March 2024 Suspected Black Marble Flooding Against Monero: Privacy, User Experience, and Countermeasures

Draft v0.3

Rucknium

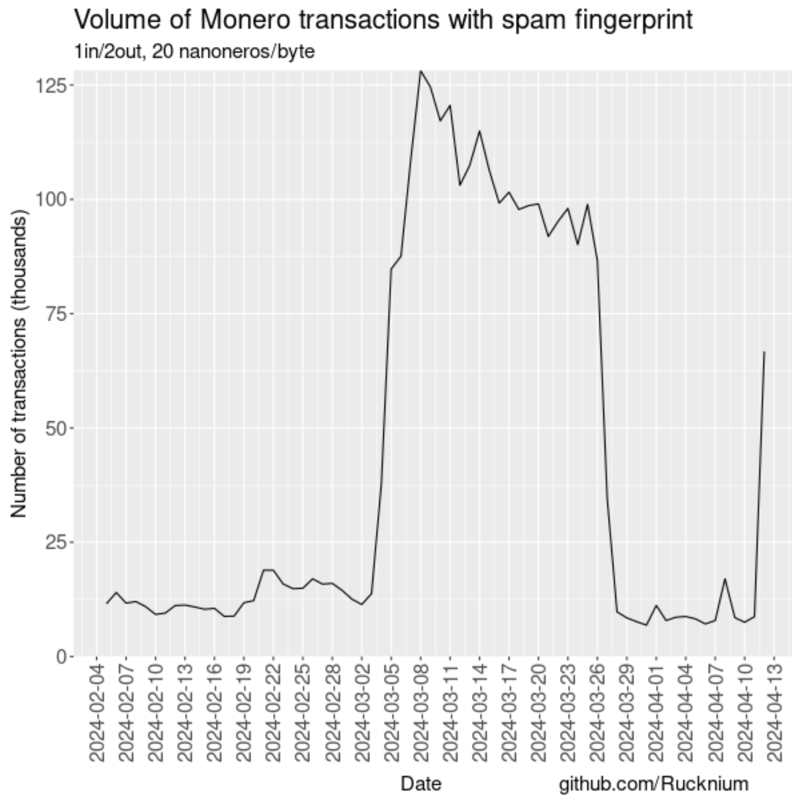
October 9, 2024

Abstract

On March 4, 2024, aggregate Monero transaction volume suddenly almost tripled. This note analyzes the effect of the large number of transactions, assuming that the transaction volume is an attempted black marble flooding attack by an adversary. According to my estimates, mean effective ring size has decreased from 16 to 5.5 if the black marble flooding hypothesis is correct. At current transaction volumes, the suspected spam transactions probably cannot be used for large-scale "chain reaction" analysis to eliminate all ring members except for the real spend. Effects of increasing Monero's ring size above 16 are analyzed.

1 March 4, 2024: Sudden transaction volume

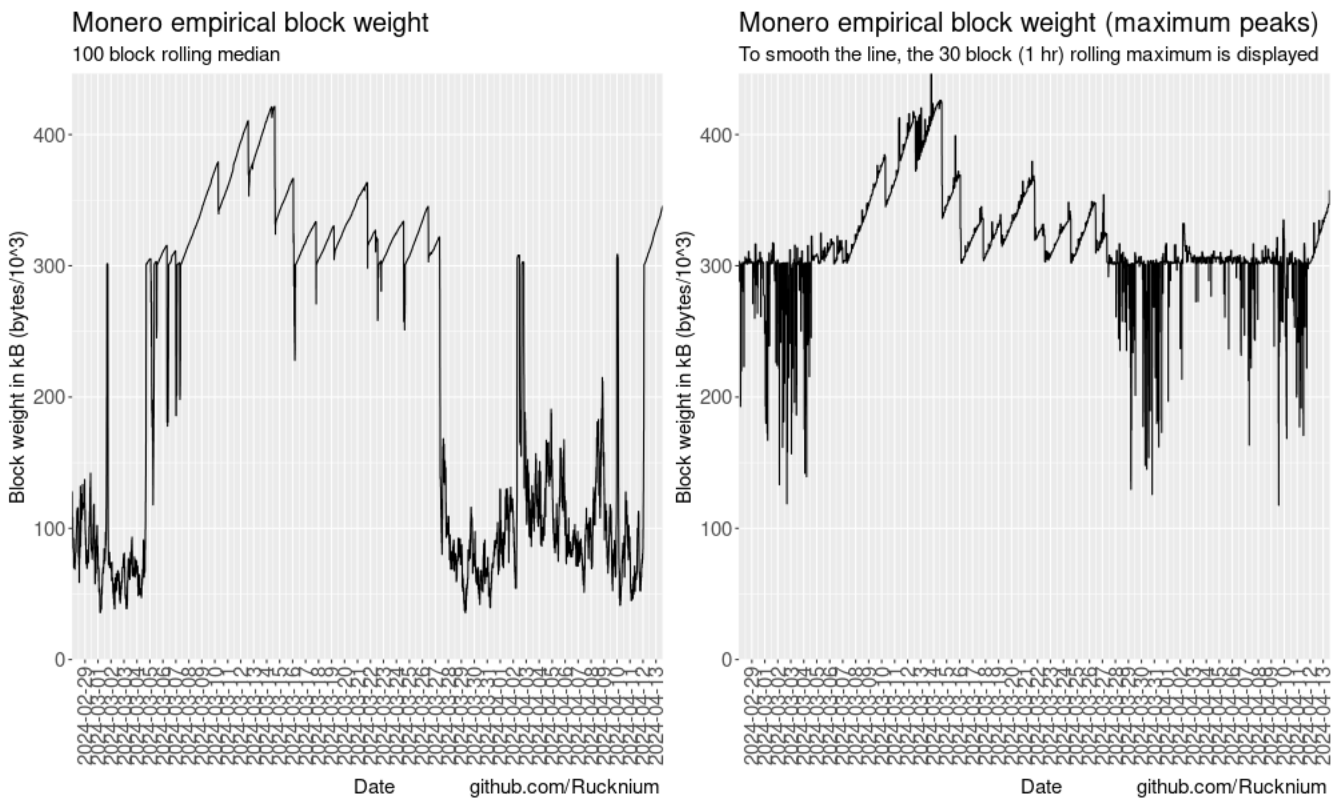
Figure 1: Volume of Monero transactions with spam fingerprint



On March 4, 2024 at approximately block height 3097764 (15:21:24 UTC), the number of 1input/2output minimum fee (20 nanoneros/byte) transactions sent to the Monero network rapidly increased. Figure 1 shows daily volume of this type of transaction increasing from about 15,000 to over 100,000.

The large volume of these transactions was enough to entirely fill the 300 kB Monero blocks mined about every two minutes. Monero's dynamic block size algorithm activated. The 100 block rolling median block size slowly increased to adjust for the larger number of transactions that miners could pack in blocks. Figure 2 shows the adjustment. The high transaction volume raised the 100 block median gradually for period of time. Then the transaction volume reduced just enough to allow the 100 block median to reset to a lower level. Then the process would restart. Block sizes have usually remained between 300 kB and 400 kB. Occasionally, high-fee transactions would allow miners to get more total revenue by giving up some of the 0.6 XMR/block tail emission and including more transactions in a block. The "maximum peaks" plot shows this phenomenon.

Figure 2: Monero empirical block weight



The sudden transaction volume rise may originate from a single entity. The motive may be spamming transactions to bloat the blockchain size, increase transaction confirmation times for real users, perform a network stress test, or execute a black marble flooding attack to reduce the privacy of Monero users. I will focus most of my analysis on the last possibility.

2 Literature review

The very first research bulletin released by the Monero Research Lab described black marble transaction flooding. [Noether et al., 2014] points out that the ring signature privacy model requires rings to contain transaction outputs that are could be plausible real spends. If a single entity owns a large share of outputs (spent or not), it can use its knowledge to rule out ring members in other users' transactions that cannot be the real spend. Since the entity knows that itself did not spend the output(s) in a particular ring, the effective ring size that protects other users' privacy can be reduced — even to an effective ring size of 1 when the entity knows the real spend with certainty. Rings with known real spends can be leveraged to determine the real spend in other rings in a "chain reaction" attack.

[Noether et al., 2014] gave the name "black marble" to the outputs owned by an anti-privacy adversary since they modeled the problem using a marble draw problem with a hypergeometric distribution. When a specific number of marbles are drawn *without* replacement from an urn containing a specific number of white and black marbles, the hypergeometric distribution describes the probability of drawing a specific number of black marbles. In my modeling I use the binomial distribution, which is the same as the hypergeometric except marbles are drawn *with* replacement. The binomial distribution makes more sense now ten years after [Noether et al., 2014] was written. The total number of RingCT outputs on the blockchain that can be included in a ring is over 90 million. The hypergeometric distribution converges to the binomial distribution as the total number of marbles increases to infinity. Moreover, Monero's current decoy selection algorithm does not select all outputs with equal probability. More recent outputs are selected with much higher probability. The hypergeometric distribution cannot be used when individual marbles have unequal probability of being selected.

[Chervinski et al., 2021] simulates a realistic black marble flood attack. They consider two scenarios. The adversary could create 2input/16output transactions to maximize the number of black marble outputs per block or the adversary could create 2input/2output transactions to make the attack less obvious. The paper uses Monero transaction data from 2020 to set the estimated number of real outputs and kB per block at 41 outputs and 51 kB respectively. The nominal ring size at this time was 11. The researchers simulated filling the remaining 249 kB of the 300 kB block with black marble transactions. A "chain reaction" algorithm was used to boost the effectiveness of the attack. In the 2in/2out scenario, the real spend could be deduced (effective ring size 1) in 11% of rings after one month of spamming black marbles. Later I will compare the results of this simulation with the current suspected spam incident.

[Krawiec-Thayer et al., 2021] analyze a suspected spam incident in July-August 2021. Transactions' inputs, outputs, fees, and ring member ages were plotted to evaluate evidence that a single entity created the spam. The analysis concluded, "All signs point towards a single entity. While transaction homogeneity is a strong clue, a the [sic] input consumption patterns are more conclusive. In the case of organic growth due to independent entities, we would expect the typically semi-correlated trends across different input counts, and no correlation between independent users' wallets. During the anomaly, we instead observed an extremely atypical spike in 1–2 input txns with no appreciable increase in 4+ input transactions."

