FOR YOUR EYES ONLY! DO NOT SHARE!

Contains instructions for an attack on Monero users' privacy!

Research Roadmap for an Overhaul of Monero's Mixin Selection Algorithm

Rucknium

2021-09-16

Abstract

This document discusses the shortcomings of Monero's current mixin selection algorithm, a specific attack that exploits those shortcomings, and a plan to overhaul the algorithm so as to eliminate those shortcomings. I first challenge the widely-believed notion that the the true spend-age distribution is knowable by explaining a simple way to estimate it. I then develop the Rucknium Ratio Attack (RRA), a potent attack on user privacy that could allow tracing of about 35 percent of all transactions that have been confirmed on the Monero blockchain since the recommendation of Möser et al. (2018) was implemented. Finally, I outline a long-term solution to the problem in the form of nonparametric estimation of the true spend-age distribution. I close with a medium-term solution — Optimal Static Parametric Estimation of Arbitrary Distributions (OSPEAD) — that is probably implementable within a few months.

NOTE TO VULNERABILITY RESPONSE PROCESS REVIEWERS: The main thing I want to result from this submission is an official determination of what should be redacted from a public version of this document in order to protect user privacy. I give my own view about information exclusions in Recommendation III. I wish to submit a CCS funding proposal that is based on this "roadmap", so your views on redactions is appreciated. The organization and tone of this roadmap will likely change before release, but the basic elements are all here. The second thing is to initiate the Monero process for mitigating vulnerabilities, whatever that may be. The third and final thing is to determine if this submission qualifies for a vulnerability bug bounty. I have attached an XMR address for use if a bounty is deemed warranted. I initially offered jberman 50 percent of any bounty due to his substantial contribution to the advances contained herein, but he has declined any payment for this particular submission. I thank jberman and isthmus for their reviews of drafts of this document; the two of them plus Syksy would be good auxiliary reviewers for this submission. jberman and isthmus have been provided copies of this final version of the submission.

Developing an attack on user privacy is an inherently dangerous affair, but I did it for the following reasons. First, the attack formed in my mind as a natural result of thinking of how to fix the mixin selection algorithm. Second, the existence of this attack should remove all doubt about the importance of addressing weaknesses in the current mixin selection algorithm. Third, knowledge of a specific attack can help in developing a fix to the mixin selection algorithm, since any proposed solution's resistance to this specific attack can be measured precisely.

In order to prove first discovery of the Rucknium Ratio Attack if it were to be independently developed and published publicly by someone else, I plan to hash this document and, separately, certain individual sections and embed the hash in the OP_RETURN of transactions on the BTC and BCH blockchains within the next week. I believe that doing so carries no risk, but please let me know if there would be any concerns.

Just as I was finishing this submission, I was made aware of Ronge, Egger, Lai, Schröder, and Yin (2021), "Foundations of Ring Sampling," which lays out a formal framework for mixin selection algorithms. I have not had time to examine the paper closely. However, it is worth noting that they claim that estimating the real spend-age distribution is not feasible as a practical matter. I show below that this claim is false. Importantly, the fact that this claim is made does suggest that the broader computer science community remains somewhat clueless about the possibility of estimating the real spend-age distribution, and therefore its use as ammunition in a potential attack. This suggests that indefinite nondisclosure of the details of the Rucknium Ratio Attack and its foundations could be beneficial and advisable.

1 The mixin selection algorithm is in need of attention by qualified statisticians

Let $f_S(x)$ denote the probability density function (PDF) of the age of the real spend outputs of transactions on the Monero blockchain.[1]

[1] Strictly speaking, $f_S(x)$ and $f_M(x)$ are both probability mass functions (PMFs) since block age is discrete, but for expositional simplicity I will treat them as PDFs.

FOR YOUR EYES ONLY! DO NOT SHARE!
Contains instructions for an attack on Monero users' privacy!

1 THE MIXIN SELECTION ALGORITHM IS IN NEED OF ATTENTION BY QUALIFIED STATISTICIANS

Let $f_M(x)$ denote the PDF of the age of the mixins used by the reference wallet software for Monero.

My unvarnished view:

The gamma probability density function (PDF) with shape parameter 19.28 and rate 1.61 suggested by Möser et al. (2018) was a decent first draft for approximating $f_S(x)$ especially given that the paper covered a lot of other ground. However, in my view it never should have been used in production code. Even if it were to have been used in production code, the estimates should have been updated regularly, given the changing spend-age dynamics over time.

I am not one for conspiracy theories, but keep in mind that one of the authors of the paper (Andrew Miller) was on the board of the Zcash Foundation at the time of its publication and now. Other authors disclose that their research was financially supported by the U.S. National Science Foundation and the U.S. Department of Homeland Security. The authors would have little incentive to develop an unassailable mixin selection algorithm even if they were capable of doing so.

Speaking of capability of doing so: If the authors had a deep understanding of statistical methods, they did not show it in this paper. I have heard of the concept of "code smell" in programming, and here I analogously detected a "statistics smell". It was not a good smell. I would be happy to go into detail, but to summarize there are many lapses in statistical rigor, especially in the key section on countermeasures. In my experience, many computer scientists know "just enough to be dangerous" when it comes to statistics, especially when they try to work with data derived from human behavior. I am not really sure what statistics training for a typical computer scientist is, but from some observations it seems to me that it concentrates on probability theory rather than statistics itself.

Probability theory deals with problems when you know the underlying probability characteristics of the object under study; you just need to calculate the data that will be generated from the known stochastic processes. Statistics, on the other hand, examines the inverse: The data that is generated is known, but the underlying stochastic process that generated the data is not known. Statistical problems are generally more difficult than probability problems and often require substantial experience with empirical data to determine the best approach. Fortunately, as an empirical microeconomist my training in statistics is very extensive as is my experience with empirical data on economic behavior of individuals.

It is no one's fault that the flawed gamma-derived mixin selection algorithm was adopted into Monero's production code. It would have been entirely reasonable to assume that a paper with 11 authors, 3 of whom are professors at top universities, would have produced a reliable algorithm. But this highlights what appears to be a substantial shortcoming in how Monero's development has proceeded: Apparently no qualified statisticians have reviewed the protocol of a privacy model that is dependent upon its resistance to statistical attack in order to protect user privacy. Or to put it in a more precise, more alarming way: No statisticians *whose goal is to protect the privacy of Monero users* have reviewed the protocol. It is highly likely that a statistician or two is working for those who wish to de-anonymize Monero users. Of course, Monero developers and researchers cannot just conjure up statisticians to work on Monero, so statistical attack has hitherto been a blind spot in Monero's defenses. However, as the result of some unknown stochastic process I'm here now to help shore up the weaknesses in the current algorithm, and I may be able to recruit other statisticians to help.

It is probably the case that the the original developer of Monero, whoever they were, did not realize the full implications of developing a privacy protocol that requires statistical obfuscation to ensure users' privacy. But if

we choose to stay on this road, I see no other choice than to follow it to its necessary destination: minimum leakage of statistically meaningful data. This could be painful and get complicated. The alternatives, I see them, are A) Keep the flawed mixin algorithm and possibly compromise users' privacy; or B) switch to a completely different privacy model that does not rely on statistical obfuscation. For example, Zcash, to my knowledge, does not require statistical obfuscation except for basic user awareness about timing and amount attacks when transferring coins from the transparent to the shielded pool and vice versa.

I do not think that an apparently "good enough" mixin selection algorithm is actually good enough. The current algorithm does provide some minimum safety for the privacy of even the most-exposed transactions, but as Monero becomes an even more attractive target for government surveillance, the statistical attacks on privacy will become increasingly intense. The field of statistics is vast. Vast, vast, vast. There are currently over 18,000 statistical packages on the Comprehensive R Archive Network (CRAN). Over 200 of them deal specifically with mixture distributions — which would be a first step of anyone wanting to undermine the privacy of Monero. And this is not accounting for statistical techniques that may be developed in the near future. As far as I can see, the best way to defend against attacks is define $f_M(x)$ in a way that minimizes the leakage of statistically meaningful data.

Recommendation I: Across all software and protocol development contexts, interdisciplinary collaboration between computer scientists, statisticians, and other disciplines is essential to eliminate blind spots that could compromise user privacy. This is a systemic problem, of course, and the Monero Project cannot hope to solve it at a global level. However, the Monero Project can lead by example.

Recommendation II: The Monero Project should actively recruit technical talent from universities and university orbits in order to identify and eliminate vulnerabilities in its protocol. I outline a sketch of such a recruitment effort here.

2 The age distribution of real spends is recoverable

It seems from conversations with Monero community members that is has been believed that obtaining the distribution of real spends was not possible now that the mixin selection suggestion from Möser et al. (2018) has been implemented. Under some plausible assumptions, this belief is not at all true.

(Repeating some definitions) Let:

$f(x)$ be the observed spend-age PDF (includes mixins and real spends). This is known. Or, more precisely, the empirical version of this is known.

