Xim: Distributed Match-and-Mix for Bitcoin

George D. Bissias A. Pinar Ozisik Brian N. Levine Marc D. Liberatore
School of Computer Science, Univ. of Massachusetts Amherst
{gbiss, pinar, brian, liberato}@cs.umass.edu

ABSTRACT

A fundamental limitation of Bitcoin and its variants is that they link pseudonymous public addresses with transactions. The movement of coin between addresses can be observed by examining the public block chain that records the history of transactions. This record enables adversaries to link addresses to individuals, and to identify multiple addresses as belonging to a single participant. Users can try to hide this information by mixing, where a participant exchanges the money in an address coin-for-coin with another participant and address. In this paper, we describe the weaknesses of extant mixing protocols, and analyze their vulnerability to Sybil, denial-of-service, and timing-based inference attacks. As a solution, we propose Xim*: a complete decentralized mixing protocol that aims to minimize the losses of an honest participant if they interact with an adversary. We bootstrap a protocol by Barber et al. with a new discovery protocol for anonymously pairing up participants. We explain the relationship between parameters used in our protocol for initial setup, and show why our protocol is effective in deterring denial-of-service, inference, and Sybil attacks.*

1. INTRODUCTION

A growing number of people have turned to decentralized *virtual currencies* (VCs) such as Bitcoin [22] or Litecoin [17] for convenience, speculation, or as a potential source of financial privacy and security. For example, people generally lack control over the efforts of credit agencies and marketing firms to mine information from financial transactions. Additionally, merchants that accept credit and debit cards have a history of security failures, such as the recent theft of credit-card data from Target [14,24].

The benefits of virtual currencies are many: low transaction fees, transactions over the Internet, and potentially, convenience and privacy. Unfortunately, VCs are not a complete solution to the problem of economic privacy. Imagine purchasing coffee from a café and revealing the balance of your bank account at the register. This scenario is the status quo when using Bitcoin and related VCs because they use a public transaction log (called a *block chain*) to prevent counterfeiting. As a result, user wallets (called *addresses*) are pseudonymous rather than anonymous, which means that while they are not explicitly tied to an individual, there is no mechanism to explicitly disguise the address's owner. Although it is broadly recognized [6,19,25,25] that Bitcoin and similar VCs are not private as-is, there are to date no robust decentralized services for providing privacy that break the links among users, addresses, and transactions.

Currently, there are three mechanisms for *mixing*, which is the process of transferring funds between two address without recording their relationship to the public block chain.

First, there exist several *centralized* mixing services. Examples include SharedCoin [5] and DarkWallet [3]. Such centralized mixing services are analogous to providing Web browsing anonymity with a single anonymizing proxy, rather than using a more robust solution such as Tor [10]. The central mixing agent must be completely trusted as it knows which users exchange funds with others.

Second, new virtual currencies have been designed with explicit support for mixing, wherein users create new addresses and move coins among them to explicitly *unlink* the addresses and past transactions. For example, Miers et al. [20] has proposed a protocol called *Zerocoin* with fully anonymous transactions based on zero-knowledge proofs. Similarly, Ladd [16] has proposed a scheme based on blind signatures. The resulting schemes are incompatible with existing VCs, of which there are many and for which the most popular comprise is an enormous amount of capital[1]. These improvements to future VCs will be useful, but we assert that a privacy solution for existing VCs is of critical importance.

Third, peers can rely on one another for mixing without a centralized or third party. For example, Barber et al. [7], Ruffing et al.'s CoinShuffle [2], CoinJoin [1], and others have proposed methods for two parties to directly mix their coin. A two-party approach removes centralized trust and is compatible with existing Bitcoin-like currencies. Unfortunately, these proposed schemes are limited in fundamental ways: none provide a complete, decentralized protocol that pairs specific partners while avoiding Sybils [11]; all are cost-free to participate, and therefore are trivially subject to denial of service attacks; none analyze performance over multiple rounds of mixing; and some leave information on the block chain about which peers paired to exchange coin, a significant vulnerability.