TODO: A few papers like [Ronge et al., 2021, Egger et al., 2022] discuss black marble attacks too.

3 Black marble theory

The binomial distribution describes the probability of drawing x number of "successful" items when drawing a total of n items when the probability of a successful draw is p. It can be used to model the number of transaction outputs selected by the decoy selection algorithm that are not controlled by a suspected adversary.

The probability mass function of the binomial distribution with $n \in \{0, 1, 2, \ldots\}$ number of draws and $p \in [0, 1]$ probability of success is

$$f(x,n,p) = \binom{n}{x} p^x (1-p)^{n-x}, \text{ where } \binom{n}{x} = \frac{n!}{x!(n-x)!} \tag{1}$$

The expected value (the theoretical mean) of a random variable with a binomial distribution is np.

Monero's standard decoy selection algorithm programmed in `wallet2` does not select outputs with equal probability. The probability of selecting each output depends on the age of the output. Specifics are in [Rucknium, 2023]. The probability of a single draw selecting an output that is not owned by the adversary, p_r, is equal to the share of the probability mass function occupied by those outputs: $p_r = \sum_{i \in R} g(i)$, where R is the set of outputs owned by real users and $g(x)$ is the probability mass function of the decoy selection algorithm.

3.1 Spam assumptions

There is some set of criteria that identifies suspected spam. The early March 2024 suspected spam transactions: 1) have one input; 2) have two outputs; 3) pay the minimum 20 nanoneros per byte transaction fee. The normal volume of these transactions produced by real users must be estimated. The volume in excess of the normal volume is assumed to be spam. I followed this procedure:

1. Compute the mean number of daily transactions that fit the suspected spam criteria for the four weeks that preceded the suspected spam incident. A separate mean was calculated for each day of the week (Monday, Tuesday,...) because Monero transaction volumes have weekly cycles. These volume means are denoted $v_{r,m}, v_{r,t}, v_{r,w}, \ldots$ for the days of the week.

2. For each day of the suspected spam interval, sum the number of transactions that fit the suspected spam criteria. Subtract the amounts found in step (1) from this sum, matching on the day of the week. This provides the estimated number of spam transactions for each day: $v_{s,1}, v_{s,2}, v_{s,3}, \ldots$

3. For each day of the suspected spam interval, randomly select $v_{s,t}$ transactions from the set of transactions that fit the suspected spam criteria, without replacement. This randomly selected set is assumed to be the true spam transactions.

4. During the period of time of the spam incident, compute the expected probability p_r that one output drawn from the `wallet2` decoy distribution will select an output owned by a real user (instead of the adversary) when the wallet constructs a ring at the point in time when the blockchain tip is at height h. The closed-form formula of the `wallet2` decoy distribution is in [Rucknium, 2023].

5. The expected effective ring size of each ring constructed at block height h is $1 + 15 \cdot p_r$. The coefficient on p_r is the number of decoys.

Figure 3 shows the results of this methodology. The mean effective ring size settled at about 5.5 by the fifth day of the large transaction volume. On March 12 and 13 there was a large increase in the number of 1input/2output transactions that paid 320 nanoneros/byte (the third fee tier). This could have been the spammer switching fee level temporarily or a service that uses Monero increasing fees to avoid delays. I used the same method to estimate the spam volume of these 320 nanoneros/byte suspected spam. The 1in/2out 320 nanoneros/byte transactions displaced some of the 1in/2out 20 nanoneros/byte transactions because miners preferred to put transactions with higher fees into blocks. Other graphs and analysis will consider only the 1in/2out 20 nanoneros/byte transactions as spam unless indicated otherwise.

Figure 3: Estimated mean effective ring size

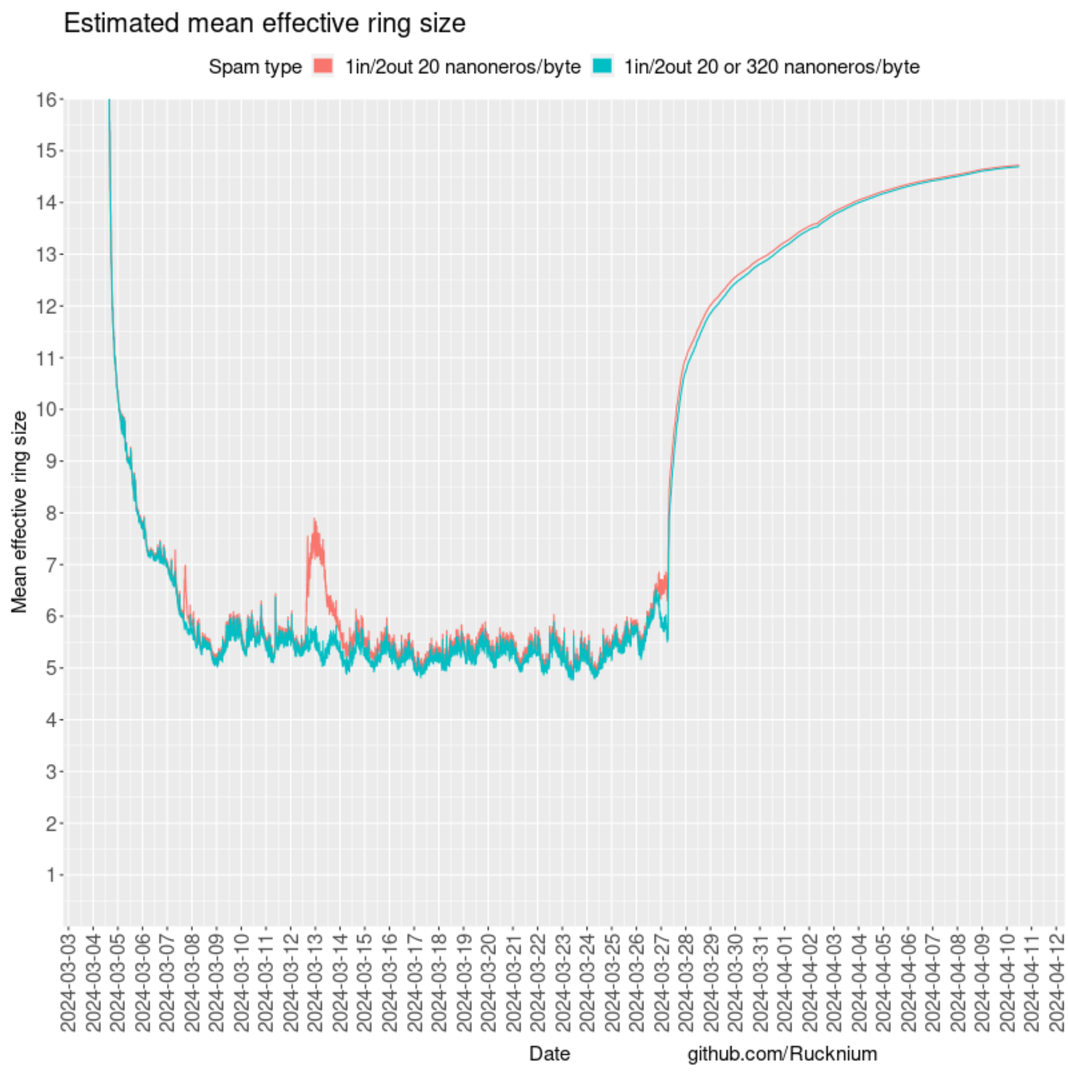
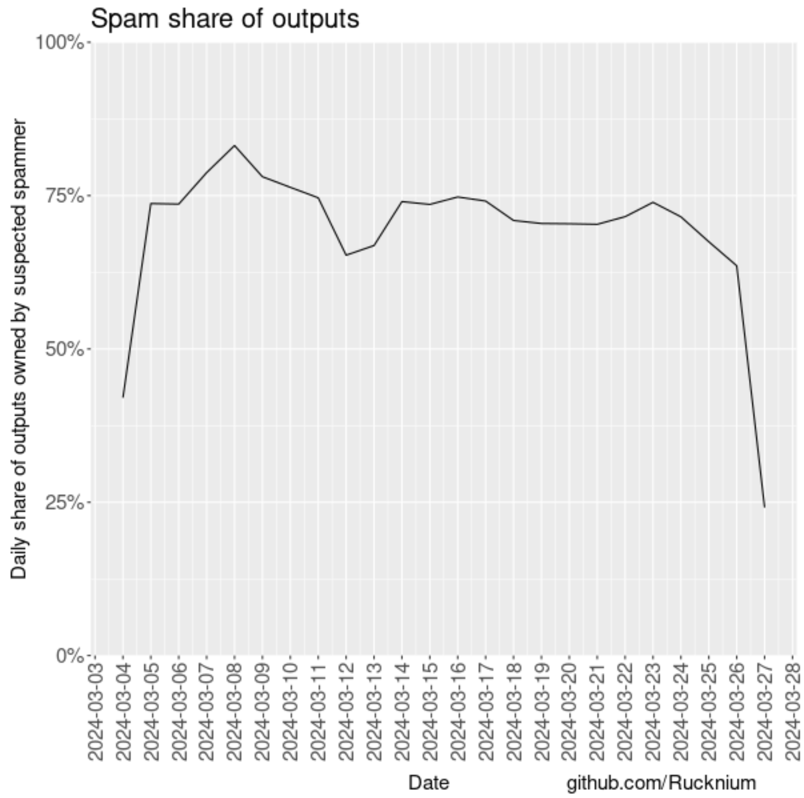


Figure 4 shows the daily share of outputs on the blockchain that are owned by the suspected spammer. The mean share of outputs since the suspected spam started is about 75 percent.

Figure 4: Spam share of outputs



3.2 Long term projection scenarios at different ring sizes

Fix the number of outputs owned by real users at r. The analysis will let the number s of outputs owned by the adversary vary. The share of outputs owned by real users is

$$p_r = \frac{r}{r+s} \tag{2}$$

The 2 expression can be written $p_r = \frac{1}{r} \cdot \dfrac{r}{1 + \frac{1}{r}s}$, which is the formula for hyperbolic decay with the additional $\frac{1}{r}$ coefficient at the beginning of the expression [Aguado et al., 2010].

