RESEARCH BULLETIN MRL-0004

Improving Obfuscation in the CryptoNote Protocol

26 January 2015

Adam Mackenzie*, Surae Noether and Monero Core Team

*Correspondence:
lab@getmonero.org
Monero Research Lab

Abstract

We identify several blockchain analysis attacks available to degrade the untraceability of the CryptoNote 2.0 protocol. We analyze possible solutions, discuss the relative merits and drawbakcs to those solutions, and recommend improvements to the Monero protocol that will hopefully provide long-term resistance of the cryptocurrency against blockchain analysis. Our recommended improvements to Monero include a protocol-level network-wide minimum mix-in policy of $n = 2$ foreign outputs per ring signature, a protocol-level increase of this value to $n = 4$ after two years, and a wallet-level default value of $n = 4$ in the interim. We also recommend a torrent-style method of sending Monero output. We also discuss a non-uniform, age-dependent mix-in selection method to mitigate the other forms of blockchain analysis identified herein, but we make no formal recommendations on implementation for a variety of reasons. The ramifications following these improvements are also discussed in some detail.

This research bulletin has not undergone peer review, and reflects only the results of internal investigation.

1 Overview

We address several security concerns in traceability when using a ring signature approach to ensure the privacy of digital currency transactions. Recall that the CryptoNote protocol proposal proves unconditional unlinkability, but untraceability remained unproven; indeed, CryptoNote is very traceable, as we demonstrate in this document.

We remind the reader that we say that two transactions are *unlinkable* when an observer cannot prove that those two transactions were sent to the same user. We say that a transaction is considered *untraceable* if all possible senders of a transaction are equiprobable. Hence, the problems with untraceability in CryptoNote suggest that, while users can *receive* CryptoNote-based cryptocurrencies with no concern for their privacy, they cannot necessarily *spend* those currencies without releasing some information about their past transactions. Of course, without a non-interactive zero-knowledge (NIZK) approach (see, for example, [2]), traceability is inevitable. However, NIZK approaches to date have had limited applications and suffer the disadvantage of computational cost. Addressing some of the security issues related to traceability without zero-knowledge techniques is the main goal of this document. Critically, we ask the reader to observe that the techniques presented do not violate

users' capabilities to provide auditable histories of their transactions in order to comply with local laws (see Section 6).

In Section 2, we review how *spending in the clear*, or *zero-mix spending* leads to traceability, as detailed in [3]. In Section 3, we identify three further routes of attack that may be used to reduce untraceability in CryptoNote, although we make no claims that this list is exhaustive. In particular, Section 3.1 details how the age of transaction outputs can be used to remove untraceability from CryptoNote. Section 3.3 details how observable data about the transactions used in a ring signature can be used in a combinatorial analysis to associate transaction outputs that are otherwise not associated together. Section 3.2 details how careless usage of transaction outputs together in a new transaction can lead an attacker to draw conclusions about the ownership of the original transaction outputs; we call this an association-by-use attack.

In Section 4, we detail some recommendations to mitigate these avenues of attack, and for other recommendations we merely make a sketch. In Section 4.1, we recommend a mandatory, network-wide minimum mix-in of $n = 2$, and we justify this choice. In Sections 4.2 and 4.3, we detail our recommendations for mitigating the combinatorial and association-by-use attacks described in Sections 3.3 and 3.2, respectively.

In Section 5 we detail our roadmap to implementing our policy suggestions. Finally, in Section 6, we discuss a method that a user can use to comply with local laws without sacrificing the privacy we have worked so hard to attain.

2 Traceability with Zero Mix-In Spending

In this section, we list several problems with untraceability in CryptoNote and other ring signature schemes. The issues detailed in [3] are not comprehensive, of course, and we describe other issues in this document. We make no claims that the issues presented in this document are comprehensive, either. In Section 3.1, we describe age-related attacks on transaction outputs, in Section 3.3, we describe certain combinatorial attacks that can be used to force traceability using data external to the ring signatures such as block height, and in Section 3.2, we describe how careless usage of transaction outputs together can reveal to an eavesdropper further information about transaction ownership. The chain reactions caused by transactions spent in the clear, described in [3], form a good starting point of the discussion.