Contributions. We propose **Xim**: the first complete solution for two-party mixing that is compatible with Bitcoin and related VCs. Our protocol includes a decentralized solution for anonymously finding partners and is designed to be a multi-round system. It is the first protocol to *simultaneously* address Sybil attackers, denial-of-service attacks, and timing-based inference attacks. We build upon and improve a solution by Barber et al. [7] for unlinkable fair exchange. Our contributions are as follows.

[1]Bitcoin alone had a market capitalization of about $5.6b on 2014-05-14.

- Xim includes a decentralized system for anonymous partnering. It requires peers to spend funds both to participate and respond to ads for mixing partners; the funds are given to the miners. *Mix partners are assured the other has paid, but no outside party can confirm or find evidence of the agreement.*
- Xim thwarts denial-of-service attacks in two ways. First, neither partner can abort without losing a non-zero amount of coins, which is not a feature of previous solutions. Second, all interactions in Xim are between only pairs of peers; in some past solutions, one peer can disrupt many other peers with one stroke.
- We show that because of Xim's participation fees, *the cost to a Sybil attacker wishing to maintain a fixed success rate grows linearly with the number of mix participants, while honest participants' costs are small, fixed, and constant with the number of participants.* We also quantify Xim's tunable success rates. For example, even when a Sybil attacker controls 1/3 of the population of mixing peers, our protocol can be set to mix with 99% success rate, with low cost and with a delay that does not increase with the amount of coin mixed.
- Our protocol thwarts timing attacks by coordinating the actions of participants. Participants agree upon the number of rounds they seek to mix and when to start mixing.
- We empirically analyze the delay performance of our protocol if used on Bitcoin. We measured the commit delays of transactions on Bitcoin for 21 days. Our measurements show that Bitcoin transaction delays fit well to a heavy-tailed, log-normal distribution, which has implications beyond the operation of Xim.

We begin by analyzing previous work on mixing, including Barber et al., CoinJoin and CoinShuffle. We show that with the exclusion of Barber et al. (which we build upon), previous work is subject to timing attacks and trivial DoS and low- or no-cost Sybil attacks.

2. PROBLEM DEFINITION AND RELATED WORK

Alice controls a virtual currency address A, which is known to be under her control. She would like to move the coin from A to a new address A', such that no one knows that A' or the coin in it are under her control. In other words, she wants addresses A and A' to be *unlinkable*. Intuitively, this condition holds if there is no sequence of transactions on the block chain that can link them to the same owner.

2.1 Decentralized Mixing

Alice can achieve this unlinkable transfer of funds trivially with a trusted partner Bob. *Mixing* schemes allow Alice to trade coin-for-coin with Bob, between his addresses B and B'. Alice agrees to transfer coin from address A into address B'. Likewise, Bob transfers the same amount of coin from his address B into A'. Now each pair (A, A') and (B, B') is unlinkable by a third party because there is no record of the partnership between Alice and Bob's address on the public block chain.

This approach has a problem: Alice may not trust Bob. Alice cannot transfer a large amount of coin into B' since Bob may not reciprocate. A solution to this problem is use of a trusted third party, in this case, a centralized mix to *(i)* pair participants and *(ii)* oversee and enforce the exchange and unlinking process.

The use of a trusted third party has well-known problems of its own: Not only must Alice and Bob trust that a centralized mix, Trent, will not steal their coins, but they must also trust that he will not help others — or be attacked by others — that want to link Alice and Bob's old and new addresses. Even if Trent has integrity, it's possible that others might launch a denial-of-service attack on Trent in order to keep him from mixing at all. A better solution is remove Trent while still allowing the mutually untrusting Alice and Bob to find one another and to fairly and unlinkably exchange coin.

Thus, *our goal is to develop a decentralized protocol that allows Alice and Bob to discover, partner, and unlinkably but fairly exchange coin* with one another. Our solution, Xim, is a protocol that functions correctly in the presence of malicious mixing partners, forces Sybil attackers [11] to pay costs linear to the number of participants in the mix, and is more robust against denial-of-service attacks than previous approaches. Xim also attempts to hide the IP address of the parties involved in mixing [15]. Our goal is not to defeat all possible inference attacks — attacks leveraging outside information to link many different addresses to the same owner — but we do show how Xim addresses both intersection and timing attacks.