Let n be the nominal ring size (16 in Monero version 0.18). The number of decoys chosen by the decoy selection algorithm is $n-1$. The mean effective ring size for a real user's ring is one (the real spend) plus the ring's expected number of decoys owned by other real users.

$$\mathrm{E}\left[n_e\right] = 1 + (n-1) \cdot \frac{r}{r+s} \tag{3}$$

The empirical analysis of Section 3.1 considered the fact that the `wallet2` decoy selection algorithm draws a small number of decoys from the pre-spam era. Now we will assume that the spam incident has continued for a very long time and all but a negligible number of decoys are selected from the spam era. We will hold constant the non-spam transactions and vary the number of spam transactions and the ring size. Figures 5, 6, and 7 show the results of the simulations.

Figure 5: Long-term projected mean effective ring size

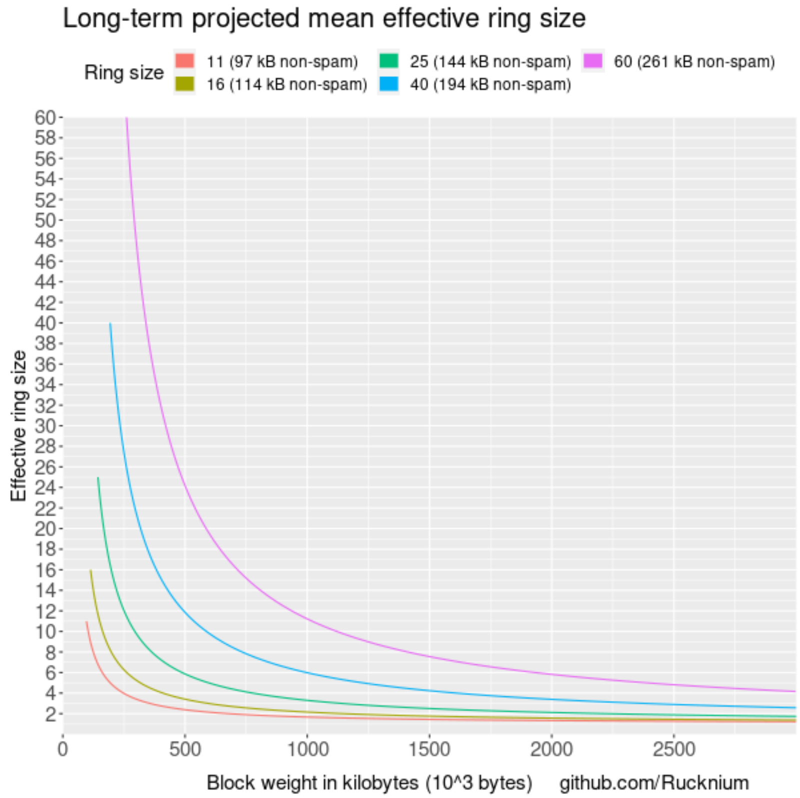


Figure 6: Long-term projected mean effective ring size (log-log scale)

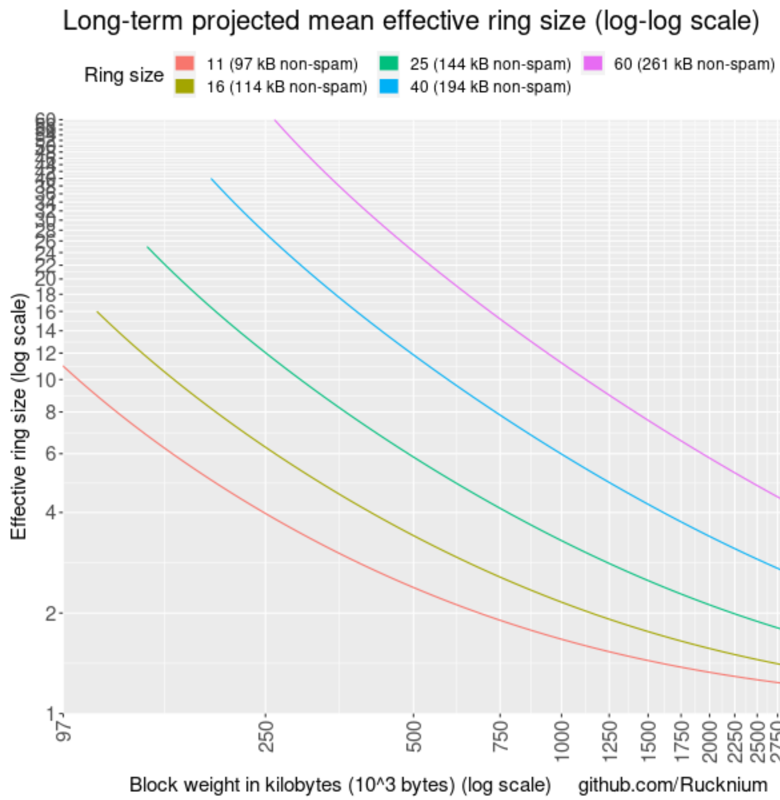
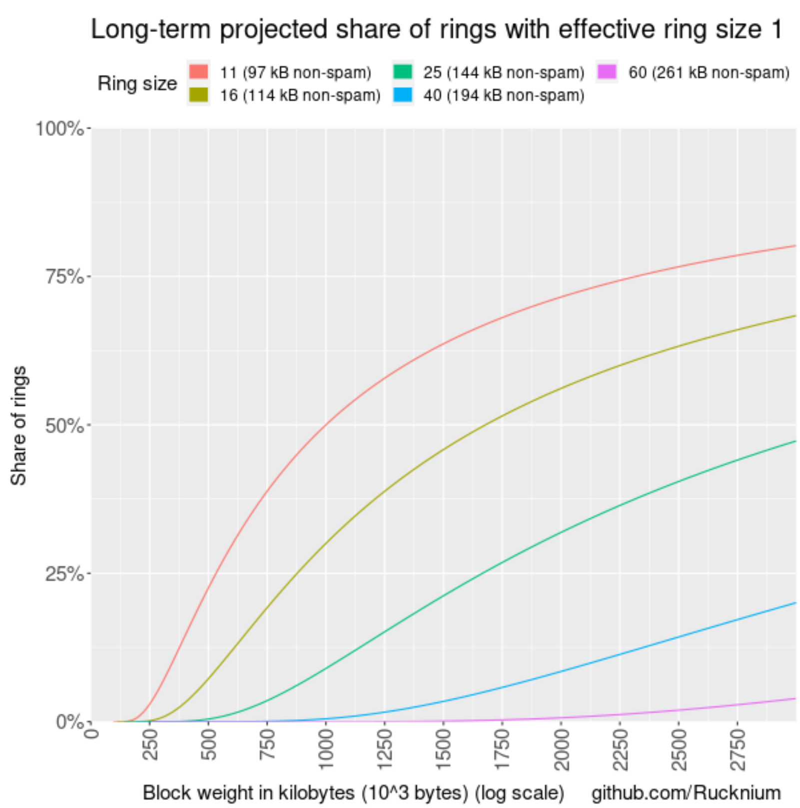


Figure 7: Long-term projected share of rings with effective ring size 1



3.3 Guessing the real spend using a black marble flooder's simple classifier

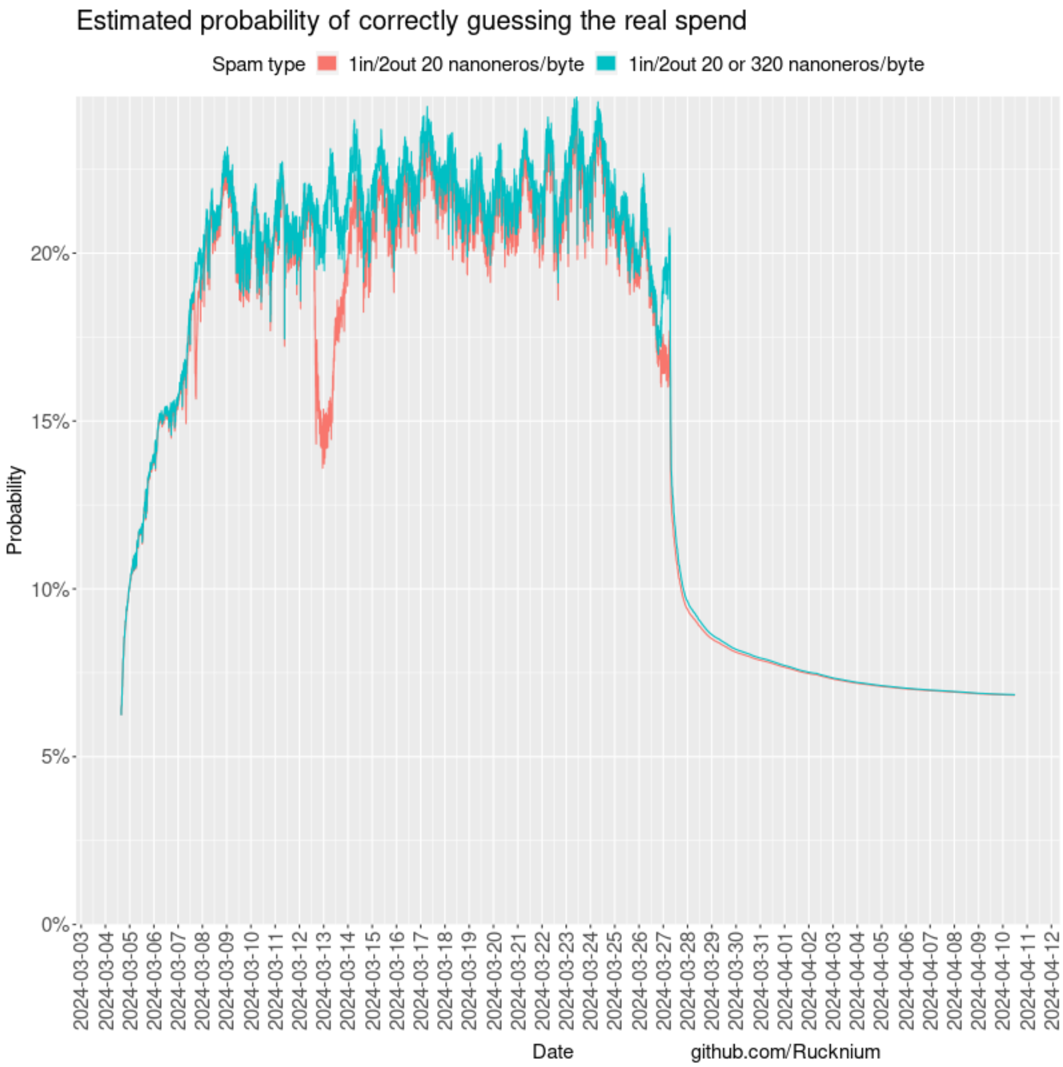
The adversary carrying out a black marble flooding attack could use a simple classifier to try to guess the real spend: Let n be nominal ring size and n_s be the number of outputs in a given ring that are owned by the attacker. n_s is a random variable because decoy selection is a random process. The adversary can eliminate n_s of the n ring members as possible real spends. The attacker guesses randomly with uniform probability that the ith ring member of the $n - n_s$ remaining ring members is the real spend. The probability of correctly guessing the real spend is $\frac{1}{n-n_s}$. If the adversary owns all ring members except for one ring member, which must be the real spend, the probability of correctly guessing the real spend is 100%. If the adversary owns all except two ring members, the probability of correctly guessing is 50%. And so forth.