$f_S(x)$ be the PDF of the age of the real spend outputs of transactions on the Monero blockchain. This is unknown (for now).

$f_M(x)$ be the PDF of the age of the mixins used by the reference wallet software for Monero. This is probably known; see below.

α the proportion of ring members that are mixins. This is known to be $10/11$.

The only important assumption that is needed for the following argument is that $f_M(x)$ is known. I will return to this issue shortly. For now, assume that $f_M(x)$ is known.

These expressions are related in the following way:

$$f(x) = (1-\alpha)\cdot f_S(x) + \alpha \cdot f_M(x) \tag{1}$$

Everything in that expression is approximately known, except for $f_S(x)$. Therefore, $f_S(x)$ can be calculated by simple arithmetic, basically. Rearranging (1) yields

$$f_S(x) = \frac{1}{1-\alpha}\left(f(x) - \alpha \cdot f_M(x)\right) \tag{2}$$

Once in hand, $f_s(x)$ can likely serve as ammunition for any number of potent attack weapons. After a bit of thinking, I have developed one such weapon: the Rucknium Ratio Attack (RRA). The RRA is explained in the next section, but first let me return to the issue of knowing $f_M(x)$.

Most likely, $f_M(x)$ is known since it is written into the code of the reference wallet. If a substantial proportion of transactions on the blockchain are created by a wallet that does not use the reference wallet's mixin selection algorithm, then we only know $f_M(x)$ "with error". The only known wallet that does not use the reference wallet's mixin selection algorithm is monero-lws/mymonero.com . The mymonero wallet does not suffer from the integer truncation bug that has recently been discovered in the reference wallet. The proportion of transactions on the Monero blockchain that mymonero has produced is unknown. However, if it is substantial then it might be estimated via estimation of a mixture distribution.

Isthmus believes that there is a wallet out there that uses a uniform distribution as well as an "exchange wallet bug". Very recently there have been some discoveries in the course of analyzing the July–August 2021 transaction volume anomaly that could help identify rogue wallets and their share of total transaction volume. See also jberman's work here. In any case, a Maximum Likelihood Estimation (or similar estimation technique) of $f_S(x)$ and $f_M(x)$ could proceed as follows. Define a decomposition of $f_M(x)$ as $f_M(x) = \alpha_1 \cdot f_{M,1}(x) + \alpha_2 \cdot f_{M,2}(x) + \cdots + \alpha_N \cdot f_{M,N}(x)$ where each $f_{M,i}(x)$ is known but the $\alpha_i$weights are unknown. Define $\alpha = \sum_{i=1}^{N} \alpha_i$. Then the following mo del could be estimated:

$$f(x) = (1-\alpha)\cdot f_S(x) + \sum_{i=1}^{N} \alpha_i \cdot f_{M,i}(x)$$

$f_s(x)$would then be a nonparametric "residual".

3 The Rucknium Ratio Attack (RRA)

3.1 Definition

The Rucknium Ratio Attack (RRA) is intuitive and requires the application of no sophisticated statistical techniques, although as a formal matter it may amount to a maximum likelihood estimator of some sort. It is so simple that laypeople — say, for instance, members of a jury in a criminal case — may be able to grasp it intuitively.

As I note later, I have not dotted all i's nor crossed all t's on the attack since my time is better spent developing defenses to this attack and any related attacks, but the core of the attack likely has no fundamental flaws.

The sketch of the attack requires just two steps:

1. Estimate $f_S(x)$ and $f_M(x)$ through the procedure outlined in Section 2 or a similar procedure.

2. Choose any ring of "interest" to analyze. The most likely real spend is then the ring member that comes from the block for which the following ratio is highest:

$$\frac{f_S(x)}{f_M(x)} \tag{3}$$

I call (3) the Rucknium Ratio. The attack operationalizes the intuitive idea that the ring member that is most likely to be the real spend is the ring member that comes from the block where the real spend distribution is thickest relative to the mixin selection algorithm's distribution.

A formal definition of the Rucknium Ratio Attack follows.

Let $\boldsymbol{R} = \{r_1, r_2, ..., r_{11}\}$ be the set of ring members. Without loss of generality, let the real spend be r_{11}.

Let $\boldsymbol{B} = \{b_1, b_2, ..., b_{11}\}$ be the block heights of the blocks that each ring member r was originally confirmed in.

Let RRA, the Rucknium Ratio Attack, be a function that maps the set of ring members onto itself, i.e.

$RRA : \boldsymbol{R} \rightarrow \boldsymbol{R}$

As we will see in a moment, RRA can be a many-to-many function or a many-to-one function.

Finally, define RRA as follows:

$$RRA(\boldsymbol{R}) \equiv \underset{r_i \in \boldsymbol{R}, b_i \in \boldsymbol{B}}{\operatorname{argmax}} \left\{ \frac{f_S(b_i)}{f_M(b_i)} \right\} \tag{4}$$

where each element b_i of $\boldsymbol{B}$ corresponds to its counterpart element r_i in $\boldsymbol{R}$.

RRA will not necessarily produce a singleton since it is possible for multiple ring members to come from the same block. It is also possible that the Rucknium Ratio $\frac{f_S(x)}{f_M(x)}$ is identical for distinct elements of $\boldsymbol{B}$. In that case, application of the RRA function to a particular $\boldsymbol{R}$ would produce a set with multiple elements.

The probability that the RRA correctly and uniquely guesses the true spend for any given $\boldsymbol{R}$ is then:

$$Pr\left(\text{guess real spend correctly} | \boldsymbol{R}\right) = Pr\left(RRA(\boldsymbol{R}) = \{r_{11}\} | \boldsymbol{R}\right) \tag{5}$$

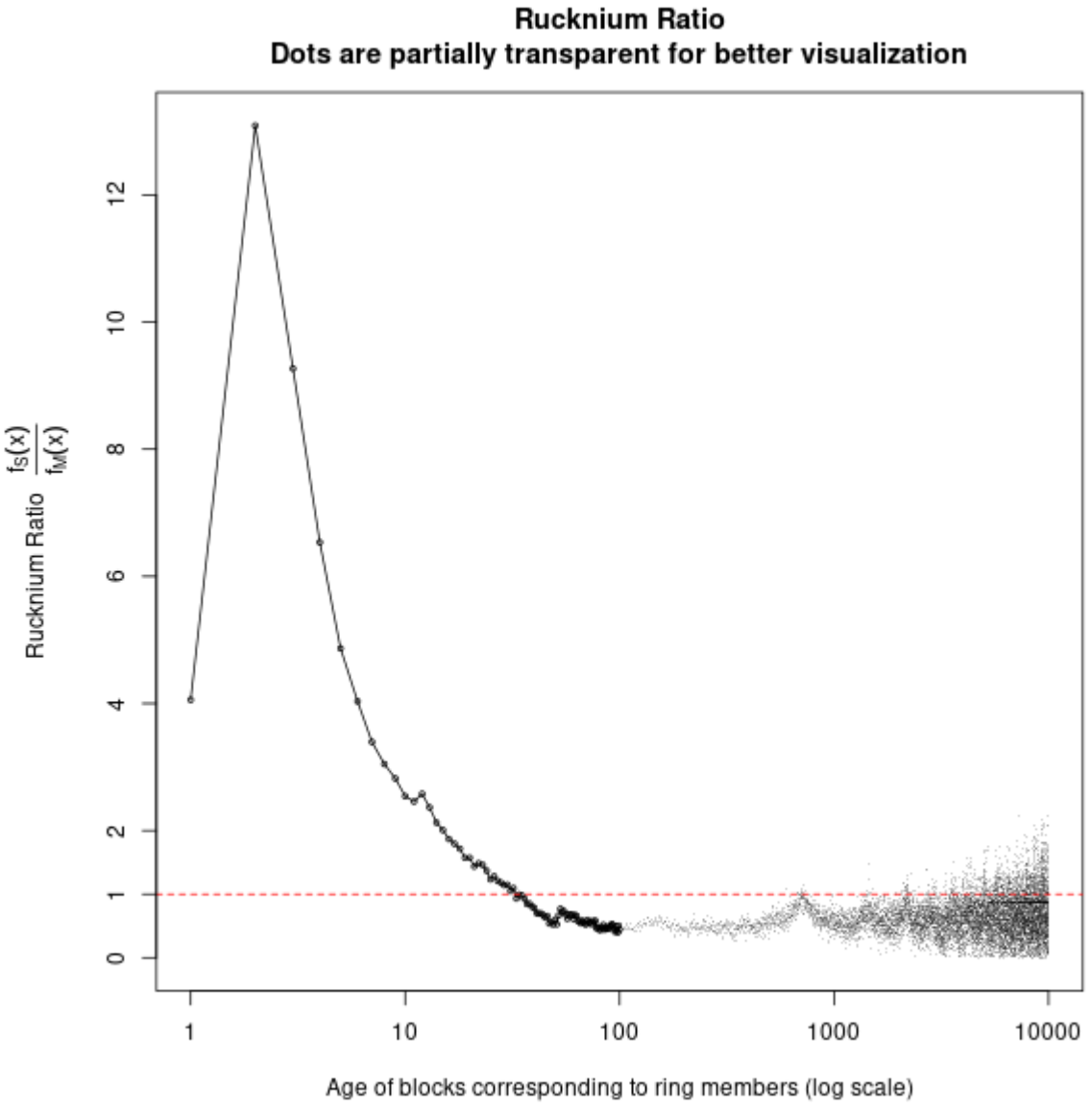
3.2 Simulations with real data demonstrate the potency of the RRA