When using a CryptoNote-style ring signature protocol in a digital currency to obscure traceability and linkability, a malicious party with a large number of transaction outputs can cause a chain reaction in traceability by sacrificing their own anonymity, allowable through so-called zero mix transactions which have no foreign outputs used as mix-ins for the associated ring-signature. Before we proceed, we wish to establish some terminology for the reader. First, recall that in transparent cryptocurrencies such as Bitcoin, transaction outputs can be inspected to determine whether they have been spent or not. Indeed, the acronym UTXO is often used, meaning *unspent transaction output*. In CryptoNote, this acryonym would be inappropriate, as it is difficult to determine whether a transaction output has been spent or not without using some form of blockchain analysis or other form of traceability attack, such as the attacks described herein. As a consequence, we will not refer

to transaction outputs as spent or unspent within this document. Second, we shall refer to the practice of spending a transaction output with a ring signature that was generated using no foreign transaction outputs used as mix-ins by *spending in the clear* or as *zero-mix* spending. Spending a transaction in the clear means generating a ring signature with no mix-ins; the ring signature is equivalent to a usual digital signature in this case.

Our initial security concerns can be visualized quite simply. Just as in the original ring signature paper, [4], consider the situation in which an employee of a government leaks a secret by collecting the public keys of n other employees (mix-ins). The leaker then uses those keys and her own private key to generate a ring signature to sign the document that they are leaking. An observer will, apparently, only have a 1 in $n+1$ chance at guessing which employee leaked the secret. What happens if the employees who owned the mix-in keys begin vigorously denying that they leaked that secret? What happens as, one by one, those employees willfully reveal their private keys to prove to an authority figure, or simply to one another, that they aren't the leak? At first, your obfuscatory power drops from 1 of $n+1$ to 1 of n after the first employee reveals their key. After the second employee reveals their key, your obfuscatory power drops from 1 of n to 1 of $n-1$, and so on. In this scenario, ring signatures become significantly weaker. As we identified in [3], when ring signatures are iteratively used, transaction after transaction, the problem compounds. Through chain reactions, the security of one transaction can depend on the security of a transaction with no directly related parties. Of course, ring signatures are still valid as *digital signatures* go, and so their ability to act as the backbone of a cryptocurrency network is not in question. The only question is to address this degradation in untraceability.

To see an example of this, say that Alice fashions a ring signature using the mix-ins $\{\text{Bob}, \text{Cynthia}, \text{Doug}\}$. For any observer, then, the list of possible senders of the transaction is clearly $\{\text{Alice}, \text{Bob}, \text{Cynthia}, \text{Doug}\}$. If Bob, Cynthia, and Doug then each spend their transaction outputs with zero mix-ins, then it is obvious any any observer that they no longer own those transaction outputs. Hence, they can be excluded from the list of possible senders of Alice's transaction, and Alice is now exposed as the sender of her transaction. It is not as if only three users, Bob, Cynthia, and Doug, spent their transactions in the clear, but as if all four users spent their transactions in the clear. Hence, any signatures using Alice's transactions as a mix-in are now also compromised. If enough transactions are affected, there can be second-order effects, leading to a chain reaction.

We modeled the creation of ring signatures in [3] using a variant of the Pólya Urn Process described in depth in [1], and we used this model to demonstrate that chain reactions in traceability are possible. Consequently, if all of the mix-ins used to generate a new ring signature are controlled by a malicious party, then that malicious party has gained control over the untraceability of the new ring signature. The Pólya Urn Process suffers from the *rich get richer* effect; an attacker in control of many outputs at the beginning of the experiment will probabilistically control more and more as time goes on unless parameters are chosen carefully.

As a saving grace, within the CryptoNote protocol, users cannot be directly associated with transaction outputs. All that can be said is whether a key image is

associated with a transaction output. Indeed, the so-called *stealth address* element of the CryptoNote protocol ensures that this does not lead to a complete failure in anonymity. To be precise, when Bob, Cynthia, and Doug spend their transaction outputs in the clear, they are revealing to the world via the blockchain that their outputs belong to their associated key images. When this occurs, and when it is clear that Alice's key image was used in none of those transactions, her *key image* is now linked to the relevant output, not Alice's identity.

2.1 Change and Dust Force Zero Mix Spending

We now show how users are regularly forced to make transactions in the clear, regardless of their intent.