2.2 Assumptions and Attacker Model

Our protocol is designed to work with Bitcoin, but should work with any VC that provides a similar set of features. In particular, we assume a VC with a public block chain and addresses controlled by public-private key pairs. We assume all participants, including attackers, can create new, empty addresses at will, without cost, and without being linked to the old addresses. That is, participants and addresses are not one-to-one. We assume transactions are driven by a scripting language that supports multi-signature addresses and that allows small arbitrary values to be included as part of transactions.

We consider several types of attackers that differ in their objectives. Each operates within the confines of the VC and mixing protocols, but we assume that the cryptographic mechanisms that support the currency are not a source of vulnerability. *(i)* A de-anonymizer uses a sequence of transactions to link activity from multiple addresses that are controlled by a single user. We assume that the information the attacker seeks has value to them. We assume he is willing to pay to obtain the information but, rationally, not more money than the information is worth. *(ii)* A defrauder seeks to profit from flaws in mixing protocol. We assume she is rational, and thus willing to risk money on a transaction if she can expect to profit on average. *(iii)* A destroyer seeks to cause honest participants to lose money, even though the attacker himself won't profit from the loss and may even spend money to cause the loss. This type of attacker is rational if there is an additional external or intrinsic value to the attacker from causing the other party to lose coins.

2.3 Related Work

There exist various protocols for mixing Bitcoin and related virtual currencies. We distinguish each protocol or service according to two primary mechanisms: how they find peers to partner with and how they fairly exchange funds.

- **Partner Selection:** Peers exchanging funds must find others and be partnered for later mixing. Some protocols

use a centralized third party to select mix partners. Others use a public bulletin board and self organize. For Xim, we propose an anonymous peer-based protocol.
- **Fair exchange:** Once partnered, peers must fairly exchange funds. Some protocols use a trusted third party, and other protocols make use of CoinJoin [1]. For Xim, we use Barber et al. [7] .

First, we summarize the operation of CoinJoin and Barber et al. Each is restricted to fair exchange of coin, and neither proposes a partnering protocol. To our knowledge, there are no detailed proposals for partner selection to review here: all use a trusted third party, except CoinShuffle, which we describe below. Second, we enumerate complete protocols that include both partner selection and fair exchange. In Section 3, we identity security vulnerabilities in these past works.

2.3.1 Decentralized Fair Exchange

Because of the decentralized architecture of Bitcoin and related VCs, it's appealing to provide a mixing service without involving a third party, and thereby avoid many vulnerabilities. The miners collectively are a third party, though the Bitcoin protocol curtails their ability to misbehave, as invalid blocks will be rejected by other miners. Miners enable a fair exchange between two or more mixers; without a third party, fair exchange is impossible [13,23]. (A protocol is *fair* if no honest participant wins or loses anything valuable if the other party does not behave according to the protocol [23].)

CoinJoin A popular exchange technique known as CoinJoin [1] was described informally in a Bitcoin related forum and has subsequently been deployed by commercial mixing services [3,5]. CoinJoin is appealing because of its simplicity. The protocol begins by identifying n participants, each with an existing input address that holds at least δ Bitcoin and a newly created output address. A single *swapping* transaction is published, which moves δ Bitcoin from each input address to one of the output addresses. Because multiple input addresses map to multiple output addresses, this transaction ensures that the n output addresses are indistinct by inspection of the block chain. Therefore, it's impossible to ascertain who controls the δ Bitcoin in any of the output addresses, and so they are unlinked from the input addresses. There are multiple weaknesses in this approach, as we detail in Section 3. CoinJoin itself is not a complete mixing protocol because it neither describes how participants should be selected, nor how the swapping transaction is actually formed. Several services build upon CoinJoin to implement complete mixes.

Barber's Fair Exchange. Barber et al. [7] propose a protocol that both unlinks exactly two addresses, and cryptographically enforces a fair exchange between them. The description of the protocol below is a simplification; Fig. 3 in the appendix outlines Barber et al.'s protocol in further detail. Although the protocol requires four transactions total (versus CoinJoin's single transaction), and does have limitations as we show in Section 3, Barber et al. provides a significantly stronger basis for Xim.