As stated in Section 2, there are some very surmountable challenges in calculating $f_M(x)$. Assume for the moment that $f_M(x)$ is calculated. Specifically, let $f_M(x)$ be the distribution generated by the official wallet implementation. Then let f(x) be the observed distribution of ring member ages. jberman produced this exact data for his analysis of his pull request #7821. The `Observed` and `Current decoy selection algo` columns represent frequency data that can be converted into approximations of $f(x)$ and $f_M(x)$, respectively, by normalizations. Then $f_S(x)$ can be computed by application of equation (2). Note that due to the long thin tails of the distributions and the noisy nature of the empirical data and jberman's simulated data, it can be the case that the calculated value of $f_S(x)$ can be negative for some values of x, which violates an axiom of probability theory. In those cases, the negative value of $f_S(x)$ was replaced by $f_M(x)$, since that would be the most conservative approach.

Figure 1 plots the Rucknium Ratio for the most recent 10,000 blocks. The Rucknium Ratio is unacceptably high for the first 10 blocks or so. Also of note are the apparent cycles, possibly on a 24-hour basis, displayed in the plot.

A Monte Carlo simulation can proceed as follows. Construct 10 million rings. This can be done by drawing 110 million pseudorandom observations from the estimated $f_M(x)$ and 10 million pseudorandom observations from $f_S(x)$. Form up the rings appropriately and apply the RRA to each ring. The simulated probability of success of

Fig. 1:


Rucknium Ratio
Dots are partially transparent for better visualization
Rucknium Ratio $\frac{f_S(x)}{f_M(x)}$
0
1
2
4
6
8
10
12
1
10
100
1000
10000
Age of blocks corresponding to ring members (log scale)

the RRA as in (5) for an arbitrary ring is then obtained. R code for carrying out this simulation is in Appendix A: RRA simulation code and attached to this document as a separate file.

With this simple attack, the simulated probability of successfully guessing the real spend is about 35.4 percent. If the existing $f_M(x)$ was doing its job perfectly, this figure would be only $1/11 \approx 9.1$ percent. Note that the simulation is conducted with the assumption that the $f_M(s)$ of the empirical data is known. Note also that the data that this simulation is based on is from a blockchain "epoch" before jberman's fix in PR #7821 was applied.

3.3 Increasing the ring size mitigates the vulnerability only weakly

Not only is the current mixin selection algorithm vulnerable to the RRA, but the Monero Project's plans to increase ring size only slightly reducing the attack potency. Only focusing on increasing the ring size is analogous to producing a lot of camouflage but yet not putting it in the right places to cover the vulnerable entity. Overhauling the mixin selection algorithm should therefore be made a high priority. Via simulations, the effect of increasing the ring size on the potency of the RRA can be estimated.

We can simply repeat the same simulation described in Section 3.2, but for every ring size $2 - 256$ instead of just a single ring size of 11. To keep the computational burden reasonable, the number of constructed rings will be reduced to 1 million.

Figure 2 displays the simulated probability of success of the RRA as in (5) as a function of ring size. The graph roughly traces out a shape of exponential decay. Even with a ring size of 256, the simulated probability of a correct guess of the real spend when the RRA is applied is about 8.5 percent. This may be considered an "acceptable" rate of guessibility, but it is a far cry from $1/256 \approx 0.39$ percent, which is what it would be if $f_M(x)$ perfectly matched $f_S(x)$.

At first glance, the slow fall-off in the potency of the RRA as ring size increases may seem incredible, but consider the following back-of-the-envelop illustration. About 13.8 percent of the mass of $f_S(x)$ is in the most recent 10 blocks alone. Only about 2.7 percent of the mass of $f_M(x)$ is in the most recent 10 blocks. Looking more closely, about 5.52 percent of the mass of $f_S(x)$ is in the most recent 3 blocks; for $f_m(x)$ the corresponding figure is 0.576 percent. Therefore, the probability of zero mixins of 255 being drawn from the most recent 3 blocks is:

$(1 - 0.00576)^{255} = 0.229$

And since the real spend and the mixins are independent, the following is the probability that no mixins were drawn from the most recent 3 blocks, but the real spend does come from one of the first three blocks:

$0.229 \cdot 0.0552 = 0.01265$

So just the first 3 blocks contribute, in a sense, 1.265 percentage points of the 8.5 percentage points of total guessibility with ring size 256.

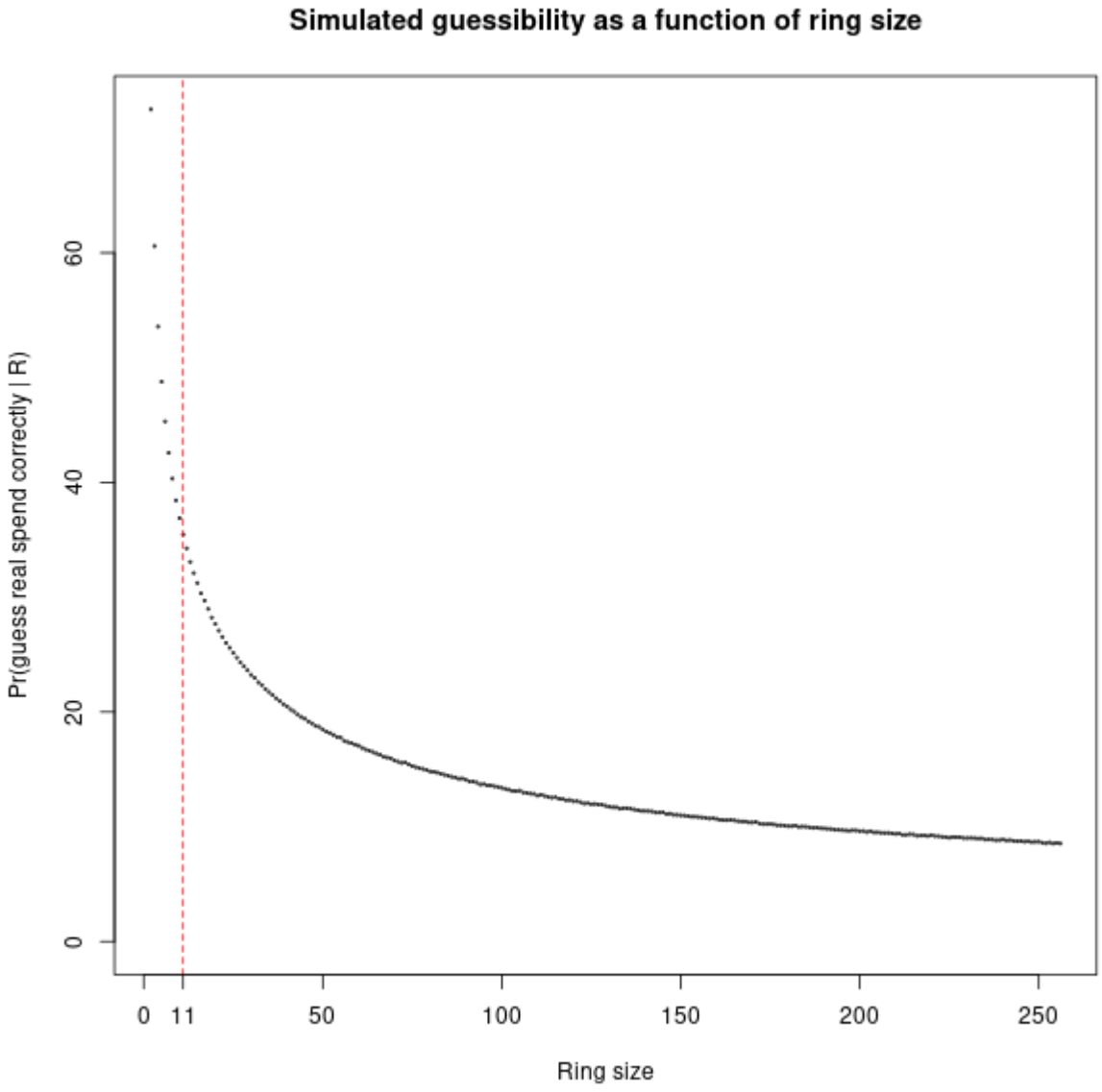
3.4 Enhancements and extensions of the RRA

Isthmus believes that certain enhancements of the RRA are possible. These take two forms:

1. If particular nonstandard forms of $f_M(x)$ can be identified, then transaction "chains" that can clearly be identified as using the nonstandard wallets would be particularly vulnerable to analysis.