One may have reasonably assumed that the original CryptoNote developers would have implemented a standardized denomination requirement for all transaction outputs, but this is not so. For example, whereas 102.5 XMR would reasonably be represented as 100 XMR, 2 XMR and 0.5 XMR, the protocol does not disallow any strange denominations such as 100 XMR, 2 XMR, 0.499999999 XMR and 0.000000001 XMR. An output is unmixable either because it has been denominated unusually, such as our example with 0.499999999 XMR (and hence the transaction output set associated with the denomination is empty) or its amount is smaller than the minimum transaction fee, such as our example with 0.000000001 XMR.

Below is a simple example of dust generation using the current denomination code. In this example, Bob has 103.32111111 in his wallet and decides to send Alice 53.32111111 XMR with 2 foreign transaction outputs used as mix-ins while receiving 50 XMR in change. Let us presume that Bob's XMR is split up by denomination such that his 103.32111111 XMR consists of an output of 100 XMR, an output of 3 XMR, an output of 0.3 XMR, an output of 0.02 XMR, and an output of dust, 0.00111111 XMR. The current denomination code will take the output of 100 XMR that Bob owns, split it into two outputs, each for 50 XMR, and send one to Alice and one to Bob. It will then take all of the remaining outputs in Bob's wallet (3, 0.3, 0.02, 0.00111111) and send them to Alice, each signed with their own ring signature with the minimum mix-in level wherever possible. Unfortunately, the dust output, .00111111 is in a unique category in the sense that the transaction output set responsible for the denomination .00111111 is empty. Hence, no mix-ins are possible. We get a diagram like this:

$$\begin{aligned}
&100 \text{ XMR (Bob) (mix-in=2)} \longrightarrow 50 \text{ XMR (Alice)}, 50 \text{ XMR (Bob)} \\
&3 \text{ XMR (Bob) (mix-in=2)} \longrightarrow 3 \text{ XMR (Alice)} \\
&0.3 \text{ XMR (Bob) (mix-in=2)} \longrightarrow 0.3 \text{ XMR (Alice)} \\
&0.02 \text{ XMR (Bob) (mix-in=2)} \longrightarrow 0.02 \text{ XMR (Alice)} \\
&0.00111111 \text{ XMR (Bob) (mix-in=0)} \longrightarrow 0.00111111 \text{ XMR (Alice)}
\end{aligned} \tag{2.1}$$

Some privacy issues with this transaction may have become obvious already; since CryptoNote does not enforce transaction outputs to adhere to a strict form of $A \times 10^B$ for an integer $1 \leqslant A \leqslant 9$, we have a so-called dust output of 0.00111111 XMR, and this output cannot be mixed unless many other dust outputs happen to have such a strange denomination. Despite the privacy allegedly afforded by ring signatures with all other outputs, Bob has effectively stripped himself of privacy by including the dust input that is unmixable. Indeed, any observer can see this strange, unmixed output bundled with this transaction, and included in the same block; the observer will then draw obvious conclusions about funds ownership.

To worsen matters, Alice now also has a dust input that can not be spent using ring signatures without compounding the previous problem. While no observer can yet determine who received the transaction due to the uconditional unlinkability of the CryptoNote protocol, Alice will remove privacy from her transactions if she later decides to use that dust as an input due to the aforementioned problems with untraceability.

3 More Issues with Traceability

In this section, we identify other security concerns related to traceability in CryptoNote. This is not intended to be a comprehensive list of all routes of attack toward violating untraceability in CryptoNote, but represents a list of the most pressing issues identified by the Monero Core Team and the Monero Research Lab as routes of attack that may erode untraceability in the event of blockchain analysis.

3.1 Temporal Associations

This section describes how the age of a transaction output causes a circumstantial traceability among transactions. There are at least two distinct problems with the age of transactions. As before, we start with chain reactions. In expectation, the older outputs have been used in more ring signatures than younger outputs; they

have simply been extant in the transaction output set for longer, and have had more opportunities to be chosen from ring signatures. Any ring signature generation scheme that ignores this will provide a disproportionate amount of chain reaction potential to the over-used transaction output. Further, the earlier an attacker gains control of a transaction output set, the greater their capability to execute a chain reaction attack.