The mean effective ring size is $\mathrm{E}[n_e]$ from 3. Does this mean that the mean probability of correctly guessing the real spend is $\frac{1}{\mathrm{E}[n_e]}$? No. The $h(x) = \frac{1}{x}$ function is strictly convex. By Jensen's inequality, $\mathrm{E}\left[\frac{1}{n_e}\right] > \frac{1}{\mathrm{E}[n_e]}$. The mean probability of correctly guessing the real spend is

$$\mathrm{E}\left[\frac{1}{n_e}\right] = \sum_{i=1}^{n} \frac{1}{i} \cdot f(i-1, n-1, \frac{\mathrm{E}[n_e]-1}{n-1}) \tag{4}$$

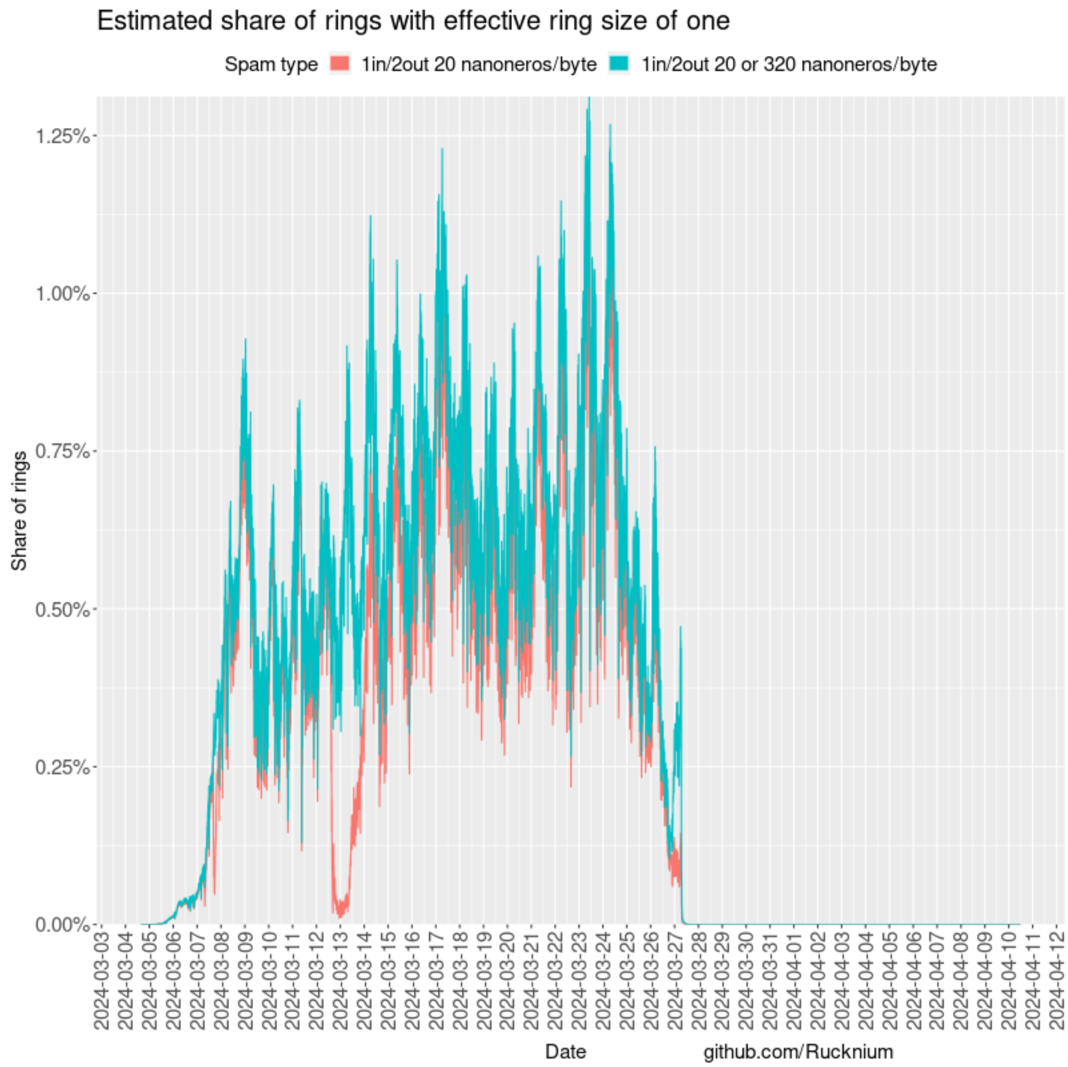
$\frac{1}{i}$ is the probability of correctly guessing the real spend when the effective ring size is i. f is the probability mass function of the binomial distribution. It calculates the probability of the decoy selection algorithm selecting $i-1$ decoys that are owned by real users. The total number of decoys to select is $n-1$ (that is the argument in the second position of f). The probability of selecting a decoy owned by a real user is $\frac{\mathrm{E}[n_e]-1}{n-1} = \frac{r}{r+s}$.

Figure 8: Estimated probability of correctly guessing the real spend



145 The probability of a given ring having all adversary-owned ring members except for the real spend is
146 $f\left(0, n-1, \frac{\mathrm{E}[n_e]-1}{n-1}\right)$. Figure 9 plots the estimated share of rings with effective ring size one.

Figure 9: Estimated share of rings with effective ring size of one



4 Chain reaction graph attacks

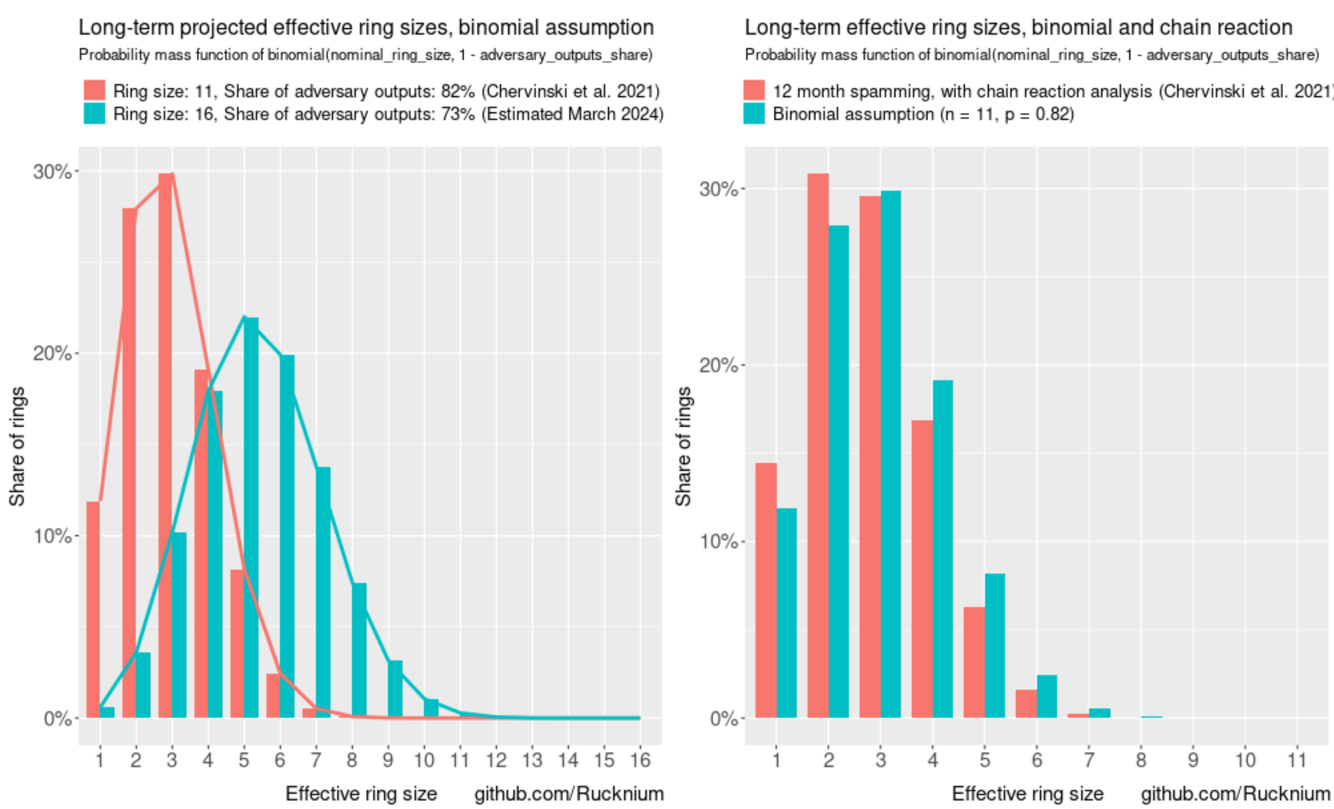
The effective ring size can be reduced further by applying a process of elimination to related rings. This technique is called a "chain reaction" or a "graph analysis attack". Say that the effective ring size in transaction A is reduced to two because of a black marble attack. One of the remaining two ring members is an output in transaction B. If the output in transaction B is known to be spent in transaction C because the effective ring size of transaction C was one, then that output can be ruled out as a plausible real spend in transaction A. Therefore, the adversary can reduce the effective ring size of transaction A to one.