2. The RRA could be combined with attack outlined in MRL-0011 so as to:

Fig. 2:


Simulated guessibility as a function of ring size
Pr(guess real spend correctly | R)
0
20
40
60
0
11
50
100
150
200
250
Ring size

(a) inform the edge weights with newfound knowledge of $f_S(x)$; and

(b) use "matches" from the RRA to inform the initial guesses for the MRL-0011 attack.

Isthmus believes that combining the RRA with MRL-0011 could be "potentially catastrophic".

Recommendation III: Information about the Rucknium Ratio Attack should never be released publicly, even after a "patch". Due to the distributed and immutable nature of Monero's blockchain, all transactions since 2018 would remain vulnerable to the attack regardless of any actions taken by the Monero Project or users now or in the future. Knowledge about the attack should only be revealed to select trusted individuals on a need-to-know basis. I would go further and say that all information in Section 2, especially equations (1) and (2), should also not be made public. This is a bit tricky since an improved mixin selection algorithm would be based on Section 2. However, revealing the methods in Section (2) are not necessary to implement a "patch" in the code, at least for OSPEAD (explained later), so the only tricky part is to have users accept "just trust us" as a response to any queries about how we are estimating $f_S(x)$.

Concealing information about how to develop an attack would not prevent the Monero Project from issuing an advisory to users that certain past behavior, such as spending outputs before z blocks had been confirmed, could expose them to risk — and therefore they should take whatever action necessary to protect themselves from exposure in the event that an adversary were to develop the RRA or similar attack. Any such public advisory regarding a specified z number of block confirmations could be based on an analysis of the Rucknium Ratio, yet any hint of the existence of the Rucknium Ratio and its significance can be excluded from such an advisory.

4 Mitigation of the vulnerability

The scope for mitigation of the vulnerability without changes to the mixin selection algorithm is probably limited. Transactions already confirmed on the blockchain obviously cannot be altered and would remain vulnerable to the RRA forever. There may be some user-level actions available to mitigate the vulnerability for individual users, but a general recommendation to all Monero users could be counterproductive. If, for instance, a general recommendation was issued to wait until z blocks had been confirmed before spending an output in order to avoid the high Rucknium Ratio for young outputs, then what would likely happen would be a pile-up of the density of $f_S(x)$ at $x = z$. This would result in arise of the Rucknium Ratio at point z, which would create the same vulnerability all over again, but just at a different x value. The Monero Project could hardly issue a recommendation to users of "Please coordinate among yourselves so as to spend your outputs at a roughly random uniform distribution between z_a and z_b blocks".

jberman's recent fix to the bug in the wallet that prevented applying the gamma distribution from the chaintip (PR #7821) does alleviate the problem somewhat by reducing the Rucknium Ratio for the most recent blocks. However, even with the fix, the problem remains

The only way forward, as far as I see it, is to launch a research project whose goal is to fix the mixin algorithm itself.

Recommendation IV: The Monero Project should launch a research project to develop a nonparametric estimate of $f_S(x)$ so that an "optimal" $f_M(x)$ may be constructed. The most obvious candidate to lead that research project is myself.

5 Long-term solution: Nonparametric estimation of $f_S(x)$ for an improved $f_M(x)$

So, how do we fix the mixin selection algorithm? To me, the obvious way is to construct $f_M(x)$ so that it matches $f_S(x)$ as closely as possible. For a $f_S(x)$ that is composed of the real-life actions of a heterogeneous group of human Monero users, there is no parametric method that can be used to construct $f_M(x)$ so that it becomes arbitrarily close to $f_S(x)$. The gamma distribution fitted by Maximum Likelihood estimation (MLE) in Möser et al. (2018) is one such parametric method that comes up short.

Fortunately for us, there is an entire class of methods focused on this specific task: nonparametric estimation of probability density functions. I will focus on kernel density estimation since it's the most commonly used and understood, being similar to the construction of histograms. However, there are specific characteristics of the Monero real spend age distribution that may suggest other methods such as splines, series, and local polynomial density estimators. (Getting into those details is beyond the scope of this document.)

What follows is a simulation comparing the convergence characteristics of parametric and nonparametric estimators of probability density functions.

Simulation setup (It is not crucial to understand this, but I include it for the sake of completeness): Let $g(x)$ be a PDF of a mixture distribution. The mixture is defined as follows:

$g(x) = 0.2 \cdot p_1(x) + 0.8 \cdot p_2(x)$

where

$p_1(x)$ is the PDF of an exponential distribution with rate parameter equal to 20; and

$p_2(x)$ is the PDF of a chi-squared distribution with 2 degrees of freedom.

This mixture distribution was not chosen to closely match all the characteristics of the Monero spend-age distribution closely. It slightly resembles it in that the bulk of the observations are near zero, with a long, thin right tail. For ease of plotting, I truncated the distribution above 5. The overall point of defining a mixture distribution for the purposes of this illustration is to have a distribution that cannot be properly estimated by any single parametric distribution family, but still have a distribution whose true underlying theoretical distribution is still known, which is useful for statistical testing purposes.

By simulation, I drew 100,000 observations from $g(x)$. Given that about 140,000 transactions are being confirmed on the Monero blockchain every week, this is a reasonable sample size for estimation. The R code that generates these results is available upon request.

Figure 3 displays a histogram of the 100,000 observations as well as a gamma distribution fitted via Maximum Likelihood Estimation (MLE). The estimated shape and scale parameters of the gamma distribution are 0.61 and 2.03, respectively (the exact value are unimportant). The gamma distribution comes reasonably close to the histogram, but the match is not great.

An alternative estimator of $g(x)$ is a kernel density estimator. Figure 4 displays a comparison of the MLE gamma estimate of $g(x)$, the kernel density estimate, and the true $g(x)$. From this we can see that the kernel density estimator performs quite well, while the gamma MLE curve traces out significant gaps.

This can be seen more clearly in a plot of the estimated cumulative distribution function (CDF) of the gamma MLE and the empirical distribution of the simulated data drawn from $g(x)$, as in Figure 5. The Kolmogorov–Smirnov statistic for the difference between these two distributions is 0.0615. By contrast, the CDFs of the kernel density estimate and the empirical distribution of the simulated data are difficult to distinguish visually in Figure 6. The Kolmogorov–Smirnov statistic here is 0.0094, which is nearly an order-of-magnitude improvement over the MLE gamma estimate. Note that here kernel density estimation is operating at a disadvantage since the true

Fig. 3:

