinVector := [];
while |MixinVector| < BaseReqMixCount do
    t ← Exp(GammaSample(shape=19.28,
      rate=1.61));
    if t > CurrentTime then
        continue;
    Let B be the block containing at least one RingCT
      output, with timestamp closest to CurrentTime-t;
    i ← uniformly sampled output among the RingCT
      outputs in B if i ∉ MixinVector and i ≠
      RealOut.idx then
        MixinVector.append(i);
return sorted(MixinVector + [RealOut.idx]);
```

(Figure 10), this appears to accurately fit the overall Monero spend-time distribution.

Sampling mixins according to the spend-time distribution. To make use of the spend-time distribution described above, we next need to identify a way of sampling transaction output indices that matches the ideal spend-time distribution as closely s possible. Our proposed mechanism is to first sample a target timestamp directly from the distribution, and then find the nearest block containing at least one RingCT output. From this block, we sample uniformly among the transaction outputs in that block. This procedure is defined Algorithm 1.

Evaluating our improved sampling strategy. To quantify the effectiveness of our new sampling scheme, we turn again to the Monte Carlo simulation described in the previous section. Figure 11 shows the effective linkability set under our proposed mixin sampling routine, extrapolated out six and twelve months.

Our scheme performs very close to ideal when applied to actual blockchain data, even for large numbers of mixins. At the default setting of 4 mixins, our method nearly doubles the effective linkability set; for large numbers of mixins, our improvement is nearly four times better. In our extrapolated dataset, our method performs slightly worse than ideal at 6 months and 12 months out, although still much better than the current method, and staying within 75% of the ideal.

Daemon privacy. In the Monero implementation, the wallet software runs as a separate process that communicates with the `monerod` full node. Only the wallet has access to the user's keys, but it does not have access to the entire blockchain dataset. We note that the current Monero mixin sampling routines are designed in part to protect against eavesdropping from the Monero daemon: the wallet generates a list of output indices in sorted order (including the index of the actual output being spent) and requests all the corresponding public keys from the daemon. Our proposed solution requires searching the blockchain database for the nearest block matching a target timestamp. Thus to implement our proposal would require expanding the wallet so that it maintains a dataset of each block header, along with the number of RingCT outputs included in each block. In future work we plan to evaluate the overhead this would impose on the wallet software.

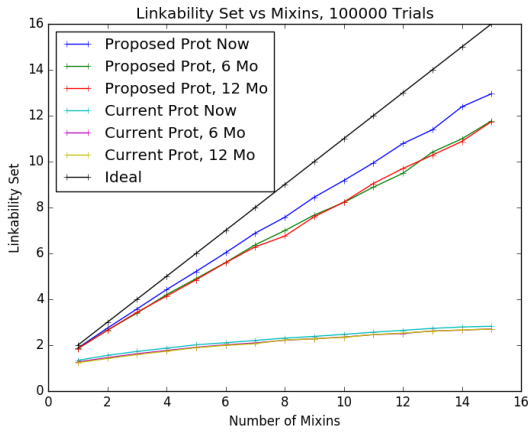


Figure 11: Projection of Guess-First vulnerability, and the improvement due to our proposed scheme.

6 Discussion and Recommendations

We have identified two weaknesses in Monero's mixin selection policy, and shown that they pose a significant risk of linkability to early Monero transactions. In light of these findings, we make the following three recommendations to the Monero developer community:

Recommendation 1. The mixing sampling distribution should be modified to closer match the real distribution. We have provided evidence that the strategy by which Monero mixins are sampled results in a different time distribution than real spends, significantly undermining the unlinkability mechanism. To correct this problem, the mixins should ideally be sampled in such a way that the resulting time distributions match.

A report from Monero Research Labs cited the difficulty of frequently tuning parameters based on data collection (especially since the data collection mechanism itself becomes a potential target for an attacker hoping to alter the parameters) [7].

Fortunately, we provide preliminary evidence that the distribution of "spend-times" changes very little over time (and in fact is similar even in different cryptocurrencies). Hence we recommend a sampling procedure based on a model of spending times derived from blockchain data, as discussed in Section 5.

Recommendation 2: The Monero community should engage in further data-backed analysis of privacy claims. We have examined the vulnerability of Monero's sampling strategies to simple linking techniques, using data from the publicly available blockchain dataset, and proposed a countermeasure that we show thwarts these techniques. However, there remain many possible analysis avenues, including network analysis, side channels, and more sophisticated statistical methods. These have been discussed informally within the community [7], but not yet substantiated with data. We suggest that future changes to the protocol should be validated with such analysis prior to deployment.

Recommendation 3: Monero users should be warned that their prior transactions are likely vulnerable to linking analysis. A significant fraction (62%) of Monero transactions with one or more mixins are deducible (i.e., contain only deducible mixins), and therefore can be conclusively linked. Furthermore, we estimate that among all transaction inputs so far, the "GuessNewest" heuristic can be used to identify the correct mixin with 80% accuracy.

Complementing this report, we have launched a block explorer website, `http://monerolink.com`,[3] which displays the linkages between transactions inferred using our techniques. We recommend additionally developing a wallet tool that users can run locally to determine whether their previous transactions are vulnerable.

