Discussion Note: Formula for Accuracy of Guessing Monero Real Spends Using Fungibility Defects

Draft v0.1

Rucknium

September 20, 2023

Abstract

Many parts of Monero's transaction format such as `tx_extra` contents, the fee paid to miners, and the decoy selection algorithm are not standardized by rules set by nodes nor blockchain consensus. Instead, alternative Monero wallet implementations are free to set these transaction characteristics in ways that are unique to the wallet implementation. Therefore, observers of the blockchain data can determine that a transaction was likely created by a nonstandard implementation. The distinguishing characteristics of transactions create many "anonymity puddles" instead of one "anonymity pool". An adversary that aims to guess the real spend of a ring signature can exploit the information contained in these characteristics, referred to as "fungibility defects".

This note defines a simple classification rule that leverages information about the fungibility defects of each ring signature's 16 members. The classification rule is applied to the rings in all transactions that have the defect. A ring member having the defect increases the probability that it is the real spend because a user will often spend "change" outputs from transactions that were created by their own nonstandard wallet. Using basic probability concepts I develop a closed-form expression for the probability that the classifier correctly classifies a ring member as the real spend. This probability, the Positive Predictive Value (PPV) is a function of ring size, the probability that a user spends change in a ring, and the proportion of transaction outputs on the blockchain that have the defect. These three values are either defined by Monero's protocol rules or can be accurately estimated directly from the blockchain data. For example, when these values are 16, 40%, and 5%, respectively, the probability that the classifier correctly classifies a ring member as the real spend is 31.7%, much higher than the $1/16 = 6.25\%$ probability of randomly guessing between the 16 ring members.

1 Introduction

Define a fungibility defect as a characteristic of a Monero transaction that marks it as being created by a "nonstandard" wallet implementation. A wallet implementation is "standard" if it creates transactions in the same way that `wallet2` does.[1] Some examples of defects are nonstandard fees, nonstandard data in `tx_extra`, and nonstandard decoy selection algorithms. These transactions produce outputs. We say that an output has the defect if the transaction that produced it has the defect. Transactions with the

[1] `https://github.com/monero-project/monero/blob/master/src/wallet/wallet2.cpp`

defect also contain one or more inputs, each of which is composed of a ring with 16 ring members that are outputs of previous transactions on the blockchain. One of the 16 ring members is the real spend. One or more of the ring members may be from a transaction that had the defect.

It is not always possible to know with certainty whether a transaction has a fungibility defect. For example, it can be difficult to determine that a set of decoys was selected with a nonstandard method because the age distribution of the members of any given ring could be generated by many different, but similar, decoy selection algorithms. Therefore, certain types of defects would require an initial step to probabilistically classify transactions as having the defect or not. In the rest of document we will assume that an observer knows with certainty whether a transaction has the defect.

Fungibility defects can help an observer guess the real spend in rings. When a user sends XMR to another user and does not fully consume the value of the input(s) used in the transaction, a "change" output is sent back to the user's wallet. Later, the user can use the change output as an input in another transaction. If the user is using a wallet that produces transactions with fungibility defects, the initial transaction will have the defect. The transaction that spends the change output will have the defect, too. If the defect is "rare" on the blockchain, then ring members with the defect will appear in the ring *as decoys* with low probability. However, the change output with the defect, *which is the real spend*, would appear with high probability. A classification rule can exploit the differences in these two probabilities to achieve high accuracy of correctly guessing the real spend.

2 Why do we care?

It may be easy to say "Eliminate all fungibility defects. Why do we need to know the exact impact on privacy? Fungibility defects are always bad." This sentiment ignore two facts:

1. Trying to eliminate certain types of fungibility defects could have downsides or be infeasible.

2. A privacy-improving change to the standard transaction construction procedure could be implemented at a point in time when users are not required to upgrade their wallet software, creating sets of "old format" standard transactions and "new format" standard transactions simultaneously on the blockchain. Users are usually only required to upgrade their wallet software when a hard fork (network upgrade) occurs.