The second problem we identify is related to traceability, but not to chain reactions. Each transaction spends a transaction output using a ring signature, which uses an associated list of transaction outputs as mix-ins. The genuine transaction output being spent is a member of this list. If n foreign transaction outputs have been used in this ring signature, then this list has $n+1$ elements on it. An observer cannot distinguish which transaction output in the list associated with the ring signature is the output actually being spent without some form of blockchain analysis. However, in practice, given a certain transaction output, an attacker may model the cumulative probability that the output has already been spent as an increasing function of time. An output that has been in the blockchain for 202612 blocks is much more likely to have already been spent than an output that has been in the blockchain for 202 blocks. Hence, in the absence of any other external information, with any given output, the youngest output used to fashion the ring signature is the most likely output to be the genuine output being spent. Conversely, the oldest output is the one least likely to be the genuine output being spent.

In this paper, we do not directly address how to mitigate these observations for a variety of reasons. If it were possible to determine whether outputs had been spent, we would simply choose outputs that had remained unspent. However, this is not a simply tractable problem without blockchain analysis. One solution to this problem is to determine a non-uniform method of choosing transaction outputs for ring signatures; choose transaction outputs based on their age such that the probability that these outputs are chosen for a ring signature is inversely related to the probability that they have been spent already. This would suggest that we, the developers of Monero, must estimate the probability distribution governing the age of transaction outputs.

This, too, is problematic. When an exchange rate is experiencing a strong long-term decline (inflation), rational users are more likely to spend their transaction outputs, for tomorrow their outputs will be worth less in terms of goods and services than today and hoarding is not economically rational. When an exchange rate is experiencing a strong long-term increase (deflation), rational users are more likely to hoard their transaction outputs for the opposite reason. Hence, the distribution of transaction output ages will at least vary over time, and, presuming *any* proportion of users are rational will certainly depend sensitively on the economic performance of the currency. It is unwise to design security recommendations around the economic performance of our protocol.

An astute observer may recommend mining Bitcoin data, which is transparent, and developing a marginal lifespan distribution from that information. However, in this instance, we are making several problematic assumptions. Assuming that the Bitcoin economy is equivalent to the Monero economy, or even a decent proxy, is problematic due to the comparative sizes of the economies. Further, choosing any

one particular distribution that does not vary over time will allow an attacker to exploit the difference between the fixed distribution of choice and the time-varying distribution in the wallet software. One would be inclined to choose distributions that vary over time, perhaps with some iterative updating method, but this requires some form of observing data and responding to those observations; as we've previously mentioned, it's not easily tractable to observe the lifespan behavior within a CryptoNote protocol.

Hence, choosing a fixed distribution in order to generate random numbers to pick outputs used in a ring signature is a suboptimal solution. Another alternative to choosing a fixed distribution is to probabilistically choose from a family of distributions; to put it simply, we could roll dice twice, first to pick a distribution, and then second to pick outputs using that previously chosen distribution for the ring signature. However, these mixed distribution approaches are, in the end, equivalent to choosing a single fixed distribution and the problems with this approach reduce to the problems in the first, fixed distribution scenario.

Paradoxically, keeping wallet software intact without modification is equivalent to choosing the uniform distribution for each ring signature! Which is to say, we are stuck between a rock and a hard place with transaction output lifespan distributions, for any choice we make is certain to be a sub-optimal decision. The Monero Core Team and the Monero Research Lab would like to follow the development philosophy that it is wise to start with smaller changes at first and then ramp those changes up over time, rather than start with drastic changes and try to scale them back. Hence, although we have identified this security issue, we are not making formal recommendations yet until we have further data to inform our choices.

3.2 Association by Use of Outputs Within a Transaction

In this Section, we describe how careless usage of outputs together can be used to link those transactions as belonging to a common user. In other words, transaction outputs that are used together are probably owned together. Using the principle from Section 3.3, we see that this is a form of the combinatorial attack. Although the *ring signature* affords observers equiprobability across all possible signers, data exterior to the ring signature may allow for extra information.

Consider the following generalized example. If Bob sends Alice some Monero, it is broken up into several transaction outputs as in Example 2.1. When Alice goes on to spend that money, she may use two or more outputs from that same initial transaction in her ring signature. If so, an observer may conclude that the outputs stemming from a common root transaction are more likely to belong to the true signer of the transaction. This can lead to a probabilistic issue with traceability. This may be hard to visualize, so here is a specific example beginning with 1 XMR in Bob's wallet.