Theorem 1 of [Yu et al., 2019] says that a "closed set" attack is as effective as exhaustively checking all subsets of outputs. The brute force attack is infeasible since its complexity is $O(2^m)$, where m is the total number of RingCT outputs on the blockchain. [Yu et al., 2019] implements a heuristic algorithm to execute the closed set attack that is almost as effective as the brute force method. [Vijayakumaran, 2023] proves that the Dulmage-Mendelsohn (DM) decomposition gives the same results as the brute force closed set attack, but the algorithm renders a result in polynomial time. The open source implementation of the DM decomposition in [Vijayakumaran, 2023] processes 37 million RingCT rings in about four hours.

In practice, how much further can chain reaction attacks reduce the effective ring size when combined with a black marble attack? [Egger et al., 2022] suggest some closed-form formulas to compute the vulnerability of different ring sizes to chain reaction attacks. However, [Egger et al., 2022] assume that decoys are selected by a partitioning process instead of Monero's actual mimicking decoy selection algorithm. It is not clear how relevant the findings of [Egger et al., 2022] are for Monero's mainnet. Monte Carlo simulations would be a better way to evaluate the risk of chain reactions.

[Chervinski et al., 2021] carries out a simulation using the old ring size of 11. In the 2input/2output spam scenario, 82% of outputs are black marbles. Assuming only the binomial distribution, i.e. no chain reaction analysis, Figure 10 compares the theoretical long-term distribution of effective ring sizes in the [Chervinski et al., 2021] scenario and the March 2024 suspected spam on Monero's mainnet. The share of rings with effective ring size 1 in the [Chervinski et al., 2021] scenario is 11.9 percent, but the share is only 0.8 percent with the suspected March 2024 spam. The mean effective ring sizes of the [Chervinski et al., 2021] scenario without chain reaction and the March 2024 spam estimate are 2.9 and 5.2, respectively.

Figure 10: Probability mass function of long-term effective ring sizes



[Chervinski et al., 2021] executes chain reaction analysis to increase the effectiveness of the attack. The second plot in Figure 10 compares the long term effective ring size achieved by [Chervinski et al., 2021] when leveraging chain reaction analysis and the effective ring size when only the binomial distribution is assumed. [Chervinski et al., 2021] increases the share of ring with effective ring size one from 11.9 to 14.5 percent. Mean effective ring size decreases from 2.94 to 2.76. This is a modest gain of attack effectiveness, but [Chervinski et al., 2021] appears to be using a suboptimal chain reaction algorithm instead of the closed set attack.

I implemented a DM decomposition simulation, using the real data from the black marble era of transactions as the starting point. The set of transactions produced by the adversary is known only to the adversary, so a reasonable guess was required. First, transactions that fit the spamming criteria were randomly assigned to black marble status in a proportion equal to the spam volume. Second, each ring was randomly assigned a real spend so that rings in non-black marble transactions would not entirely disappear in the next step. Third, outputs in black marble transactions were removed from the rings of non-black-marble transactions, except when the "real spend" assigned in the previous step would be removed. Fourth, all black marble transactions were removed from the dataset. The transaction graph left after these deletions is not necessarily internally consistent (i.e. funds might not actually be able to flow between transactions), but the objective is to approximate a chain reaction attack. Fifth, I used a modified version of the DM decomposition developed by [Vijayakumaran, 2023] to simulate a chain reaction attack.[1]

After the black marble outputs were removed but before the DM decomposition was applied, 0.57 percent of rings in the simulated dataset had a single ring member left. The real spend could be deduced in these 0.57 percent of rings. This simulated estimate is consistent with the results in Figure 9 that uses the $f\left(0, n-1, \frac{\mathrm{E}[n_e]-1}{n-1}\right)$ formula instead of a simulation. After the DM decomposition was applied to the simulated dataset, the share of rings whose real spend could be deterministically deduced increased to 0.82 percent. Therefore, the DM decomposition would increase the black-marble adversary's ability to deterministically deduce the real spend by 44 percent. My simulation results can be compared to the results of [Chervinski et al., 2021] in a different parameter environment, which found a 22 percent increase from a chain reaction attack (the share of rings with effective ring size one increased from 11.9 to 14.5 percent).

5 Countermeasures

See `https://github.com/monero-project/research-lab/issues/119`

TODO

[1] `https://github.com/avras/cryptonote-analysis`
`https://www.respectedsir.com/cna`

6 Estimated cost to suspected spammer

When the 1in/2out 20 nanoneros/byte spam definition is used, the total fees paid by the spam transactions over the 23 days of spam was 61.5 XMR. The sum total of the transaction sizes of the spam transactions was 3.08 GB.

When the 1in/2out 20 or 320 nanoneros/byte spam definition is used, the total fees paid by the spam transactions over the 23 days of spam was 81.3 XMR. The sub total of the transaction sizes of the spam transactions was 3.12 GB.

7 Transaction confirmation delay

Monero's transaction propagation rules are different from BTC's rules for good reasons, but two of the rules can make transactions seem like they are "stuck" when the txpool (mempool) is congested. First, Monero does not have replace-by-fee (RBF). When a Monero node sees that a transaction attempts to spend an output that is already spent by another transaction in the txpool, the node does not send the transaction to other nodes because it is an attempt to double spend the output. (Monero nodes do not know the real spend in the ring, but double spends can be detected by comparing the key images of ring signatures in different transactions.) Monero users cannot increase the fee of a transaction that they already sent to a node because the transaction with the higher fee would be considered a double spend. BTC has RBF that allows a transaction to replace a transaction in the mempool that spends the same output if the replacement transaction pays a higher fee. One of RBF's downsides is that merchants cannot safely accept zero-confirmation transactions because a malicious customer can replace the transaction in the mempool with a higher-fee transaction that spends the output back to themselves. Without RBF, Monero users must wait for their low-fee transaction to confirm on the blockchain. They cannot choose to raise their "bid" for block space even if they were willing to pay more. They have to get it right the first time. Fee prediction is especially important for Monero users when the txpool is congested because of the lack of RBF, but very little Monero-specific fee prediction research has been done.

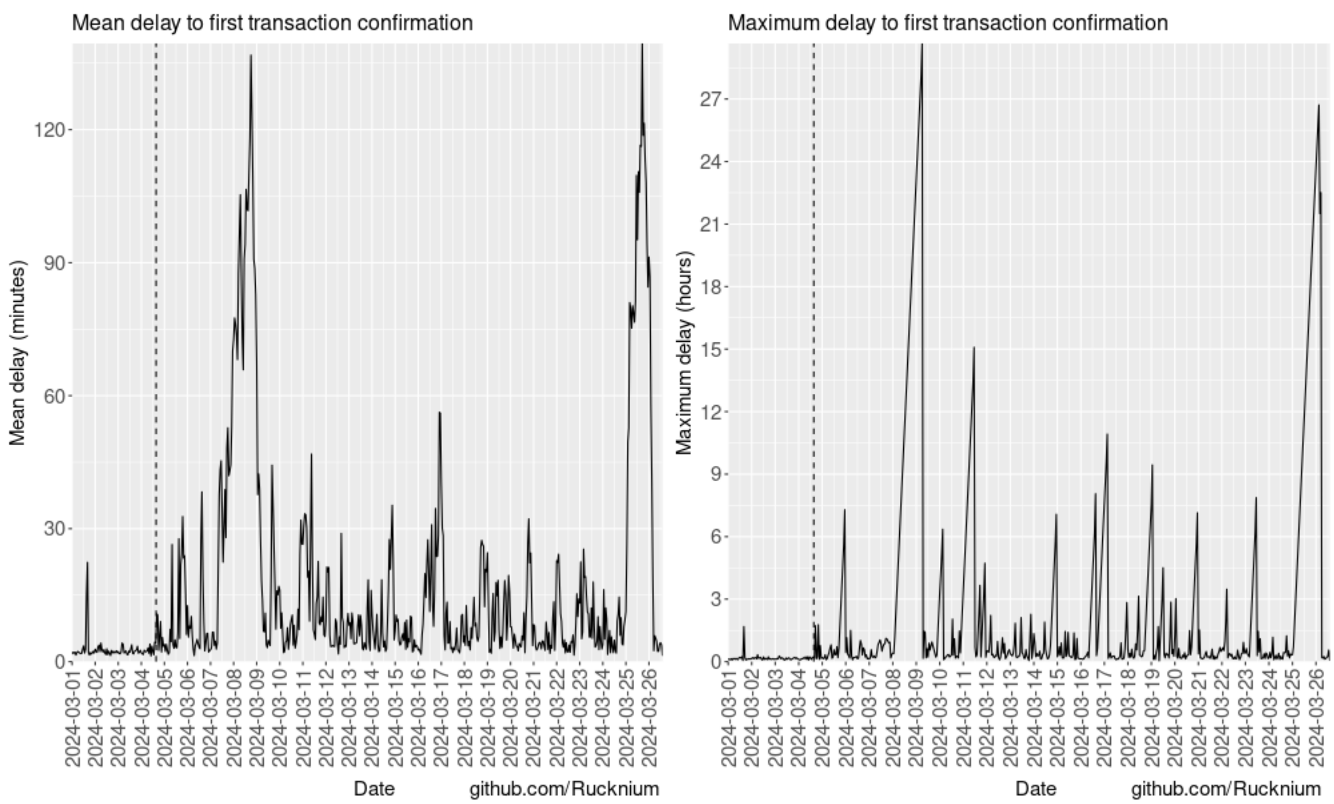
Unlike BTC, Monero also does not have child-pays-for-parent (CPFP), which allows users to chain multiple transactions together while they are still in the mempool. With CPFP, users can spend the output of the unconfirmed parent transaction and attach a higher fee to the child transaction. Miners have an incentive to include the parent transaction in the block because the child transaction is only valid if the parent transaction is also mined in a block. Monero transaction outputs cannot be spent in the same block that they are confirmed in. Actually, Monero users need to wait at least ten blocks to spend new transaction outputs because benign or malicious blockchain reorganizations can invalidate ring signatures.[2]

[2] "Eliminating the 10-block-lock" `https://github.com/monero-project/research-lab/issues/95`

Monero's transaction propagation rules can create long delays for users who pay the same minimum fee that the suspected spammer pays. When users pay the same fee as the spam, their transactions are put in a "queue" with other transactions at the same fee per byte level. Their transactions are confirmed in first-in/first-out order because the `get_block_template` RPC call to `monerod` arranges transactions that way.[3] Most miners use `get_block_template` to construct blocks, but P2Pool orders transactions randomly after they have been sorted by fee per byte.[4]

The first plot in Figure 11 shows the mean delay of transaction confirmation in each hour. The plot shows the mean time that elapsed between when the transaction entered the txpool and when it was confirmed in a block. Each hour's value in the line plot is computed from transactions that were confirmed in blocks in that hour. This data is based on txpool archive data actively collected from a few nodes.[5] The mean includes transactions with and without the spam fingerprint. Usually mean confirmation time was less than 30 minutes, but sometimes confirmations of the average transaction were delayed by over two hours.