Requiring the contents of `tx_extra` to be encrypted has been suggested as a way to achieve uniformity of `tx_extra` contents.[2] However, there is no certain way to automatically determine that `tx_extra` contents are actually encrypted. A statistical test could be performed, but some valid transactions would be rejected and some non-encrypted `tx_extra` contents would be let through the statistical test filter. A statistical test is an imperfect solution.

[2] `https://github.com/monero-project/monero/issues/6668`

Restricting allowed values for the transaction fee can also have downsides. There needs to be a signal to miners that users are willing to pay more to include into their transactions into a full block. Monero's dynamic block size adjustment requires the signal. Restricting the set of allowed fee values could mute the signal. Users may be willing to pay current_fee + ϵ, but may not be willing to pay the next-highest fee in the set of allowed fees. The current Seraphis upgrade proposal would require "fee discretization" to reduce the set of allowed fees.[3]

Requiring a specific decoy selection algorithm (DSA) has been suggested, too.[4] Depending on how a standard DSA is enforced, it could be difficult to change the DSA if a better one is found.

Even when a specific characteristic of transactions is not required by protocol rules, `wallet2`'s behavior establishes a "standard". What if a better standard is developed and deployed at a point in time that is not a hard fork? Does the privacy risk of two simultaneous "standard" formats outweigh the benefit to users? This has happened already. There were improvements to `wallet2's` decoy selection algorithms in September 2021 and April 2023 without a hard fork.[5] Two improvements to the DSA (OSPEAD and excluding coinbase outputs) are in development and theoretically could be implemented in `wallet2` before the next hard fork.[6] Computing specific numbers for privacy risks of having two different "standard" formats would allow conclusive cost-benefit analysis of changes to `wallet2`'s transaction construction procedure without a hard fork.

3 Existing research

Mitchell Krawiec-Thayer (isthmus) used exploratory data analysis to analyze the prevalence of fungibility defects at MoneroKon 2019.[7] Krawiec-Thayer followed up with "Heuristic Framework for Generalized Transaction Tree Analysis", which formalized fungibility defects and visualized how the defects could help analyze Monero's transaction graph.[8] Goodell (Surae) & Noether (Sarang) (2021?) is a draft Monero Research Lab research bulletin.[9] They develop a maximum likelihood estimator for probabilistically estimating Monero's real transaction graph. They set up a Monte Carlo simulation to estimate the probability of false positives and false negatives of their technique. However, the simulation apparently was never executed and the probability of false positives and false negatives were never computed. Rucknium's Monerotopia 2023 presentation lists fungibility defects and raises questions about how they could be

[3]https://gist.github.com/UkoeHB/f508a6ad973fbf85195403057e87449e

[4]https://github.com/monero-project/research-lab/issues/87

[5]https://www.getmonero.org/2021/09/20/post-mortem-of-decoy-selection-bugs.html
https://www.getmonero.org/2023/06/08/10block-old-decoy-selection-bug.html

[6]https://github.com/monero-project/research-lab/issues/93
https://github.com/monero-project/research-lab/issues/109

[7]Krawiec-Thayer, Mitchell (2019) "Visualizing Monero: A Figure is Worth a Thousand Logs."
https://youtube.com/watch?v=XIrqyxU3k5Q

[8]Krawiec-Thayer, Mitchell (2022) "Heuristic Framework for Generalized Transaction Tree Analysis."
https://github.com/Mitchellpkt/heuristics_framework_doc

[9]Goodell, Brandon & Noether, Sarang (2021?) "Disclosure Attacks and Privacy on Blockchains."

eliminated.[10]

This discussion note is different from the work by Krawiec-Thayer, Goodell, and Noether in two ways. Their work attempted to analyze how fungibility defects affecting the guessing probability for segments of the transaction graph. I do "single hop" analysis of one ring at a time. Goodell & Noether worked towards false positive/negative computation using a Monte Carlo simulation. I develop a closed form expression for false positive probability, which generally is considered more useful and more rigorous than a Monte Carlo simulation.