Bob wishes to send 0.75 XMR to Alice, and will pay 0.01 XMR in fees. The Monero that Alice receives, 0.75 XMR, will be delivered as two new unspent transaction outputs with amounts 0.7 XMR and 0.05 XMR. Further, an output of 0.01 XMR must be delivered as a fee. This leaves 0.24 XMR as change, which will be delivered to Bob in two unspent transaction outputs of amount 0.2 and 0.04 XMR each. At some point in time later, Alice realizes she has 0.75 XMR and decides to go

spend it some place. When she does so, both of her outputs, 0.7 XMR and 0.05 XMR, are included in her ring signature. An observer could then look at her ring signature and draw a conclusion that whoever signed that ring signature probably is the owner of both the 0.7 XMR and the 0.05 XMR. Even if Alice used a ring signature with many mix-ins, since she used two outputs that originated from the same transaction, those outputs are more likely to be identifiable as owned by the true signer. The transaction could look something like Table 1.

Table 1: Linking transaction outputs by association

(a) Transaction 1 by Bob

In	Out
1.0 XMR (Bob)	0.7 XMR (Alice) 0.05 XMR (Alice) 0.2 XMR (Bob) 0.04 XMR (Bob) 0.01 XMR (Fee)

(b) Transaction 2 by Bob

In	Out
0.7 XMR (Alice)	⋯
0.05 XMR (Alice)	⋯

Notice a few things about this example. First, this example is almost identical to the combinatorial attack using block height as will be described in Section 3.3. However, in this example, these outputs do not just share block height, but in fact they share a common root transaction. Second, this problem is symmetric with respect to Bob and Alice. Bob encounters a hazard also when he tries to spend his change output of 0.2 or 0.04 in a transaction. His ring signatures specify his originating transaction, as well, so if he decides to spend both his 0.2 and 0.04 XMR in the same transaction, they can be linked to him.

3.3 Combinatorial Attacks to Reveal Outputs

In this section, we generalize the attack described in Section 3.2 to describe a so-called combinatorial attack, during which a combinatorial analysis can be executed in order to draw a conclusion about output ownership. The primary conclusion of this section is simply that transactions should be broken up into several smaller transactions across periods of time as in a torrent, and users should re-send themselves their funds periodically.

Any data available to an observer about the transactions used to fashion a ring signature can be used in a combinatorial analysis to draw a conclusion about funds ownership. Consider the following example using block height. Lets say Cynthia received 10, 20, and 30 XMR at block height $39,859$ in separate transactions. In a much later block, an observer, Eve, sees three transactions of 10, 20, and 30 XMR, respectively. Eve sees that the 10 XMR transaction uses a mix-in from block $39,859$, and so does the 20 XMR transaction, and so does the 30 XMR transaction. By checking all combinations available, Eve concludes that it is more likely that whoever received 60 XMR at block height $39,859$ is spending those funds than any other user, although Cynthia and Eve do not necessarily know each others' identities.

This information can be used in tandem with other routes of attack. To be specific, *if a set of distinct transactions has some common data* (such as a common block height) *and if all transactions in this set use mix-ins with some other common data,*

then it is likely that these transactions are related by a common user. This provides some information to attackers about traceability. Our definition of a combinatorial attack seems strikingly general, and, in fact, includes the association-by-use attack described in Section 3.2. A combinatorial analysis of a set of transactions with some common data will lead to very few conclusions using this principle. That is to say, although this sort of *combinatorial attack* may not work most of the time, when it does work, it works very well.

4 Our Recommendations

In this Section, we specify our recommendations to the Monero Core Team for developing the Monero cryptocurrency. Our recommendations, while not perfect solutions to each of the above problems, mitigate the exposure users experience to traceability attacks. We recommend any and all CryptoNote developers to implement at least the network-wide minimum mix-in requirement of $n \geqslant 3$, and the larger the better.

4.1 Requiring Minimum Mix-Ins

In this section, we attempt to mitigate the chain reaction phenomenon by recommending a change to the Monero protocol *requiring* a minimum $n \geqslant 2$ mix-ins per transaction whenever at least 2 foreign outputs are available for mixing, and setting the default in wallet software to $n \geqslant 4$ mix-ins per transaction. We also justify this recommendation.