Ethics. We released an initial draft of this report to Monero core developers, and intend to support the development and evaluation of countermeasures. However, we believe it is in the best interests of Monero users that we publish this report immediately, without waiting for countermeasures to be deployed. One reason for our decision is that the data from the Monero blockchain is public and widely replicated, and thus delaying this report would not mitigate post-hoc analysis, which can be carried out at any future time. Second, countermeasures built into future versions of the Monero software would not affect the vulnerability of

[3] We do not plan to maintain this website for more than a year; we will therefore update the online version of this article to point to a snapshot on Internet Archive.

transactions occurring between the time of our publication and the deployment of such future versions.

Acknowledgement. Andrew Miller is supported in part by the Initiative for Cryptocurrencies and Contracts. Arvind Narayanan is supported by NSF Awards CNS-1421689 and CNS-1651938.

References

[1] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP)*, pages 459–474. IEEE, 2014.

[2] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[3] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, May 2015.

[4] EhVedadoOAnonimato. `https://forum.getmonero.org/20/general-discussion/2361/question-on-mixin-selection`, September 2015.

[5] EhVedadoOAnonimato. `https://forum.getmonero.org/6/ideas/2372/using-time-neighbors-in-mixin-selection-in-order-to-solve-temporal-associations`, September 2015.

[6] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014.

[7] Adam Mackenzie, Surae Noether, and Monero Core Team. MRL-0004: Improving obfuscation in the cryptonote protocol. `https://lab.getmonero.org/pubs/MRL-0004.pdf`, January 2015.

[8] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan 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.

[9] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[10] Surae Noether, Sarang Noether, and Adam Mackenzie. MRL-0001: A note on chain reactions in traceability in cryptonote 2.0. `https://lab.getmonero.org/pubs/MRL-0001.pdf`, September 2014.

[11] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[12] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[13] Nicolas van Saberhagen. Cryptonote v 2. 0. *HYPERLINK https://cryptonote. org/whitepaper. pdf*, 2013.

A Detailed Description of our Models of Monero Sampling Routines

In Algorithms 2,3,4 we give pseudocode for the mixin selection procedures used in our simulation (Section 4). The changes between each successive version are highlighted in blue. We note that these are simplified models of the mixin sampling behavior in the Monero client, in particular they elide over edge cases that avoid spending "locked" coins that have recently been mined. The full code listing of the Monero client can be found on the Monero git repository.[4] [5]

Algorithm 2: SAMPLEMIXINS(*RealOut*, *NumMixins*)
[vPre0.9.0]

Let TopGIdx be the index of the most recent transaction output with denomination *RealOut.amount*;
BaseReqMixCount := $\lfloor (NumMixins + 1) \times 1.5 + 1 \rfloor$;
MixinVector := [];
while |MixinVector| < BaseReqMixCount **do**
    $i \leftarrow$ `UniformSelect(0, TopGIdx);`
    if $i \notin$ MixinVector and $i \neq$ RealOut.*idx* **then**
        MixinVector.append(i);
Let FinalVector be a uniform random choice of *NumMixins* elements from MixinVector;
return *sorted*(FinalVector + [RealOut.idx]);

Algorithm 3: SAMPLEMIXINS(*RealOut*, *NumMixins*)
[v0.9.0]

Let TopGIdx be the index of the most recent transaction output with denomination *RealOut.amount*;
BaseReqMixCount := $\lfloor (NumMixins + 1) \times 1.5 + 1 \rfloor$;
MixinVector := [];
while |MixinVector| < BaseReqMixCount **do**
    $i \leftarrow$ `TriangleSelect(0, TopGIdx);`
    if $i \notin$ MixinVector and $i \neq$ RealOut.*idx* **then**
        MixinVector.append(i);
Let FinalVector be a uniform random choice of *NumMixins* elements from MixinVector;
return *sorted*(FinalVector + [RealOut.idx]);

Algorithm 4: SAMPLEMIXINS(*RealOut*, *NumMixins*)
[v0.10.1]

Let TopGIdx be the index of the most recent transaction output with denomination *RealOut.amount*;
BaseReqMixCount := $\lfloor (NumMixins + 1) \times 1.5 + 1 \rfloor$;
Let RecentGIdx be the index of the most recent transaction output prior to 5 days ago with denomination *RealOut.amount* prior to 5 days ago;
BaseReqRecentCount := MAX(1, MIN(
TopGIdx - RecentGIdx + 1,
BaseReqMixCount $\times$ RecentRatio));
if RealOut.*idx* $\geq$ RecentGIdx **then**
    BaseReqRecentCount -= 1
MixinVector := [];
while |MixinVector| < BaseReqRecentCount **do**
    $i \leftarrow$ `UniformSelect(RecentGIdx, TopGIdx);`
    if $i \notin$ MixinVector and $i \neq$ RealOut.*idx* **then**
        MixinVector.append(i);
while |MixinVector| < BaseReqMixCount **do**
    $i \leftarrow$ `TriangleSelect(0, TopGIdx);`
    if $i \notin$ MixinVector and $i \neq$ RealOut.*idx* **then**
        MixinVector.append(i);
Let FinalVector be a uniform random choice of *NumMixins* elements from MixinVector;
return *sorted*(FinalVector + [RealOut.idx]);

[4] https://github.com/monero-project/monero/blob/v0.9.0/src/wallet/wallet2.h#L570

[5] https://github.com/monero-project/monero/blob/v0.10.0/src/wallet/wallet2.cpp#L3605