Change outputs have been analyzed for years in bitcoin chain analysis research. Moser & Narayanan (2022) is a recent example.[11]

4 Scope and assumptions

The analysis of this discussion note has these limits and assumptions:

1. Only rings of transactions that have the defect are analyzed. There are potential "black marble" externalities that the defects inflict on standard transactions, but they will not be analyzed in this note (yet).[12]

2. It is assumed that the wallets that produce the transactions with defects have decoy selection algorithms (DSAs) that select decoys independently from each other and independently from the real spend. `wallet2` selects decoys this way. Probably most nonstandard wallets select decoys this way, even if the probability distribution of their DSA is not the same as `wallet2`'s. However, transactions that use "cached" ring members that are not selected independently have been observed on the blockchain.

3. Decoy selection algorithms generally do not select from all outputs with equal probability. Instead, more recent outputs are more likely to be selected. This discussion note simplifies this factor by assuming a constant share of outputs have the defect.

4. Transactions with a specific defect may have a different share of the number of rings and outputs on the blockchain. For example, if transactions that have the defect usually have fewer rings, i.e. fewer inputs, than other transactions on the blockchain, then their share of rings on the blockchain may be lower than their share of outputs. This fact does not really affect the analysis except in Section 8 when comparing anonymity puddles to anonymity droplets.

[10]Rucknium (2023) "A Statistical Research Agenda for Monero"
`https://github.com/Rucknium/presentations/blob/main/Rucknium-Monerotopia-2023-Slides.pdf`

[11]Moser, Malte & Narayanan, Arvind (2022) "Resurrecting Address Clustering in Bitcoin"
`https://arxiv.org/abs/2107.05749`

[12]Noether, Surae, Noether, Sarang, & Mackenzie, Adam (2014). "A Note on Chain Reactions in Traceability in Cryptonote 2.0."
Rucknium (2023) "Empirical Privacy Impact of Mordinals (Monero NFTs)"
`https://www.reddit.com/r/Monero/comments/12kv5m0/empirical_privacy_impact_of_mordinals_monero_nfts/`

5. All analysis occurs at the ring level. In theory, information about multiple rings in a single transaction could be used to get higher classification accuracy, but that is not attempted in this note.

6. The analysis pretends that coinbase outputs do not exist.

7. It is assumed that the probability that a spent non-change output has the defect is equal to the share of outputs with the defect on the blockchain. This assumption seems reasonable, but is not critical for the overall analysis. Another assumption could be made. Then the formula would be slightly different.

5 A classifier

Let n be the ring size.

Let D be the number of outputs with defects that may appear in a ring. It is a random integer variable with support (i.e. can take the values) $0, 1, 2, ..., n$.

Consider a classification rule that is applied only to rings in transactions that have the defect. We must define its rules by the number of defects D in each ring that the rule is applied to.

The first case is when D is zero, i.e. when no ring members are from transactions with the defect. In this case the best the classifier can do is to randomly guess with equal probability among the n ring members. Therefore, its probability of successfully guessing the real spend is $\frac{1}{n}$.

The next case is when D is one. In this case the classifier guesses that the real spend is the single ring member with the defect. This guess will not always be correct. A ring that contains one output with the defect may be a user spending a change output from their wallet. Or the user may be spending a non-change address but the decoy selection algorithm randomly selected an output with the defect from the blockchain as a decoy. The next section will provide a formula for determining the probability of successfully guessing the real spend.

Let $D = 2$. There are two ring members with the defect. With no additional information to choose between the two ring members, the classifier chooses one of them at random to classify it as the real spend. When $D = 3, 4, ..., n$ follow the same classification procedure as with $D = 2$, except select randomly between the $3, 4, ..., n$ ring members, respectively, that have the defect.

6 Classifier accuracy

The proposed classifier will not correctly classify the real spend of 100 percent of rings in the transactions with the defect. It will make mistakes. How often will it make mistakes?

Below is a standard table explaining different classification outcomes.