Figure 11: Delay to first transaction confirmation



The second plot in Figure 11 shows the *maximum* waiting time for a transaction to be confirmed. The

[3] `https://github.com/monero-project/monero/blob/9bf06ea75de4a71e3ad634e66a5e09d0ce021b67/src/cryptonote_core/tx_pool.cpp#L1596`

[4] `https://github.com/SChernykh/p2pool/blob/dd17372ec0f64545311af40b976e6274f625ddd8/src/block_template.cpp#L194`

[5] `https://github.com/Rucknium/misc-research/tree/main/Monero-Mempool-Archive`

value of the line at each hour is the longest time that a transaction waited to be confirmed in one of the block mined in the hour or the amount of time that a transaction was still waiting to be confirmed at the end of the hour (whichever is greater). There were a handful of transactions that paid fees below the 20 nanoneros/byte tier that the spam was paying. These transactions did not move forward in the queue when the spam transactions were confirmed. Instead, they had to wait until the txpool completely emptied. Exactly 100 transactions waited longer than three hours. They paid between 19465 and 19998 piconeros per byte. Most of the transactions appeared to have set fees slightly lower than 20 nanonerpos per byte because they had an unusual number of inputs. 92 of them had four or more inputs. The remaining eight of them had just one input. Those eight may have been constructed by a nonstandard wallet.

8 Real user fee behavior

During the suspected spam, users must pay more than the minimum fee to put their transactions at the front of the confirmation queue. If users pay more than the minimum fee, usually their transactions would be confirmed in the next mined block. Monero's standard fee levels are 20, 80, 320, and 4000 nanoneros per byte. Users are not required to pay one of these fee levels, but all wallets that are based on `wallet2` do not allow users to choose custom fees outside of the four standard levels because of the privacy risk of unusual transactions.[6]

The "auto" fee level of the Monero GUI and CLI wallets is supposed to automatically change the fee of a transaction from the lowest tier (20 nanoneros/byte) to the second tier (80 nanoneros/byte) when the txpool is congested. Unfortunately, a bug prevented the automatic adjustment. On March 9, 2024 the Monero Core Team released the 0.18.3.2 version of Monero and the GUI/CLI wallet that fixed the bug.[7] Users are not required to upgrade to the latest wallet version, so probably many users still use the version that is not automatically adjusting fees.

The first plot of Figure 12 shows the share of trasnactions paying each of the four fee tiers. Any transactions that do not pay in the standard ranges $\{[18,22],[72,82],[315,325],[3000,4100]\}$ were not included in the plot. The 320 nanoneros/byte tier is interesting. About 10 percent of transactions paid 320 nanonero/byte until Februray 17, 2024. The date could have something to do with Monero being delisted from Binance on February 20, 2024.[8] Then on March 12-13, 2024 there was a burst of 320 nanonero/byte transactions. The 0.18.3.2 GUI/CLI wallet release could not explain the burst since the auto fee adjustment would only increase fees from 20 to 80 nanoneros/byte. The burst of 320 nanonero/byte transactions must have been either from a central service producing fees or from the suspected spammer.

The second plot of Figure 12 shows the same data with the suspected spam transactions eliminated

[6] `https://github.com/Rucknium/misc-research/tree/main/Monero-Nonstandard-Fees`

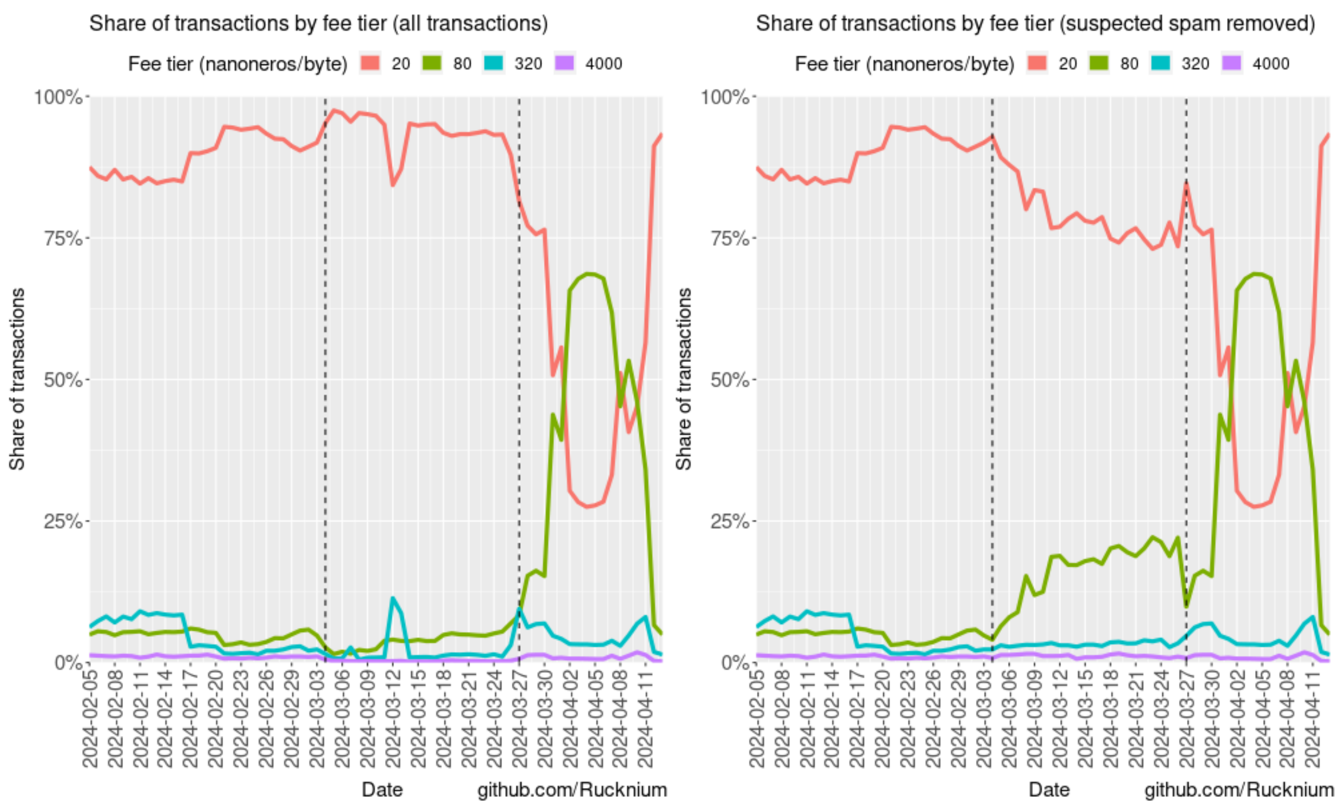
[7] "Monero 0.18.3.2 'Fluorine Fermi' released" `https://www.getmonero.org/2024/03/09/monero-0.18.3.2-released.html`

"wallet2: adjust fee during backlog, fix set priority" `https://github.com/monero-project/monero/pull/9220`

[8] `https://decrypt.co/218194/binance-finalizes-monero-delisting`

both the 80 and 320 nanoneros/byte transactions with the spam fingerprint were removed. There is a modest increase in 80 nanonero/byte transactions after the spam started.