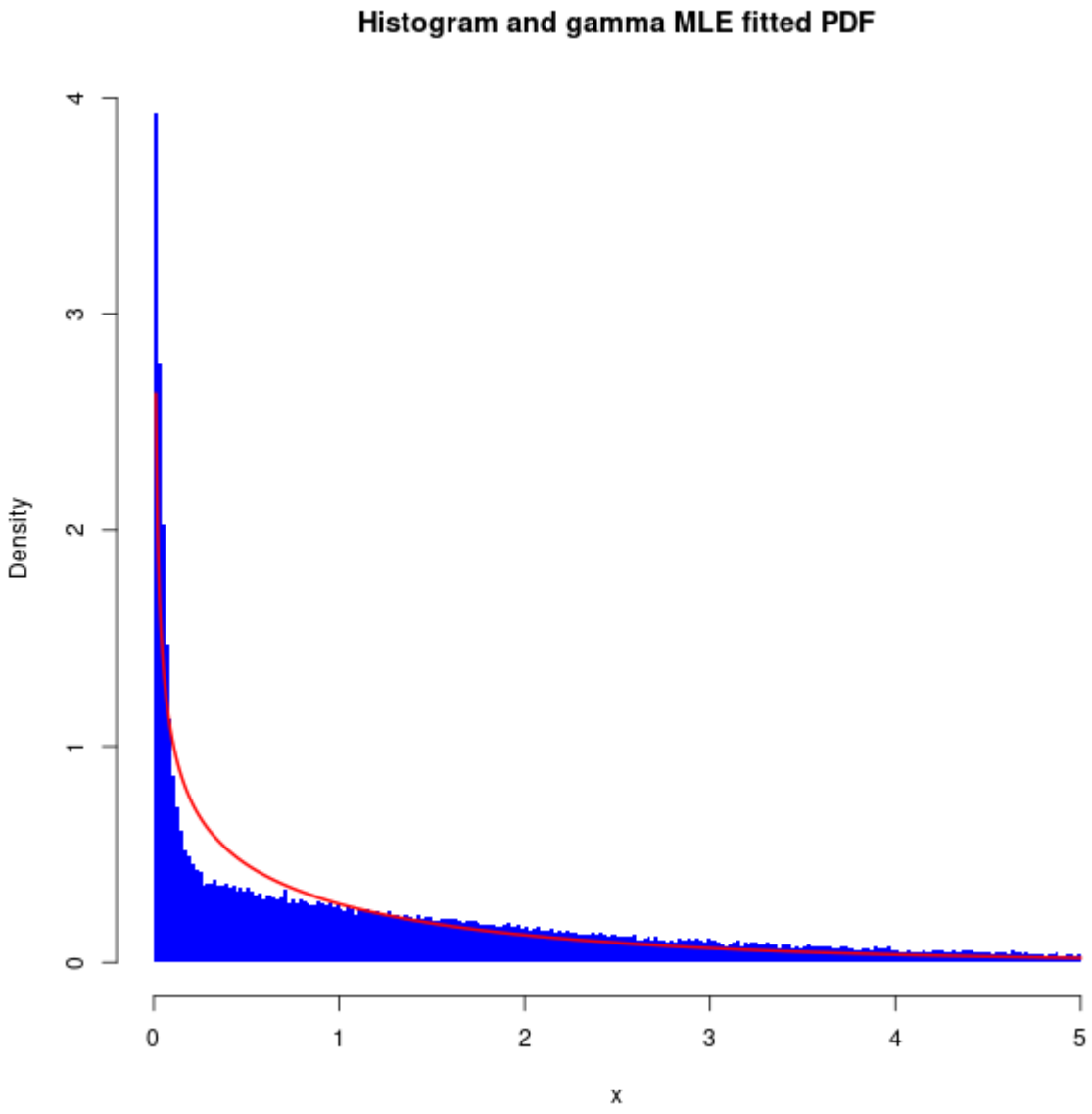
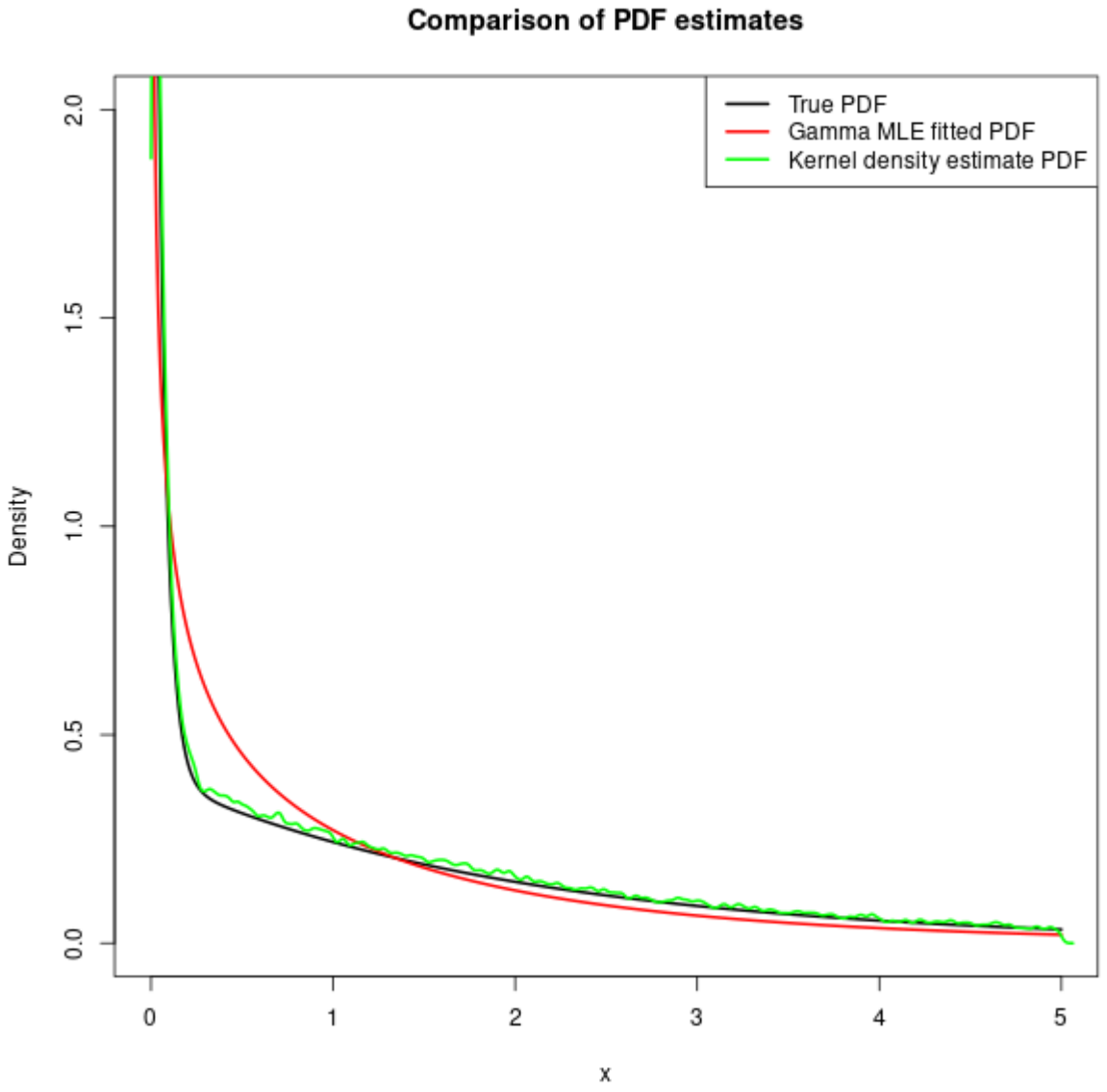


Fig. 4:


Comparison of PDF estimates
True PDF
Gamma MLE fitted PDF
Kernel density estimate PDF
Density
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
5
x

distribution is very discontinuous at 0.

6 Statistical properties of nonparametric estimators of PDFs

Let h be the bandwidth of $\hat{f_S}(x)$, a kernel density estimator of $f_S(x)$. Under the assumption that $f_S(x)$ is continuous (an assumption that is approximately correct for the empirical Monero output spend-age data, except at the zero boundary) with n being the number of observations used in the estimation, it can be shown that

$$\text{as } h \to 0 \text{ and } nh \to \infty, \hat{f_S}(x) \xrightarrow{p} f_S(x) \tag{6}$$

i.e. the estimator $\hat{f_S}(x)$ converges in probability to the true distribution $f_S(x)$.

Therefore, we have this wonderful result that with a sufficiently large sample n (and h chosen appropriately), the difference between $\hat{f_S}(x)$ and $f_S(x)$ shrinks toward zero. Setting $f_M(x)$ to this nonparametric kernel density estimator $\hat{f_S}(x)$ achieves the goal of minimizing the leakage of statistically meaningful data in a very simple way. We are done! Sort of.

So, why aren't nonparametric methods more widely used in statistical analysis? I can think of a few reasons:

1. Your sample size needs to be quite large. As is typical in statistics, here you do not get something in exchange for nothing. Unsurprisingly, when you assume more information about your distribution, which is the case with parametric methods, you need less information from the empirical sample, so your sample size can be small. Of course, if you guess wrong, your method falls apart badly, as we have seen. In technical terms, parametric estimators generally converge at rate $n^{-1/2}$ while nonparametric estimators converge at a rate slower than $n^{-1/2}$. With a small sample size, both the bias and variance terms in nonparametric estimators can be large. Fortunately, the sample size of Monero transactions is relatively large, so this concern does not apply.

2. Since you do not obtain a finite set of parameters, nonparametric methods can inhibit interpretation of results. Many questions in the natural and social sciences are best answered with a statement about the estimated value of a specific parameter of a distribution. Non parametric methods may give you interesting pictures, but their meaning is often difficult to interpret. This shortcoming of nonparametric methods does not really apply to the mixin selection algorithm. In a certain sense we want to thwart any interpretation, in fact.

3. Your research question may not involve the distribution at all, but just the mean or some other point estimand. This does not apply to our effort to overhaul the mixin selection algorithm, since applying the mean is not enough to conceal real spends.

4. The object under study is multivariate. This issue is similar to the sample size requirements. The complexity of nonparametric methods increase exponentially, literally, with the number of variables whose joint PDF you need to estimate. The curse of dimensionality hits nonparametric methods hard. Since we are only dealing with a univariate probability distribution, this problem also does not inhibit us from pursuing nonparametric methods.