		Classified as the real spend	
		Yes	No
Actually is the real spend	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

We will be concerned with the left column: true positives and false positives. The classifier always classifies exactly one ring member as the true spend. The Positive Predictive Value (PPV), also known as the precision, of a ring member classifier is the percentage of ring members classified as a real spend that are actually the real spend. It is the number of true positives divided by the sum of true positives and false positives: $PPV = TP/(TP + FP)$. The PPV is roughly equivalent to the "guessing probability" definition in Ronge et al. (2021).[13]

Our classifier encounters a ring that has a single ring member ($D = 1$) with the defect. The classifier classifies that ring member as the real spend. What is the probability that the classification is a true positive? A false positive?

In this case, the classifier would correctly classify a ring member as the real spend, i.e. produce a true positive, when the user actually spent a change output (which has the defect by assumption) and the decoy selection algorithm did not include any outputs with the defect as decoys. The classifier would also produce a true positive in the rare case that the user spends a non-change output, but that non-change output by coincidence was sent from a wallet that produces the defect. (If the classifier aimed to correctly classify spending of change outputs, it would by chance get it wrong in this case, but we defined the objective of the classifier as correctly classifying the real spend regardless of whether it is a change output.) The classifier would produce a false positive if the user did not spend a change output and the decoy selection algorithm selected an output with the defect as a decoy.

Let C be the random event that a user spends a change output in a ring that is in a transaction with the defect. $\Pr(C)$ is the probability that the event occurs. $1 - \Pr(C) = \Pr(\neg C)$ is the probability that the user spends an output in a ring that is not a change output.

$\Pr(D = d)$ is the probability that a ring has d members with the defect. The d is the realized value of the random D variable.

Decompose D into the number of decoys D_{decoy} in a ring that has the defect and the number of real spends D_{real} (zero or one) that have the defect. The decomposition is $D = D_{decoy} + D_{real}$.

Let β be the proportion of outputs on the blockchain that have the defect. Assume that a non-change output that a user spends has β probability of having the defect. As a probability expression: $\Pr(D_{real} = 1|\neg C) = \beta$. $\Pr(A|B)$ is the probability of event A conditional on B.

Let $\Pr(A \cap B)$ denote the probability of events A and B both occurring. As explained in words above, a ring will have exactly one ring member with the defect in three different cases:

[13] Ronge, V., Egger, C., Lai, R. W. F., Schroder, D., & Yin, H. H. F. (2021). "Foundations of Ring Sampling." Proceedings on Privacy Enhancing Technologies, 2021(3), 265–288.

Case 1: The user actually spent a change output and the decoy selection algorithm did not include any outputs with the defect. The probability of this occurring is $\Pr(D = 1 \cap C)$. The algebraic manipulations below show that $\Pr(D = 1 \cap C)$ is equivalent to

$$\Pr(D_{decoy} = 0) \cdot \Pr(C) \tag{1}$$

$\Pr(D = 1 \cap C)$	
$= \Pr(D_{decoy} = 0 \cap D_{real} = 1 \cap C) + \Pr(D_{decoy} = 1 \cap D_{real} = 0 \cap C)$	Use Law of Total Probability and $D = D_{decoy} + D_{real}$
$= \Pr(D_{decoy} = 0 \cap D_{real} = 1 \cap C) + 0$	All change outputs have the defect by assumption, so $\Pr(D_{real} = 0 \cap C) = 0$
$= \Pr(D_{decoy} = 0) \cdot \Pr(D_{real} = 1 \cap C)$	Selection of decoys is independent of the probability of spending change
$= \Pr(D_{decoy} = 0) \cdot \Pr(C)$	All change outputs have the defect by assumption, so $\Pr(D_{real} = 1 \cap C) = \Pr(C)$

Case 2: The decoy selection algorithm chose no outputs with the defect and the user spent a non-change output, but that non-change output coincidentally had the defect. The probability is $\Pr(D_{decoy} = 0 \cap D = 1 \cap \neg C)$. The algebraic manipulations below show that $\Pr(D_{decoy} = 0 \cap D = 1 \cap \neg C)$ is equivalent to

$$\Pr(D_{decoy} = 0) \cdot \Pr(\neg C) \cdot \beta \tag{2}$$

$\Pr(D_{decoy} = 0 \cap D = 1 \cap \neg C)$	
$= \Pr(D_{decoy} = 0 \cap D_{real} = 1 \cap \neg C)$	Use $D = D_{decoy} + D_{real}$
$= \Pr(D_{decoy} = 0) \cdot \Pr(D_{real} = 1 \cap \neg C)$	Selection of decoys is independent of the joint probability of the real spend having the defect and not spending change.
$= \Pr(D_{decoy} = 0) \cdot \Pr(\neg C) \cdot \beta$	$\Pr(D_{real} = 1 \mid \neg C) = \beta$ by assumption. By the definition of conditional probability, $\Pr(D_{real} = 1 \cap \neg C) = \Pr(\neg C) \cdot \beta$