Figure 12: Share of transactions by fee tier



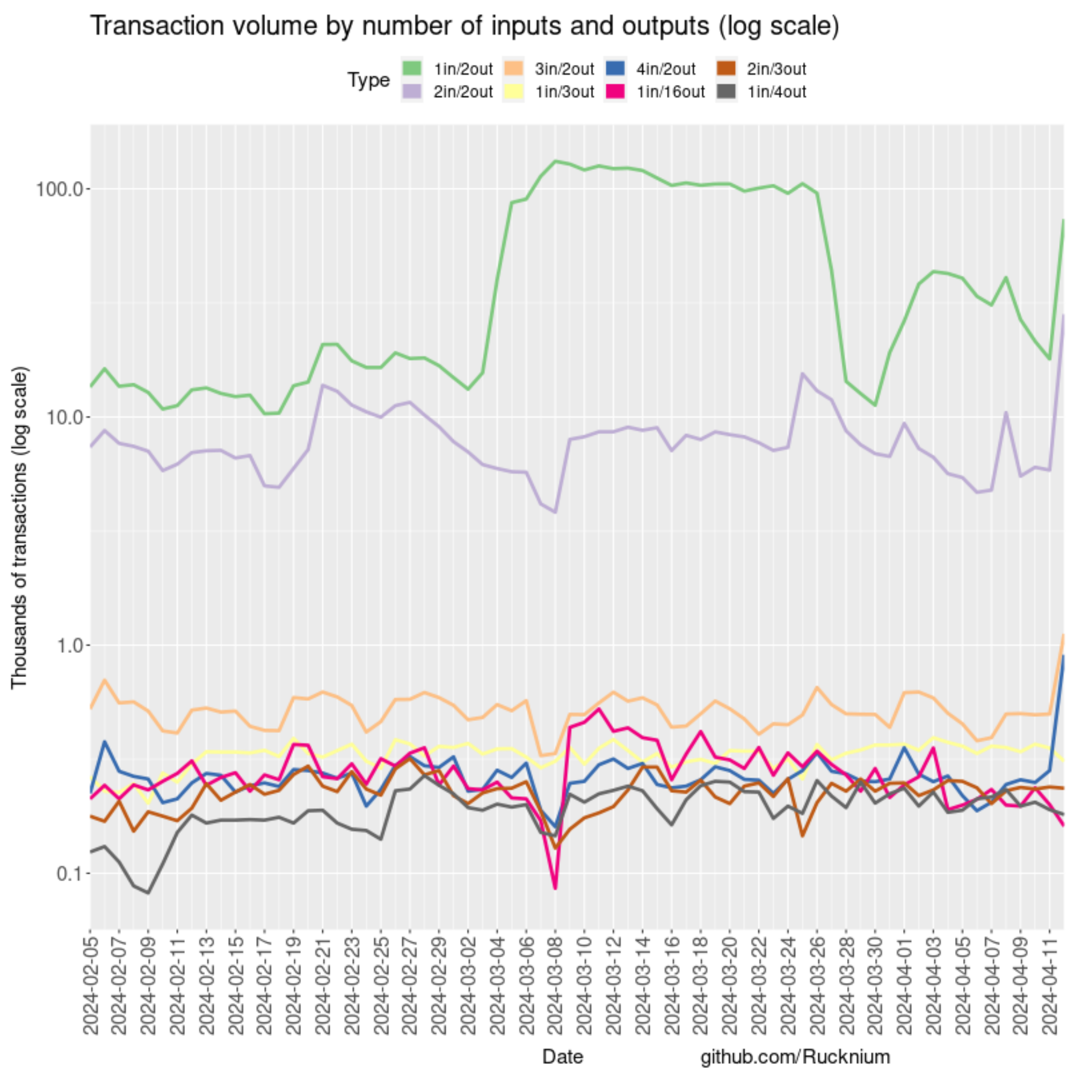
The mempool archive data suggest that merchants using zero-confirmation delivery were still safe during the spam incident. Once submitted to the network, transactions did not drop out of the mempool. They just took longer to confirm. There were only two transaction IDs in the mempool of one of the mempool archive nodes that did not confirm during the spam period. Both occurred on March 8 when the mempool was very congested. The two "disappearing transactions" could happen if someone submits a transactions to an overloaded public RPC node, the transactions does not propagate well, and then the user reconstructs the transactions with another node. The first transaction will not confirm because it is a double spend. Seeing a transaction in the mempool that never confirms happens sometimes during normal transaction volumes, too. Single transactions like that appeared on February 14, 17, and 23 and March 1 in the mempool archive data.

9 Evidence for and against the spam hypothesis

Is the March 4, 2024 transaction volume a result of many real users starting to use Monero more, or is it spam created by a single entity? [Krawiec-Thayer et al., 2021] analyzed the July/August 2021 sudden rise in transaction volume. We concluded that it was likely spam. Our evidence was: 1) There was a sharp increase of 1in/2out and 2in/1out transactions, but the volume of other transaction types did not increase, 2) All the suspected spam paid minimum fees, 3) The distribution of ring members became much younger, suggesting that the spammer was rapidly re-spending outputs as quickly as possible.

Available time has not permitted a full run of the [Krawiec-Thayer et al., 2021] analysis on the March 2024 suspected spam data. It is easy to do a quick check of transaction volume by input/output type. Figure 13 plots the eight most common in/out transaction types on a log scale. Only the volume of 1in/2out transactions increased on March 4, supporting the spam hypothesis.

Figure 13: Transaction volume by number of inputs and outputs (log scale)



More can be done to generate evidence for or against the spam hypothesis. [Krawiec-Thayer et al., 2021] analyzed the age of all ring members. Using the OSPEAD techniques, the distribution of the age of the real spends can be estimated.[9]

Dandelion++ can defeat attempts to discover the origin of most transactions because the signal of

[9] `https://github.com/Rucknium/OSPEAD`

the real transaction is covered by the Dandelion++ noise. When the signal is huge like the spam, some statistical analysis could overcome the Dandelion++ protection. Nodes can use the `net.p2p.msg:INFO` log setting to record incoming fluff-phase transactions. From April 14, 2024 to May 23, 2024, peer-to-peer log data was collected from about ten Monero nodes to try to establish evidence that the suspected black marble transactions originated from a single node.[10] Two factors have made this difficult. First, network topology information, i.e. which nodes are connected to each other, is not easily obtained. [Cao et al., 2020] used the last_seen timestamp in peer-to-peer communications to estimate the node topology, but the timestamp has been removed from Monero's node code.[11] Topology information would have allowed a "node crawler" to move through the network toward the likely source of the transaction spam. Second, log data collection started after the spam wave ended, and no new spam waves appeared. Therefore, the aim of the data analysis had to change. The following analysis uncovers facts about Monero's network and transaction propagation during normal operation that could provide a foundation for future research on the network's privacy and transaction propagation properties.

The number of unique IP addresses of peer nodes in the dataset is about 13,600. This may be a rough estimate of the total number of nodes on the network. Counting nodes this way can create both under-counts and over-counts because of nodes entering and leaving the network, nodes changing IP addresses, and multiple nodes behind the same IP address. In any case, the 13,600 figure is similar to a May 29, 2024 count by `monero.fail` of about 12,000 nodes on the network.[12]

The stability of the network topology is one of the factors that influences the effectiveness of Monero's Dandelion++ network privacy protocol. When nodes are connected to each other for a long time, it is easier for an adversary to get information about network topology and use it to try to discover the true node origin of a transaction ([Sharma et al., 2022]). The rate of connection creation and destruction could also affect the vulnerability of the network to partitioning and eclipse attacks ([Franzoni & Daza, 2022]).

A node can have two basic type of connections: incoming and outgoing. A node's "incoming" connections are connections that the node's peer initiated. A node's "outgoing" connections are connections that the node initiated. By default, nodes that are behind a firewall or residential router usually do not accept incoming connections. The default maximum number of outgoing connections is 12. There is no limit on incoming connections by default, but usually nodes accepting incoming connections have between 50 and 100 incoming connections.

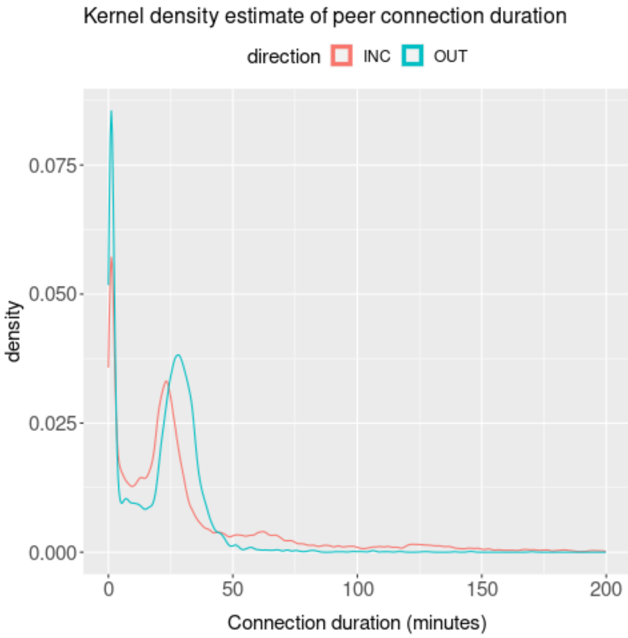
---

[10] Thanks to cyrix126, Nep Nep, and anonymous node operators for contributing log data.

[11] `https://github.com/monero-project/monero/pull/5681` and `https://github.com/monero-project/monero/pull/5682`

[12] https://web.archive.org/web/20240529014020/https://monero.fail/map

Figure 14: Peer connection duration



Based on the timestamps of transaction gossip messages from nodes that accept incoming connections, the median duration of incoming connections was 23 minutes. For outgoing connections, the median duration was 23.5 minutes. A small number of connections last for much longer. About 1.5 percent of incoming connections lasted longer than 6 hours. About 0.2 percent of incoming connections lasted longer than 24 hours. No outgoing connections lasted longer than six hours. This means that some peer nodes chose to keep connections alive for a long period of time. Node operators can manually set the `--add-priority-node` or `--add-exclusive-node` node startup option to maintain longer connections. Figure 14 is a kernel density estimate of the duration of incoming and outgoing connections. A small number of connections last for only a few minutes. A large number of connections end at about 25 minutes.

Monero's fluff-phase transaction propagation is a type of gossip protocol. In most gossip protocols, nodes send each unique message to each peer one time at most. Monero nodes will sent a transaction to the same peer multiple times if the transaction has not been confirmed by miners after a period of time. Arguably, this behavior makes transaction propagation more reliable, at the cost of higher bandwidth usage. Usually, transactions are confirmed immediately when the next block is mined, so transactions are not sent more than once. If the transaction pool is congested or if there is an unusually long delay until the next block is mined, transactions may be sent more than once. In the dataset, about 93 percent of transactions were received from the same peer only once. About 6 percent were received from the same peer twice. About 1 percent of transactions were received from the same peer more than twice.

Table 1 shows the median time interval between receiving duplicate transaction from the same peer. Up to the seventh relay, the ith relay has a delay of $f(i) = 5 \cdot 2^{i-2}$. After the seventh relay, the data suggests that some peers get stuck broadcasting transactions every two to four minutes.[13]

A Monero node's fluff-phase gossip message can contain more than one transaction. Usually, when a stem-phase transaction converts into a fluff-phase transaction, it will be the only transaction in its gossip message. As transactions propagates through the network, they will tend to clump together into gossip messages with other transactions. The clumping occurs because nodes maintain a single fluff-phase delay timer for each connection. As soon as the "first" transaction is received from a peer, a Poisson-distributed random timer is set for each connection to a peer. If a node receives a "second", "third", etc. transaction before a connection's timer expires, then those transaction are grouped with the first one in a single message that eventually is sent to the peer when the timer expires. Table 2 is shows the distribution of clumping. About 25 percent of gossip messages contained just one transaction. Another 25 percent contained two transactions. The remaining messages contained three or more transactions.