5. Non-scientific practical issues. Nonparametric methods are yet another entire class of statistical methods that need to be learned about. Researchers just might not be bothered. Peer reviewers may not be familiar with them. A conference paper may need to be submitted on a tight deadline and it may be easier to fall back on

Fig. 5:

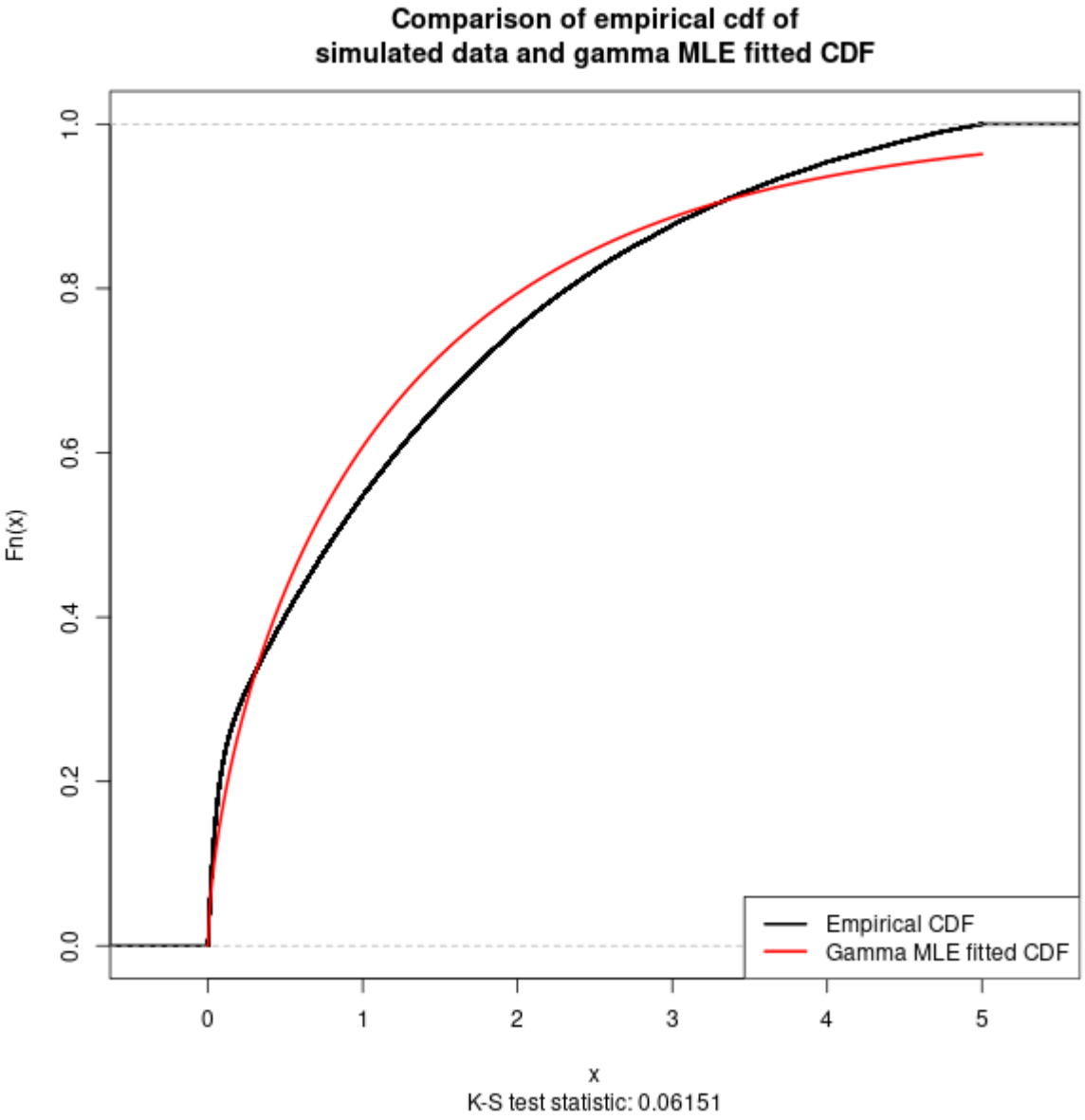
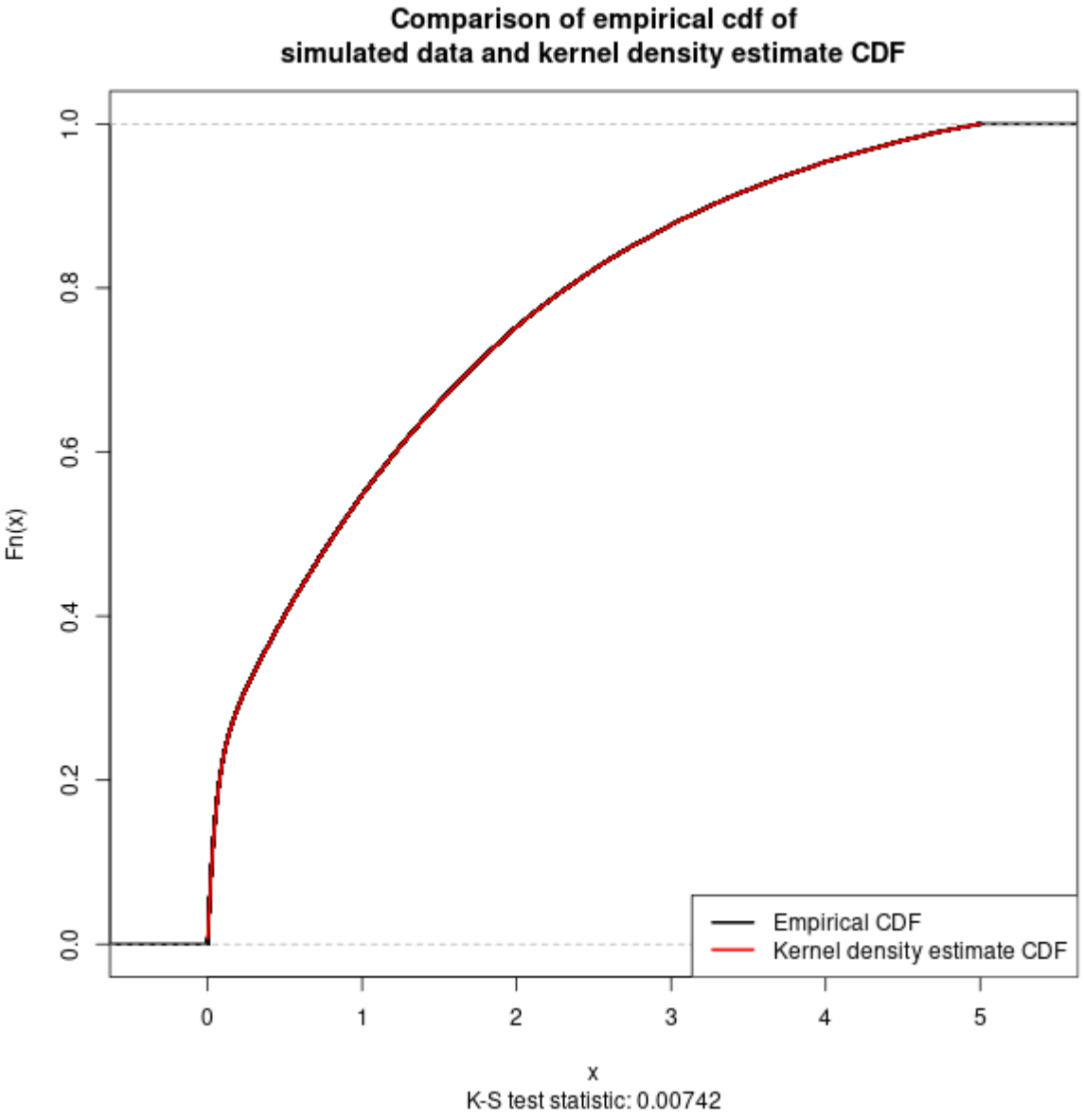

Comparison of empirical cdf of
simulated data and gamma MLE fitted CDF
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
x
Empirical CDF
Gamma MLE fitted CDF
K-S test statistic: 0.06151

Fig. 6:


Comparison of empirical cdf of
simulated data and kernel density estimate CDF
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
x
Empirical CDF
Kernel density estimate CDF
K-S test statistic: 0.00742

familiar methods. None of these issues should concern us; let's reach for the highest standards of rigor that we can.

Although nonparametric estimation of a univariate probability density function is relatively basic and well-studied as far as statistical techniques go, we will face many challenges and choices. Among them are:

1. Which type of nonparametric estimator to use. This could be kernel density, splines, series, or local polynomial density estimators, and maybe more that I am unaware of.

2. How to determine the tuning parameters. With kernel density estimation, this is the bandwidth h. Other estimators have their corresponding tuning parameters. There are many different approaches to determining the value of the tuning parameters, several of them involving cross-validation techniques. The literature has many suggestions about how to do this, but none are really "best" for all circumstances, so we will have to investigate many avenues and see how they fit into our particular problem.

3. How to deal with the fact that $f_S(x)$ is literally a moving target. $f_S(x)$ is not static. Presumably, it is evolving through time as usage patterns change. Therefore, $f_S(x)$ is actually a set of probability distributions indexed by time t as $f_{S,t}(x)$.

7 OSPEAD: A parametric approach for implementation in the medium term

The nonparametric approach to the mixin selection algorithm will take a lot of time to develop and test. In the medium term, it may make sense to develop a parametric approach that is a large improvement on the current algorithm, yet is feasible to develop and deploy quickly. I outline such an approach below.

We have seen that parametric approaches will never be perfect. Inevitably, they will fail to match the true spend distribution in some ways, large or small. Due to this unfortunate reality, choosing among various parametric options directly implies choosing to more strongly protect the privacy of some users — and grant less protection to other users. Or more precisely, the spend-age of some real outputs are better obfuscated by a greater number of mixins. These tradeoffs cannot be avoided, and so they must be handled somehow. We may not have asked to play god with users' privacy, but playing god is inevitable.