Case 3: The user did not spend a change output and the decoy selection algorithm included exactly

one outputs with the defect. The probability of this occurring is $\Pr(D = 1 \cap D_{decoy} = 1 \cap \neg C)$. The algebraic manipulations below show that $\Pr(D = 1 \cap D_{decoy} = 1 \cap \neg C)$ is equivalent to

$$\Pr(D_{decoy} = 1) \cdot \Pr(\neg C) \cdot (1 - \beta) \tag{3}$$

$\Pr(D = 1 \cap D_{decoy} = 1 \cap \neg C)$	
$= \Pr(D_{real} = 0 \cap D_{decoy} = 1 \cap \neg C)$	Use $D = D_{decoy} + D_{real}$
$= \Pr(D_{decoy} = 1) \cdot \Pr(D_{real} = 0 \cap \neg C)$	Selection of decoys is independent of the joint probability of the real spend not having the defect and not spending change.
$= \Pr(D_{decoy} = 1) \cdot \Pr(\neg C) \cdot (1 - \beta)$	$\Pr(D_{real} = 0 \mid \neg C) = 1 - \beta$ by assumption. By the definition of conditional probability, $\Pr(D_{real} = 0 \cap \neg C) = \Pr(\neg C) \cdot (1 - \beta)$

We can determine the probability of a true positive and a false positive. The classifier will produce a true positive in Cases 1 and 2. Using the fact that $\Pr(\neg C) = 1 - \Pr(C)$ and collecting terms of expressions (1) and (2), the sum of the probabilities of Cases 1 and 2 is $\Pr(D_{decoy} = 0) \cdot (\Pr(C) + \beta \cdot (1 - \Pr(C)))$.

The classifier will produce a false positive in Case 3, which occurs with probability $\Pr(D_{decoy} = 1) \cdot \Pr(\neg C) \cdot (1 - \beta)$.

We can fill in the table for $D = 1$

		Classified as the real spend
		Yes
Actually is the real spend	Yes	$\Pr(D_{decoy} = 0) \cdot (\Pr(C) + \beta \cdot (1 - \Pr(C)))$
	No	$\Pr(D_{decoy} = 1) \cdot (1 - \Pr(C)) \cdot (1 - \beta)$

The Positive Predictive Value of the classifier when $D = 1$ is

$$PPV_{D=1} = \frac{\Pr(D_{decoy} = 0) \cdot (\Pr(C) + \beta \cdot (1 - \Pr(C)))}{\Pr(D_{decoy} = 0) \cdot (\Pr(C) + \beta \cdot (1 - \Pr(C))) + \Pr(D_{decoy} = 1) \cdot (1 - \Pr(C)) \cdot (1 - \beta)} \tag{4}$$

Consider the PPV for all cases: $D = 0, 1, 2..., n$. When $D = 0$, the classifier randomly guesses. Its true positive probability in the case is $\frac{1}{n}$. The $D = 0$ event only occurs when the decoy selection algorithm selects zero decoys with the defect, the real spend is not a change output, and the real spend does not have the defect, i.e. $D_{decoy} = 0 \cap \neg C \cap D_{real} = 0$. Therefore, $\Pr(D = 0) = \Pr(D_{decoy} = 0 \cap \neg C \cap D_{real} = 0)$. The algebraic manipulations below show that $\Pr(D_{decoy} = 0 \cap \neg C \cap D_{real} = 0) = \Pr(D_{decoy} = 0) \cdot \Pr(\neg C) \cdot (1 - \beta)$. Therefore, the contribution of this case to the overall true positive probability is $\frac{1}{n} \cdot \Pr(D_{decoy} = 0) \cdot \Pr(\neg C) \cdot (1 - \beta)$.