Table 1: Time between duplicate transaction receipts

Number of times received	Median minutes elapsed since previous time tx received	Number of txs (rounded to 10)
2nd	5.75	5,447,330
3rd	11.93	1,386,920
4th	21.92	596,310
5th	41.96	90,900
6th	85.30	22,180
7th	161.26	16,200
8th	2.03	9,710
9th	1.97	8,930
10th	1.99	8,930
11th	2.02	8,930
12th	4.03	1,090
13th	4.03	1,070
14th	4.03	1,070
15th	4.03	1,050
16th	4.00	1,050
17th	240.03	60

A subset of the nodes that collected the peer-to-peer log data also collected network latency data through ping requests to peer nodes. The data can be used to analyze how network latency affects transaction propagation. When it takes longer for a peer node to send a message to the data logging node, we expect that data logging node will receive transactions from high-latency nodes later, on average, than from low-latency nodes. I estimated an Ordinary Least Squares (OLS) regression model to evaluate this hypothesis. First, I computed the time interval between the first receipt of a transaction from any node and the time that each node sent the transaction: `time_since_first_receipt`. Then, the round-trip ping time was divide by two to get the one-way network latency: `one_way_ping`. The regression equation was `time_since_first_receipt = one_way_ping + error_term`

[13]boog900 stated that "re-broadcasts happen after 5 mins then 10, then 15 increasing the wait by 5 mins each time upto [sic] 4 hours where it is capped". The form of this additive delay is similar to the exponential delay that the empirical data suggests. https://libera.monerologs.net/monero-research-lab/20240828#c418612

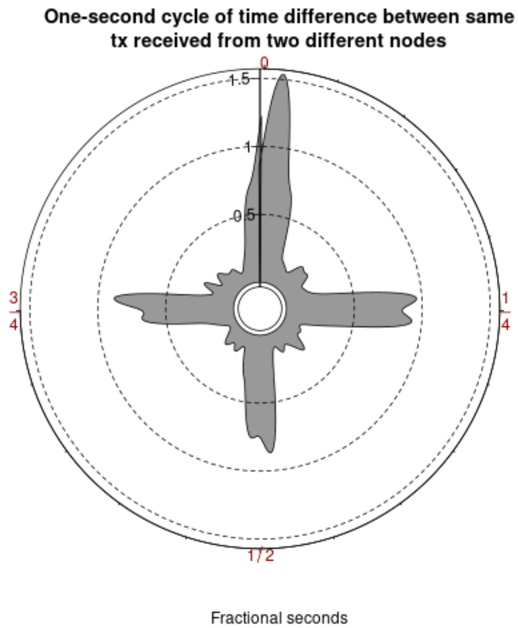
The estimated coefficient on `one_way_ping` was 7.5 (standard error: 0.02). This is the expected direction of the association, but the magnitude seems high. The coefficient means that a one millisecond increase in ping time was associated with a 7.5 millisecond increase in the time to receive the transaction from the peer. If the effect of ping on transaction receipt delay only operated through the connection between the peer node and the logging node, we may expect an estimated coefficient value of one. There are at least two possible explanations for the high estimated coefficient. First, assume that the logging nodes were located in a geographic area with low average ping to peers. And assume that the high-ping peers were located in an area with high average ping to peers. Then, the high-ping nodes would have high delay in sending *and* receiving transactions from the "low-ping" cluster of nodes. That effect could at least double the latency, but the effect could be even higher because of complex network interactions. Second, only about two-thirds of peer nodes responded to ping requests. The incomplete response rate could cause sample selection bias issues.

Table 2: Transactions clumping in gossip messages

Number of txs in message	Share of messages (percentage)
1	25.05
2	25.78
3	19.54
4	12.72
5	7.38
6	4.00
7	2.12
8	1.11
9	0.59
10	0.34
> 10	1.27

Occasionally, two of the logging nodes were connected to the same peer node at the same time. Data from these simultaneous connections can be analyzed to reveal the transaction broadcast delay patterns. The logging nodes did not try to synchronize their system clocks. The following analysis used the pair of logging nodes whose system clocks seemed to be in sync.

Figure 15: Time difference between tx receipt, one-second cycle



During the data logging period, Monero nodes drew a random variable from a Poisson distribution to create transaction broadcast timers for each of its connections. The distribution may change to exponential in the future.[14] The raw draw from the Poisson distribution set the rate parameter λ to 20 seconds. Then, the draw is divided by 4. The final distribution has a mean of 5 seconds, with possible values at each quarter second. If a node is following the protocol, we should observe two data patterns when we compute the difference between the arrival times of a transaction between two logging nodes. First, the differences should usually be in quarter second intervals. Second, the difference should follow a Skellam distribution, which is the distribution that describes the difference between two Poisson-distributed independent random variables. These patterns will not be exact because of difference in network latencies.

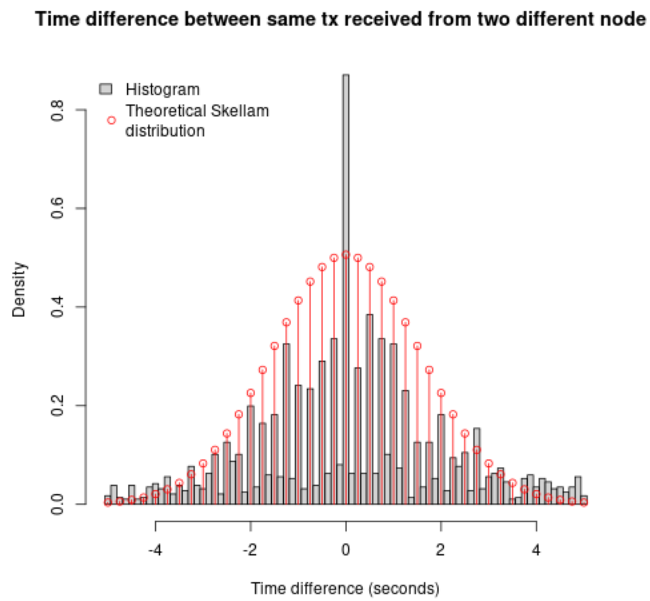
Figure 15 shows a circular kernel density plot of the time difference between two nodes receiving the same transaction from the same peer node. The data in the plot was created by taking the remainder (modulo) of these time differences when divided by one second. The results are consistent with expectations. The vast majority of time differences are at the 0, 1/4, 1/2, and 3/4 second mark.

Figure 16 shows a histogram of the empirical distribution of time differences and a theoretical Skellam distribution.[15] The histogram of the real data and the theoretical distribution are roughly similar except that the number of empirical observation at zero is almost double what is expected from the theoretical distribution. A zero value means that the two logging nodes received the transaction from the peer node at almost the same time.

[14] `https://github.com/monero-project/monero/pull/9295`

[15] The Skellam distribution probability mass function has been re-scaled upward by a factor of 8 to align with the histogram. Each second contains 8 histogram bins.

Figure 16: Histogram of time difference between tx receipt



References

[Aguado et al., 2010] Aguado, J., Cid, C., Saiz, E., & Cerrato, Y. (2010). Hyperbolic decay of the dst index during the recovery phase of intense geomagnetic storms. *Journal of Geophysical Research: Space Physics*, 115(A7). https://doi.org/https://doi.org/10.1029/2009JA014658

[Cao et al., 2020] Cao, T., Yu, J., Decouchant, J., Luo, X., & Verissimo, P. (2020). Exploring the monero peer-to-peer network. *Financial Cryptography and Data Security*, 578–594. https://link.springer.com/chapter/10.1007/978-3-030-51280-4_31

[Chervinski et al., 2021] Chervinski, O. J., Kreutz, D., & Yu, J. (2021). Analysis of transaction flooding attacks against monero. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1–8. https://doi.org/10.1109/ICBC51069.2021.9461084

[Egger et al., 2022] Egger, C., Lai, R. W. F., Ronge, V., Woo, I. K. Y., & Yin, H. H. F. (2022). On defeating graph analysis of anonymous transactions. *Proceedings on Privacy Enhancing Technologies*, 2022(3). https://petsymposium.org/2022/files/papers/issue3/popets-2022-0085.pdf

[Franzoni & Daza, 2022] Franzoni, F. & Daza, V. (2022). Sok: Network-level attacks on the bitcoin p2p network. *IEEE Access*, 10, 94924–94962. https://doi.org/10.1109/ACCESS.2022.3204387

[Krawiec-Thayer et al., 2021] Krawiec-Thayer, M. P., Neptune, Rucknium, Jberman, & Carrington (2021). *Fingerprinting a flood: forensic statistical analysis of the mid-2021 monero transaction volume anomaly.* https://mitchellpkt.medium.com/

fingerprinting-a-flood-forensic-statistical-analysis-of-the-mid-2021-monero-transaction-volume-a Available at https://mitchellpkt.medium.com/fingerprinting-a-flood-forensic-statistical-analysis-of-the-mid-2021-monero-transaction-volume-a19cbf41ce60

[Noether et al., 2014] Noether, S., Noether, S., & Mackenzie, A. (2014). *A note on chain reactions in traceability in cryptonote 2.0.* Research Bulletin. https://www.getmonero.org/resources/research-lab/pubs/MRL-0001.pdf

[Ronge et al., 2021] Ronge, V., Egger, C., Lai, R. W. F., Schröder, D., & Yin, H. H. F. (2021). Foundations of ring sampling. *Proceedings on Privacy Enhancing Technologies*, 2021(3), 265–288. https://doi.org/doi:10.2478/popets-2021-0047

[Rucknium, 2023] Rucknium (2023). *Closed-form expression of monero's wallet2 decoy selection algorithm.* https://github.com/Rucknium/misc-research/tree/main//Monero-Decoy-Selection-Closed-Form/pdf. Available at https://github.com/Rucknium/misc-research/tree/main//Monero-Decoy-Selection-Closed-Form/pdf

[Sharma et al., 2022] Sharma, P. K., Gosain, D., & Diaz, C. (2022). *On the anonymity of peer-to-peer network anonymity schemes used by cryptocurrencies.* https://doi.org/10.48550/ARXIV.2201.11860

[Vijayakumaran, 2023] Vijayakumaran, S. (2023). Analysis of cryptonote transaction graphs using the dulmage-mendelsohn decomposition. *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs).* https://aftconf.github.io/aft23/program.html

[Yu et al., 2019] Yu, Z., Au, M. H., Yu, J., Yang, R., Xu, Q., & Lau, W. F. (2019). New empirical traceability analysis of cryptonote-style blockchains. *Financial Cryptography and Data Security*, 133–149. https://link.springer.com/chapter/10.1007/978-3-030-32101-7_9