Choosing which users to protect ultimately is not a technical question. It is a normative question. That is to say, it is not a question about *what is*. "What is" can often be answered with enough research effort by technicians. Normative questions are concerned with what ought to be. Within a traditional framework, "society" or "policymakers" would answer these questions. Within the Monero context, the groups answering these questions may be Monero devs, the Monero Research Lab, and Monero community members who are willing to understand the issues at hand. Technical researchers can present the answers to positive questions that help inform the deliberations about the normative questions, but they alone cannot make the final call on these matters.

The discipline of economics tackles frequently these questions about trading off one individual's welfare for that of another, so I will bring in some economic concepts. But first, some mathematical setup. As before,

Let $f_S(x)$ denote the probability density function (PDF) of the age of the real spend outputs of transactions on the Monero blockchain.

Let $f_M(x)$ denote the PDF of the age of the mixins used by the reference wallet software for Monero. Now define the following function:

$h(x) = \max\{0, f_S(x) - f_M(x)\}$

At any particular value x, then, $h(x)$ is the "privacy deficit" suffered by a user who chooses to spend an output that has age$x$. For the current purposes, additional privacy coverage above the target level of $f_S(x) = f_M(x)$ will be considered of no benefit, partly because for any interval on the support of $f_S(x)$ and $f_M(x)$ for which $f_S(x) < f_M(x)$ inevitably results in another interval that will have a privacy deficit, due to the zero-sum nature of the probability distributions.

How do we construct $f_M(x)$ for best overall privacy if we restrict ourselves to parametric distributions, and therefore cannot achieve $f_S(x) \approx f_M(x)$ for all values of x? Well, it depend on how we define "best". Below are six approaches. I have the mathematical definitions of these worked out in my head, but they are not written here:

1) Privacy impoverishment
2) Economic welfare
3) Inequality minimization
4) Worst-case-scenario minimization
5) Maximum Likelihood Estimation
6) Maximize resistance to a specific attack (likely the RRA) [this item may be redacted before release]

Now that a set of objective functions have been defined, one could imagine optimizing these objective functions in a numerical optimization procedure where each parametric distribution family with support of $[0, \infty)$ is permuted over 1-6 above.

I call this approach Optimal Static Parametric Estimation of Arbitrary Distributions (OSPEAD), which is an acronym that I just made up so that it can be referred to by a name in discussions in the Monero Research Lab and the wider Monero community. It is "Optimal" in some sense, at least in reference to one of the six criteria outlined above. It is "Static" in that it is not dynamic. An estimate is made once, then implemented and released, and is not adjusted depending on temporal shifts in $f_S(x)$. It is a "Parametric Estimation" since it is parametric. The estimation is done of an "Arbitrary Distribution" since the real spend distribution is not a specific parametric one, but is in some sense "arbitrary".

Recommendation V: As a medium-term solution to "stop the bleeding", the Monero Project should work to operationalize Optimal Static Parametric Estimation of Arbitrary Distributions (OSPEAD). Implementation of OSPEAD would require to completion of much of the initial work to estimate $f_S(x)$ that would also be required for the nonparametric approach, so not much extra labor would need to be expended to implement OSPEAD.

A Appendix: Data for Figure 2: Simulated guessibility as a function of ring size

ring.size	guessibility	ring.size	guessibility	ring.size	guessibility	ring.size	guessibility
2	72.5012	41	20.2236	80	14.8169	119	12.3242
3	60.5882	42	20.0369	81	14.7812	120	12.2153
4	53.5727	43	19.7598	82	14.7143	121	12.2247
5	48.7711	44	19.554	83	14.6116	122	12.1439
6	45.3109	45	19.4144	84	14.5225	123	12.02
7	42.573	46	19.1737	85	14.4465	124	12.0437
8	40.331	47	19.009	86	14.3477	125	11.9673
9	38.4334	48	18.7778	87	14.2993	126	11.9857
10	36.8847	49	18.7009	88	14.1777	127	11.9378
11	35.4512	50	18.4791	89	14.1945	128	11.9138
12	34.2561	51	18.2825	90	14.0879	129	11.8292
13	33.0742	52	18.1669	91	13.9317	130	11.7355
14	32.0973	53	18.0036	92	13.9423	131	11.6984
15	31.2342	54	17.8241	93	13.8525	132	11.6925
16	30.3308	55	17.7633	94	13.6843	133	11.5679
17	29.7038	56	17.4893	95	13.7098	134	11.6183
18	28.9702	57	17.3701	96	13.5968	135	11.611
19	28.2215	58	17.2866	97	13.5883	136	11.5642
20	27.6658	59	17.1557	98	13.5198	137	11.4979
21	27.0826	60	17.057	99	13.4686	138	11.4401
22	26.5358	61	16.8666	100	13.3782	139	11.3759
23	26.0151	62	16.7201	101	13.3063	140	11.3705
24	25.5879	63	16.6219	102	13.2141	141	11.3613
25	25.1412	64	16.5125	103	13.1253	142	11.3329
26	24.7264	65	16.3744	104	13.1035	143	11.2314
27	24.3091	66	16.2666	105	13.1477	144	11.2338
28	23.9704	67	16.0869	106	12.9838	145	11.2552
29	23.6244	68	16.0288	107	12.9447	146	11.112
30	23.223	69	15.9613	108	12.8793	147	11.1354
31	22.969	70	15.7724	109	12.8533	148	11.0326
32	22.5744	71	15.6817	110	12.7135	149	11.0269
33	22.3436	72	15.5569	111	12.7847	150	10.9913
34	21.9896	73	15.5948	112	12.678	151	10.9753
35	21.726	74	15.4265	113	12.5874	152	10.9195
36	21.476	75	15.2845	114	12.5281	153	10.8727
37	21.1885	76	15.1892	115	12.5641	154	10.869
38	20.9454	77	15.0941	116	12.4443	155	10.8385
39	20.679	78	15.0289	117	12.4087	156	10.8104
40	20.4868	79	14.9434	118	12.3086	157	10.7796

A APPENDIX: DATA FOR FIGURE 2: SIMULATED GUESSIBILITY AS A FUNCTION OF RING SIZE

ring.size	guessibility	ring.size	guessibility	ring.size	guessibility
158	10.7091	197	9.6471	236	8.8911
159	10.737	198	9.6864	237	8.8736
160	10.6777	199	9.6766	238	8.8214
161	10.5986	200	9.6163	239	8.8351
162	10.5744	201	9.6361	240	8.8778
163	10.5658	202	9.5289	241	8.8049
164	10.6061	203	9.6106	242	8.8286
165	10.5429	204	9.5042	243	8.7642
166	10.4849	205	9.5099	244	8.7327
167	10.4605	206	9.47	245	8.7567
168	10.4554	207	9.4584	246	8.6941
169	10.3781	208	9.4641	247	8.7275
170	10.3693	209	9.4164	248	8.6634
171	10.4257	210	9.3728	249	8.6805
172	10.278	211	9.4003	250	8.69
173	10.2189	212	9.2914	251	8.6107
174	10.2221	213	9.2962	252	8.5447
175	10.2294	214	9.3481	253	8.6345
176	10.1971	215	9.3238	254	8.5343
177	10.1301	216	9.1997	255	8.5823
178	10.1307	217	9.2272	256	8.5382
179	10.1001	218	9.2353		
180	10.0472	219	9.1923		
181	10.0499	220	9.2664		
182	10.0953	221	9.182		
183	9.9882	222	9.1696		
184	10.0144	223	9.1212		
185	9.9966	224	9.0993		
186	9.9496	225	9.0391		
187	9.8963	226	9.0986		
188	9.9196	227	9.0873		
189	9.8669	228	9.0834		
190	9.8523	229	9.0083		
191	9.8212	230	9.0087		
192	9.8059	231	9.0171		
193	9.7326	232	9.0024		
194	9.7259	233	8.9709		
195	9.7009	234	8.9839		
196	9.6776	235	8.8782		