$\Pr(D_{decoy} = 0 \cap \neg C \cap D_{real} = 0)$	
$= \Pr(D_{decoy} = 0) \cdot \Pr(\neg C \cap D_{real} = 0)$	Selection of decoys is independent of the joint probability of the real spend not having the defect and not spending change.
$= \Pr(D_{decoy} = 0) \cdot \Pr(\neg C) \cdot (1 - \beta)$	$\Pr(D_{real} = 0 \mid \neg C) = 1 - \beta$ by assumption. By the definition of conditional probability, $\Pr(\neg C \cap D_{real} = 0) = \Pr(\neg C) \cdot (1 - \beta)$

When $D > 1$, the calculation is similar to the case of $D = 1$ except the classifier is randomly selecting between multiple candidates with equal probability. Therefore, the probability of correct classification when $D > 1 \cap C$ occurs is not 100 percent. It is $\frac{1}{D}$.

Let d be the realized value of the random D variable. Following the example above when $D = 1$, in the $D = d$ case the probability of a true positive is $\frac{1}{d} \cdot \Pr(D_{decoy} = d - 1) \cdot (\Pr(C) + \beta \cdot (1 - \Pr(C)))$. The denominator of the $PPV_{classifier}$ for every possible number of D is $\sum_{d=0}^{n} \Pr(D = d) = 1$ by the Law of Total Probability. We only need to consider the numerator to calculate $PPV_{classifer}$, which is

$$PPV_{classifier} = \frac{1}{n} \cdot \Pr(D_{decoy} = 0) \cdot (1-\beta) \cdot (1-\Pr(C)) + \sum_{d=1}^{n} \frac{1}{d} \cdot \Pr(D_{decoy} = d-1) \cdot (\Pr(C) + \beta \cdot (1 - \Pr(C))) \tag{5}$$

To compute $PPV_{classifier}$ we need to know the values of n, β, $\Pr(C)$, and $\Pr(D_{decoy} = d)$. The value of n is the ring size, a value defined by the Monero protocol. The β is the proportion of outputs on the blockchain that have the defect. It can be computed easily from the blockchain data. Estimation of $\Pr(C)$ and $\Pr(D_{decoy} = d)$ is not as simple. The next section explains how to do it.

7 Estimators for $\Pr(C)$ and $\Pr(D_{decoy} = d)$

As before, let β be the proportion of outputs on the blockchain that have the defect. Ignore the time dimension of decoy selection. When a decoy selection algorithm selects a single ring member, there is a β probability that the ring member has the defect. When the decoy selection algorithm selects a total of n decoys, the probability that the set of decoys contains some d number of outputs with the defect is determined by a binomial distribution. Let f be the probability mass function of this probability:

$$f(d, n, \beta) = \binom{n}{d} \beta^{d} (1 - \beta)^{n-d}, \text{ where } \binom{n}{d} = \frac{n!}{d!(n-d)!} \tag{6}$$

The probability that d decoys have the defect is $\Pr(D_{decoy} = d) = f(d, n - 1, \beta)$ since $n - 1$ decoys are selected. Replace $\Pr(D_{decoy} = d)$ in (5) with $f(d, n - 1, \beta)$ and use the fact $f(0, n - 1, \beta) = (1 - \beta)^{n-1}$ to obtain

$$PPV_{classifier} = \frac{1}{n} \cdot (1-\beta)^n \cdot (1-\Pr(C)) + \sum_{d=1}^{n} \frac{1}{d} \cdot f(d-1, n-1, \beta) \cdot (\Pr(C) + \beta \cdot (1-\Pr(C))) \tag{7}$$

We need to estimate $\Pr(C)$, the probability that the real spend of a ring in a transaction with the defect is itself an output with the defect. Consider the event $D = 0$. The previous section showed$\Pr(D = 0) = \Pr(D_{decoy} = 0) \cdot \Pr(\neg C) \cdot (1-\beta)$. We know $\Pr(D_{decoy} = 0) = (1-\beta)^{n-1}$ and $\Pr(\neg C) = 1 - \Pr(C)$. Therefore, $\Pr(D = 0) = (1-\beta)^{n-1} \cdot (1-\Pr(C)) \cdot (1-\beta) = (1-\beta)^n \cdot (1-\Pr(C))$. Solving for $\Pr(C)$ gives

$$\Pr(C) = 1 - \frac{\Pr(D = 0)}{(1-\beta)^n} \tag{8}$$

We now have everything we need to estimate $PPV_{classifier}$ from the data on the blockchain.