Specifically, we recommend that the Monero protocol be modified such that a node refuses to relay a transaction containing a ring signature without the requisite number of mix-ins, unless that transaction is denominated such that it lies in a transaction output set without a sufficient number of transaction outputs. We also recommend that the Monero protocol be modified such that a block is rejected if it contains any transactions containing ring signatures without the requisite number of mix-ins.

Of course, any mandatory minimum mix-in $n \geqslant 2$ allows for a probabilistic guarantee that any traceability chain reaction will terminate. Specifically, an attacker will have to spend more funds than they will reveal, as we demonstrated in [3]. The only question is "what value of n should be enforced?" This has been a source of heated debate within the Monero Research Lab and the Monero Core Team. Large values of n will lead to greater security at the expense of an unreasonably large blockchain and greater transaction fees. As previously mentioned, we are not using NIZK techniques and no ring signature method can match that level of security, and so any blockchain size exceeding the size of an NIZK blockchain (e.g. Zerocoin) is unacceptable. On the other hand, smaller values of n may save blockchain space, but could possibly compromise security. Luckily, the size, in terms of bytes, of a ring signature grows approximately linearly with the number of foreign keys used in the ring signature, and increasing n uniformly improves system resilience to chain reaction attacks as described in [3]. This resilience improvement is not linear, however, and there is no optimal point. In terms of security, $n = 3$ is superior to $n = 2$, and $n = 4$ is superior to $n = 3$, and so on.

Using Monte Carlo simulations of chain reactions at [5], we have concluded that individual users can be protected against chain reaction attacks by using at least $n = 3$ mix-ins for their ring signatures. Quantifiable security metrics can verify these results in at least two ways. The first metric we used to verify these results demanded that the success of a chain reaction attack depend on the output set size (yielding a choice of $n \geqslant 3$). Indeed, using this first metric, we see that, regardless of the size of an output set, a 50% attacker can be 95% confident of controlling one in ten new transactions when $n = 2$, but when $n \geqslant 3$, the attacker's success depends sensitively on output set size. The second metric we used demanded that an attacker controlling 50% of an output set of arbitrary size must wait for tens of thousands of transactions before they may be 95% confident of gaining control over a single new transaction (yielding a choice of at least $n \geqslant 4$).

However, it is still true that, for $n = 2$, chain reactions are still probabilistically guaranteed to burn themselves out, ensuring that degradation of system integrity is somewhat graceful and that any chain reaction attacker must spend more capital than they reveal. After much debate, we have decided that setting the default mix-in value within wallet software to $n = 4$ but allowing users who are less concerned about anonymity and more concerned about transaction fees to lower their mix-ins to $n = 2$ seems to be prudent. As previously mentioned, the Monero Core Team and the Monero Research Lab would like to follow the development philosophy that it is wise to start with smaller changes at first and then ramp those changes up over time, rather than start with drastic changes and try to scale them back. We are also formally recommending that the protocol be modified such that, after the blockchain has gotten sufficiently long, the network-wide minimum mix-in policy be increased to $n = 4$. After 2 years, Monero will automatically become more secure.

As an aside, notice that this is not the only way to address the chain reaction problem. An alternative way to address the chain reaction problem induced by allowing transactions to be spent in the clear is a tag-based solution. This solution recommends tagging some transaction outputs as safe to mix, and some as unsafe to mix; equivalently, this would involve pruning mix-in sets. Before validating a block, a miner could verify that no ring signature in the block uses a zero mix-in transaction output as a signature input by checking tags.

However, such tag-based solutions have some problems associated with them. For example, while a single block may not contain any ring signatures which use zero mix-in transaction outputs as signature inputs, due to chain reactions as described in [3], the block may still contain traceable transactions. Indeed, the transactions present in a block may be separated from dangerous "in-the-clear" transactions by several other transactions. Further, in a tag-based solution that still allows for zero mix-in transactions, users may still opt-out of benefiting from the privacy that ring signatures provide. It would be preferable to find a solution that maintains privacy as a uniform policy across the network, allows users to opt-in to greater privacy, and allows for compliance with local laws as described in Section 6.