B Appendix: Rucknium Ratio Attack Monte Carlo R Code

```
xmr <- read.csv("output_age_data.csv", stringsAsFactors = FALSE)
# From https://github.com/monero-project/monero/files/6968268/output_age_data.zip
xmr$Observed.pdf <- xmr$Observed/sum(xmr$Observed)
# Convert f(x) to a "probability density function", more or less
xmr$Current.decoy.selection.algo.pdf <- xmr$Current.decoy.selection.algo /
sum(xmr$Current.decoy.selection.algo)
# Do the same for f_M(x)
alpha <- 10/11
xmr$f_S <- (1/(1-alpha)) *
(xmr$Observed.pdf - alpha * xmr$Current.decoy.selection.algo.pdf)
# Construct f_S(x) according to equation (2) in the paper
xmr$f_S[xmr$f_S < 0] <- xmr$Current.decoy.selection.algo.pdf[xmr$f_S < 0]
# Sometimes the observed can be below what the idealized algorithm would have
# selected since there is noise in the long tails.
xmr$f_S <- xmr$f_S / sum(xmr$f_S)
# Make f_S(x) be a proper PDF
xmr$Rucknium.Ratio <- xmr$f_S / xmr$Current.decoy.selection.algo.pdf
png("Rucknium-Ratio.png", width = 600, height = 600)
par(mar = c(5, 6, 4, 2) + 0.1)
# c(5, 4, 4, 2) + 0.1
plot(xmr$Rucknium.Ratio[1:10000], log = "x", cex = 0.5,
col = rgb(0, 0, 0, alpha = c(rep(1, 100), rep(0, 9900))),
main = "Rucknium Ratio\nDots are partially transparent for better visualization",
xlab = "Age of blocks corresponding to ring members (log scale)",
ylab = expression("Rucknium Ratio " * frac(f[S](x), f[M](x))) )
lines(x = 1:100, y = xmr$Rucknium.Ratio[1:100])
points(x = 101:10000, y = xmr$Rucknium.Ratio[101:10000], col = rgb(0, 0, 0, alpha = 0.2), pch
= ".")
abline(h = 1, lty = 2, col = "red")
axis(2, 1, "1")
dev.off()
par(mar = c(5, 4, 4, 2) + 0.1 )
sum(xmr[1:10, "f_S"])
# [1] 0.1380584
sum(xmr[1:10, "Current.decoy.selection.algo.pdf"])
# [1] 0.02727423
sum(xmr[1:3, "f_S"])
# [1] 0.05523403
sum(xmr[1:3, "Current.decoy.selection.algo.pdf"])
# [1] 0.005761155
set.seed(314)
n.rings <- 10000000
# 10 million
```

```
simulation.mixin.quantity <- 10
rings <- matrix(c(
sample(nrow(xmr), size = simulation.mixin.quantity * n.rings, replace = TRUE,
prob = xmr$Current.decoy.selection.algo.pdf),
sample(nrow(xmr), size = 1 * n.rings, replace = TRUE, prob = xmr$f_S)
), byrow = FALSE, ncol = simulation.mixin.quantity + 1)
attack.prob.11 <- apply(rings, 1, FUN = function(x) {
which.max(xmr$Rucknium.Ratio[x])
})
# Note that which.max() returns the index of the "first" maximum if the maximum
# is not unique.  Therefore, the real spend was chosen to be the "last"
# element of the set so that it would be clear that the real spend, if
# guessed, was the result of unique guess
t(t(100 * prop.table(table(attack.prob.11)) ))
# Main result
#####################################################################
# Simulation for different ring sizes follows
# Note:  This takes about an hour to run
#####################################################################
set.seed(314)
n.rings <- 1000000
# 1 million
max.ring.size <- 256
guessibility.results <- list()
for (i in seq_len(max.ring.size - 1)) {
simulation.mixin.quantity <- i
rings <- matrix(c(
sample(nrow(xmr), size = simulation.mixin.quantity * n.rings, replace = TRUE, prob =
xmr$Current.decoy.selection.algo.pdf),
sample(nrow(xmr), size = 1 * n.rings, replace = TRUE, prob = xmr$f_S)
), byrow = FALSE, ncol = simulation.mixin.quantity + 1)
xmr$Rucknium.Ratio <- xmr$f_S / xmr$Current.decoy.selection.algo.pdf
attack.prob <- apply(rings, 1, FUN = function(x) {
which.max(xmr$Rucknium.Ratio[x])
})
guessibility.results[[i]] <- t(t(100 * prop.table(table(attack.prob)) ))
print( guessibility.results[[i]])
}
guessibility <- sapply( guessibility.results, FUN = function(x) {
x[nrow(x), ]
})
guessibility <- unname(guessibility)
```

```
367 png("Guessibility-by-ring-size.png", width = 600, height = 600)
plot(seq_along(guessibility) + 1, guessibility, ylim = c(0, max(guessibility)),
cex = 0.25,
main = "Simulated guessibility as a function of ring size",
xlab = "Ring size",
ylab = "Pr(guess real spend correctly | R)")
abline(v = 11, lty = 2, col = "red")
axis(1, 11, "11")
dev.off()
guessibility.data.frame <- data.frame(ring.size = seq_along(guessibility) + 1, guessibility =
guessibility)
summary(lm(log(guessibility) ~ ring.size, data = guessibility.data.frame))
write.csv(guessibility.data.frame,
file = "Guessibility-by-ring-size-data.csv", row.names = FALSE)
```

C Appendix: Comparison of Maximum Likelihood Estimation and Nonparametric Estimation of Simulated Mixture Distribution

```
# install.packages("distr")
# install.packages("fitdistrplus")
# install.packages("ks")
# library(ks)
# library(distr)
# library(fitdistrplus)
set.seed(314)
n.obs <- 100000
x.trunc <- 5
mixed.dist <- distr::UnivarMixingDistribution(
distr::Exp(20),
distr::Chisq(df = 2),
mixCoeff = c(0.2, 0.8))
r.mixed.dist <- distr::r(mixed.dist)
d.mixed.dist <- distr::d(mixed.dist)
x <- r.mixed.dist(n.obs)
x <- x[x <= x.trunc]
length(x)
summary(x)
hist(x, breaks = 200, freq = FALSE, col = "blue", border = NA)
mle.gamma <- fitdistrplus::fitdist(x, "gamma", start = list(shape = 2, scale = 2))
png("Histogram-and-fitted-gamma-PDF.png", width = 600, height = 600)
hist(x, breaks = 200, freq = FALSE, col = "blue", border = NA,
main = "Histogram and gamma MLE fitted PDF")
lines(
seq(0, x.trunc, by = 0.01),
dgamma(seq(0, x.trunc, by = 0.01), shape = mle.gamma$estimate["shape"], scale =
mle.gamma$estimate["scale"]),
col = "red", lwd = 2)
dev.off()
kde.est <- ks::kde(x, gridsize = 5000)
x.seq <- seq(0, x.trunc, length.out = 500)
```

```
371 png("Comparison-of-PDF-estimates.png", width = 600, height = 600)
plot(0, 0, xlim = c(0, x.trunc),
ylim = c(0, 2),
col = "transparent",
xlab = "x",
ylab = "Density",
main = "Comparison of PDF estimates")
legend( "topright",
legend = c("True PDF", "Gamma MLE fitted PDF", "Kernel density estimate PDF"),
col = c("black", "red", "green"),
lty = 1, lwd = 2)
lines(x.seq, d.mixed.dist(x.seq), col = "black", lwd = 2)
lines(x.seq,
dgamma(x.seq, shape = mle.gamma$estimate["shape"], scale = mle.gamma$estimate["scale"]),
col = "red", lwd = 2)
lines(kde.est$eval.points[kde.est$eval.points > 0],
kde.est$estimate[kde.est$eval.points > 0], col = "green", lwd = 2)
dev.off()
ks.test.gamma<- ks.test(x, pgamma, shape = mle.gamma$estimate["shape"], scale =
mle.gamma$estimate["scale"])
ks.test.gamma
png("Comparison-of-CDF-gamma.png", width = 600, height = 600)
plot(ecdf(x), do.points = FALSE, col = "black", lwd = 2,
main = "Comparison of empirical cdf of\nsimulated data and gamma MLE fitted CDF",
sub = paste0("K-S test statistic:  ", round(ks.test.gamma$statistic, digits = 5)))
lines(seq(0, x.trunc, by = 0.01), col = "red", lwd = 2,
pgamma(seq(0, x.trunc, by = 0.01), shape = mle.gamma$estimate["shape"], scale =
mle.gamma$estimate["scale"]))
legend( "bottomright",
legend = c("Empirical CDF", "Gamma MLE fitted CDF"),
col = c("black", "red"),
lty = 1, lwd = 2)
dev.off()
kde.cdf.est <- ks::kcde(x, gridsize = 5000)
kde.cdf.est.fun <- approxfun(kde.cdf.est$eval.points, kde.cdf.est$estimate, yleft = 0, yright
= 0)
ks.test.kde <- ks.test(x, kde.cdf.est.fun)
ks.test.kde
```

```
372 png("Comparison-of-CDF-kernel-density.png", width = 600, height = 600)
    plot(ecdf(x), do.points = FALSE, col = "black", lwd = 2,
    main = "Comparison of empirical cdf of\nsimulated data and kernel density estimate CDF",
    sub = paste0("K-S test statistic:  ", round(ks.test.kde$statistic, digits = 5)))
    lines(seq(0, x.trunc, by = 0.01),
    kde.cdf.est.fun(seq(0, x.trunc, by = 0.01)), col = "red", lwd = 2)
    legend( "bottomright",
    legend = c("Empirical CDF", "Kernel density estimate CDF"),
    col = c("black", "red"),
    lty = 1, lwd = 2)
    dev.off()
```