Let $\#\{D = 0 \cap R_{txD}\}$ be the number of rings on the blockchain that have zero ring members with the defect, yet the ring is within a transaction that has the defect.

Let $\#\{R_{txD}\}$ be the number of rings that are within a transaction that has the defect.

Define a plug-in estimator for $\Pr(D = 0)$:

$$\hat{\mu}_{D=0} = \frac{\#\{D = 0 \cap R_{txD}\}}{\#\{R_{txD}\}} \tag{9}$$

We need an estimator for β, the proportion of outputs on the blockchain that are within a transaction that has the defect.

Let $\#\{\mathcal{O}_{txD}\}$ be the number of outputs on the blockchain that are within a transaction that has the defect.

Let $\#\{\mathcal{O}\}$ be the total number of outputs on the blockchain.

Define a plug-in estimator for β:

$$\hat{\beta} = \frac{\#\{\mathcal{O}_{txD}\}}{\#\{\mathcal{O}\}} \tag{10}$$

We can define an estimator for $\Pr(C)$ using $\hat{\mu}_{D=0}$ and $\hat{\beta}$:

$$\hat{\mu}_C = 1 - \frac{\hat{\mu}_{D=0}}{(1-\hat{\beta})^n} \tag{11}$$

(Note that there is another suitable estimator for $(1-\hat{\beta})^n$. It is also the probability that rings in transactions *without* the defect have zero ring members with the defect. Therefore, a different estimator for $(1-\hat{\beta})^n$ could be $\#\{D=0 \cap R_{tx\neg D}\}/\#\{R_{tx\neg D}\}$.)

We have all items in place to create an estimator of $PPV_{classifier}$

$$\widehat{PPV}_{classifier} = \frac{1}{n} \cdot (1-\hat{\beta})^n \cdot (1-\hat{\mu}_C) + \sum_{d=1}^{n} \frac{1}{d} \cdot f(d-1, n-1, \hat{\beta}) \cdot \Big(\hat{\mu}_C + \hat{\beta} \cdot (1-\hat{\mu}_C)\Big) \tag{12}$$

8 Privacy impact

$\widehat{PPV}_{classifier}$ is a function of the ring size n, the estimated proportion of outputs on the blockchain that have the defect $\hat{\beta}$, and the estimated proportion of rings in transactions that have the defect whose real spend is a change output $\hat{\mu}_C$. Ring size n is set by the Monero protocol. How does varying $\hat{\beta}$ and $\hat{\mu}_C$ affect the value of $\widehat{PPV}_{classifier}$? And what are reasonable values for $\hat{\beta}$ and $\hat{\mu}_C$?

In my preliminary research on nonstandard fees on the blockchain, I have found at least 10 percent of transactions have a nonstandard fee. Each type of nonstandard fee is its own defect with its own "anonymity puddle". The nonstandard fee puddles are about 0.1%-5% in size, as a share of transactions on the blockchain. Therefore, the set of $\hat{\beta}$ to explore should at least cover 0.1%-5%. The set should extend even higher to analyze what could happen if there is a change in `wallet2`'s transaction construction behavior between hard forks. The share of transactions using the "new" `wallet2` method would rise from nothing to a majority when more users update their wallet software.