4.2 Mitigating Combination Attacks

The easiest way to obfuscate ownership of the funds from eavesdropping by Eve during a combinatorial attack would be for Bob to simply send outputs owned by

himself to himself separately every few random periods of time. This skews the blockchain analysis performed by Eve in Section 3.3 and, in fact, in Section 3.1. In this section, we merely specify that no user resend all of her outputs to herself at the same time. Furthermore, any receiver of funds, by contrast, should request that the sender break the transaction up into pieces in a torrent and send the required amount over a period of time . This way, if Eve is anticipating a certain transaction amount within a certain window of time, she can not readily ascertain if some combination of outputs from all transactions in a given block might correspond to the exact amount which she expects the recipient to be sent.

By re-sending transactions to oneself iteratively over intervals of time with random length, and by breaking all transactions (including a resend transaction) into multiple smaller transactions, also sent over an interval of time, we dramatically weaken the ability for an eavesdropper to glean information from the blockchain based solely on block height. Notice that this recommendation is a wallet-level recommendation, not a protocol-level recommendation.

4.3 Mitigating Association by Use

To prevent a so-called "Assocation by Use" attack, an alternative transaction generation scheme could be used. Rather than generating a single transaction with as few unspent transaction outputs as possible, we can generate a separate transaction for each output. Hence, if Bob wants to send Alice 0.75 XMR out of a wallet with 1 XMR, then he will actually send two transactions. One transaction will send 0.7 XMR and one transaction will send 0.05 XMR. First, the usual denomination and change-splitting method will be applied before sending 0.7 XMR out of a wallet with 1 XMR within, as in Table 2. Then, with his change, Bob will complete the transaction; after his first transaction, this will leave Bob with a spare 0.09 XMR output, from which he can execute the usual denomination and change-splitting method before sending 0.05 XMR to Alice as in Table 3.

Table 2: Transaction 1

(a) Transaction 1A by Bob

In	Out
1.0 XMR (Bob)	0.7 XMR (Alice) 0.2 XMR (Bob) 0.09 XMR (Bob) 0.01 XMR (Fee)

(b) Transaction 1B by Alice

In	Out
0.7 XMR (Alice)	$\cdots$

Table 3: Transaction 2

(a) Transaction 2A by Bob

In	Out
0.09 XMR (Bob)	0.05 XMR (Alice) 0.03 XMR (Bob) 0.01 XMR (Fee)

(b) Transaction 2B by Alice

In	Out
0.05 XMR (Alice)	$\cdots$

In the above scheme, Bob generates only generates one extra output, but in doing so is able to protect Alice from being uncovered in the future by transactional

association of outputs used in ring signatures. Mining fees are increased according to the number of transaction outputs received, but anonymity is preserved.

Notice, however, that there is a subtlety under this alternative scheme. If Bob uses his own change output in order to send the 0.05 XMR in Transaction $2A$, this can lead to the same circumstantial likelihood analysis that we were trying to avoid in the first place. Outputs may be associated as being spent from a common transaction, or from a common *tree* of transactions, or spent from a common block, or spent from within a short span of time. This list may not be comprehensive. Of course, these scenarios are harder and harder for Eve to analyze compared to the simple scenario initially presented in Section 3.2.

Two outputs from disjoint transactions with two different block heights are less likely to come from a common owner than two outputs from a common transaction, or than two outputs from two transactions linked by a single step, or than two outputs from a similar block height. Hence, since Bob can also use any other output he owns in order to send the remaining 0.05 XMR, not just his change output, he should endeavor to do so if possible. This leads us to the natural solution, which is simple to implement in its basic form: use one and only one input for every transaction, and output one and only one output to the recipient. This is repeated until all desired funds are transferred to the recipient.

This is, of course, an inefficient method of generating transaction outputs. To increase efficiency, multiple inputs may also be used to generate a transaction if and only if the outputs referenced do not lie within a few blocks of each other.

5 Roadmap to our Recommendations

Implementing a network-wide minimum mix-in policy $n = 2$ to the Monero architecture can be done in several ways.

(1) Within a wallet, users may be required to send via wallet-to-wallet RPC with a minimum mix-in, unless those transactions lie in transaction output sets without enough mix-ins.

(2) Miners may be requested or required to only include mixed transactions satisfying the minimum mix-in, unless those transactions lie in transaction output sets without enough mix-ins.