The value of $\hat{\mu}_C$ is determined by a combination of user behavior and wallet input selection rules. If a user receives a large amount of XMR from an external source in one payment and re-spends many times from that one payment into their wallet, then the share of their transactions that only use change outputs will be high, meaning the value of $\hat{\mu}_C$ would be high. When multiple outputs are available to be spent in a wallet, wallet software can be programmed to choose to spend change outputs with higher probability, which would also generate a higher value of $\hat{\mu}_C$. In my empirical investigation of nonstandard fees on the blockchain, I have seen estimated values of $\hat{\mu}_C$ between 40% and 50%. Exploring a range of 30%-70% is reasonable.

Table 1 shows the PPV for ranges of $\hat{\beta}$ and $\hat{\mu}_C$ when ring size is 16. Completely random guessing between 16 ring members with equal probability would produce a PPV of $1/16 = 6.25\%$. Both $\hat{\beta}$ and $\hat{\mu}_C$ have a large influence on PPV.

The numbers in the table support the theory that having smaller defect puddles ("droplets") is worse for privacy than having one larger defect puddle. Looking at the $\hat{\mu}_C = 50\%$ column, having a single defect puddle with 10% of blockchain outputs produces a PPV of 28.6% for those approximately 10% of rings on the blockchain that have the common defect. But having 10 distinct defects each with 1% share of blockchain outputs produces a 49.5% PPV for those approximately 10% of rings on the blockchain that have one of those 10 distinct defects.

Table 1: Positive Predictive Value when ring size is 16

	% rings of defective txs with change output as the real spend								
% blockchain outputs with defect	30	35	40	45	50	55	60	65	70
0.1	34.15	38.80	43.45	48.10	52.75	57.40	62.05	66.70	71.35
1	32.23	36.56	40.89	45.22	49.54	53.87	58.20	62.53	66.86
2	30.27	34.27	38.28	42.28	46.28	50.28	54.29	58.29	62.29
5	25.37	28.56	31.74	34.93	38.12	41.30	44.49	47.68	50.86
10	19.65	21.88	24.12	26.35	28.58	30.82	33.05	35.28	37.52
25	11.80	12.72	13.65	14.57	15.50	16.42	17.35	18.27	19.20
50	8.12	8.44	8.75	9.06	9.37	9.69	10.00	10.31	10.62

Table 2 is the same as Table 1 except ring size is 128. When $\hat{\beta}$ is very small and ring size is 128, the PPV is similar to the PPV with ring size 16. However, when $\hat{\beta}$ increases, PPV of ring size 128 decreases rapidly compared to the PPV of ring size 16.

Table 2: Positive Predictive Value when ring size is 128

	% rings of defective txs with change output as the real spend								
% blockchain outputs with defect	30	35	40	45	50	55	60	65	70
0.1	28.72	33.38	38.03	42.69	47.34	52.00	56.66	61.31	65.97
1	17.51	20.30	23.09	25.87	28.66	31.45	34.24	37.03	39.81
2	11.38	13.15	14.92	16.68	18.45	20.22	21.98	23.75	25.52
5	5.23	5.97	6.71	7.45	8.19	8.93	9.67	10.42	11.16
10	2.89	3.24	3.59	3.95	4.30	4.65	5.00	5.35	5.70
25	1.48	1.60	1.72	1.84	1.95	2.07	2.19	2.30	2.42
50	1.02	1.05	1.09	1.13	1.17	1.21	1.25	1.29	1.33

9 Simulation results

I wrote code to simulate how a wallet with a defect would create rings. Then I applied the classification rules to those rings and computed the PPV of the classification rule. The purpose of the Monte Carlo simulation is to provide some corroborating evidence that the estimator I developed for the PPV is "correct", i.e. is consistent. (I will not formally prove consistency in this draft.) The parameters of the simulation were:

Number of rings = 10 million

$n = 16$

$\beta = 0.05$

$\Pr(C) = 0.4$

The actual PPV in this simulation was 31.7407 when explicitly following the classification rule. The estimated PPV in the simulation, $\widehat{PPV}_{classifier}$, was 31.6962. Therefore, the actual PPV was about 0.1% different from the estimated PPV. The estimated $\Pr(C)$ was 39.92552%, very close to the true value of 40%.

The code to reproduce this simulation is available at

`https://github.com/Rucknium/misc-research/tree/main/Monero-Fungibility-Defect-Classifier/code`