(3) The peer-to-peer protocol may be modified such that transactions that do not satisfy the minimum mix-in for all their ring signatures are not relayed, unless those transactions lie in transaction output sets without enough mix-ins.

Furthermore, if a transaction is included in a block with height greater than a certain critical value, $n = 2$ shall not be sufficient, but in fact we will require $n = 4$. We recommend this block height to correspond to about two to three years after publication of this document, but the specifics are unimportant. Our roadmap to implement a network-wide minimum mix-in policy is to implement (1) soon, skip (2), and plan (3) for a later date after vigorous testing on our testnet. The primary obstacle to immediate implementation is transaction *dust*. To overcome the obstacle, we define a *legal transaction output form* as any number $A \times 10^B$ where A is an integer $1 \leqslant A \leqslant 9$ and B is any integer $-12 \leqslant B$. Hence, a transaction output with amount 0.00111111 XMR does not follow the legal transaction output form, but the transaction outputs $0.001, 0.0001, 0.00001, 0.000001, 0.0000001$, and 0.00000001 XMR are each legal. We propose the following policy:

(i) Within a wallet, users may be required to produce only transaction outputs adhering to the legal transaction output form.
(ii) Miners may be requested or required to only include transactions with all outputs satisfying the legal transaction output form.
(iii) The peer-to-peer protocol may be modified such that transactions that do not satisfy the legal transaction output form for all of their transaction outputs are not relayed.

Implementing these will allow a user to use a non-legal transaction output as a transaction input, so long as all outputs from that transaction adhere to this strict form. No new dust will be created in the system, and dust will only be removed from the system over time. Again, our roadmap to implement a legal transaction output form is to attack (*i*) first, skip (*ii*), and plan (*iii*) for a later date after vigorous testing.

In order to encourage users to re-send transactions to themselves iteratively over intervals of time with random length, we recommend that, within a wallet, a warning be displayed if a random but minimum number of blocks have been added to the blockchain since an output has been received in the wallet. After 6 to 8 weeks have transpired, for example, all outputs in a wallet should be re-sent.

Furthermore, wallet software should specify that
(a) one and only one transaction output should be sent to a recipient at a time;
(b) a transaction that must consist of several outputs should be broken up into several smaller transactions spread over time as in a torrent; and
(c) each transaction output received by a recipient should use transaction inputs with dissimilar block heights and dissimilar root transactions.

6 Auditability for Compliance Purposes

Businesses using ring signatures and stealth addressing through Monero will have the reasonable desire to comply with the corresponding local laws within their jurisdictions. Typically, these laws require the business to prove ownership of their own funds, and demonstrate where funds may have been sent. To prove ownership of any unspent funds on the block chain, a Monero account holder may publish a zero mix-in ring signature of their public key. Please be aware that a zero-mix-in ring signature is just an ordinary digital signature. An auditor can then scan the block chain for the presence of the output public key, and verify that the key image produced from the zero mix-in ring signature has not yet been used on the block chain. To protect the privacy of the business and its customers, the auditor should then destroy the signature and the key image.

7 Conclusion

We identified several routes an attacker could use to execute a blockchain analysis to degrade the untraceability of the CryptoNote 2.0 protocol. The zero mix spending problem identified in [3], the age-based association problem, the association-by-usage problem, and a more generalized combinatorial analysis problem are each identified. We recommend a network-wide minimum mix-in of $n = 2$ to address zero mix chain reaction problems, and we discuss how a mix-in of $n = 4$ is likely

to be larger than necessary for most practical purposes. We recommend a non-uniform transaction output selection method for ring generation to mitigate age-based association problems. We recommend partitioning all transactions into many smaller sub-transactions such that they occur over an interval of time, as in a torrent, and such that each sub-transaction has one and only one output, in order to mitigate the combinatorial analysis problem.

References

1. Norman Lloyd Johnson and Samuel Kotz. *Urn models and their application: an approach to modern discrete probability theory*. Wiley New York, 1977.
2. Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.
3. Surae Noether, Sarang Noether, and Adam Mackenzie. Mrl-0001: A note on chain reactions in traceability in cryptonote 2.0. Technical report.
4. Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology-ASIACRYPT 2001*, pages 552–565. Springer, 2001.
5. Tacotime. Saturation of unspent transaction outputs, September 2014.