The protocol is initialized by a cut-and-choose exchange of values resulting in Bob knowing values x and y, and Alice knowing x, y', and z, such that (with arbitrarily high probability) $y' = H(y)$ and $z = H(x + y)$. They then craft two transactions containing contracts that are then broadcast to the miners. Specifically, a transaction TxCommB is created with a contract that allows only two possible results: Bob's coin is transferred to an address Alice controls, or it is sent back to himself as a refund after a timeout. A second transaction TxCommA commits Alice's coin to the analogous contract: her coin is transfered to an address Bob controls, or it is sent back to herself after a timeout. Further, TxCommA includes the value $z = H(x + y)$ and prevents the transfer of funds to Bob unless the follow-on transaction reveals z. TxCommB similarly includes the value $y' = H(y)$ and requires a follow-on transaction that reveals y. In the third transaction, Bob claims the funds committed by TxCommA, revealing $x + y$ to Alice. Because she already knows the value of x, she computes y and claims the funds committed by TxCommB. If one party aborts, the transactions are constructed such that the honest party can recover their own funds. The protocol's recovery makes use of Bitcoin's mechanisms for ensuring that transactions are not valid until a specific date in the future (see `https://en.bitcoin.it/wiki/Contracts`).

2.3.2 Complete Pairing and Mixing Protocols

Centralized Pairing or Mixing. We classify any protocol as centralized if it uses a trusted third party for partner discovery or centralized fair exchange. Whether pairing, mixing, or both, the central agent is a convenience, but it must be completely trusted as it knows which users exchange funds with others. As we discuss in Section 3, such centralized mixing services are analogous to providing Web browsing anonymity with a single anonymizing proxy rather than Tor.

MixCoin [8] allows centralized mixes to issue warranties, which users can redeem to damage the mix's reputation if it fails in its promises. The mix collects all-or-nothing fees according to a rate parameter ρ. Until recently Blockchain.info offered a centralized mixing service called Send Shared [4]. It has been replaced with SharedCoin [5], which uses a centralized server for both participant selection as well as CoinJoin transaction creation and publication. DarkWallet [3] operates similarly and imposes no fees on mixing participants (other than normal transaction fees).

Decentralized Pairing and Mixing. Only two protocols provide fully distributed pairing and mixing: Our protocol Xim, and CoinShuffle [2].

CoinShuffle is a variation on CoinJoin where $n \geq 2$ participants meet through a public bulletin board and agree to mix. Ownership of output addresses is disguised from participants by means of a cryptographically secure shuffling protocol in the spirit of communication mix networks. The swapping transaction is fashioned so that it must be signed by all n participants in order to be considered valid.

3. ANALYSIS OF EXISTING MIXES

In this section, our goal is to identify the vulnerabilities and limitations of existing protocols so as to justify our design of Xim, which is detailed in Section 4. A summary of our results appears in Figure 1.

3.1 Centralized Mixes

Mixes can be centralized by a third party, but using such an architecture imposes costs and risks. Third parties may charge large fees for the services they provide, and there is no guarantee that they won't conspire with one of the parties or steal from both [19,21]. Bonneau et al.'s [8] mix reputation

Protocol	Pairing Protocol	Exchange Protocol	Evidence of Pairing?	Sybil Attack Costs	DoS Attack Costs, Per Round
MixCoin [8]	Centralized with warranties	Centralized	On block chain	mixing fee rate ρ	N/A
SharedCoin [5]	Centralized	n-way CoinJoin	On block chain	tx fee / successful Sybil	None
DarkWallet [3]	Centralized	n-way CoinJoin	On block chain	tx fee / successful Sybil	None
CoinShuffle [2]	Centralized	n-way CoinJoin	On block chain	tx fee / successful Sybil	None
Xim	Anonymous Decentralized Pairing	Barber et al. [7]	None; timing attack required	tx fee / attempted Sybil	$\tau/2 > 0$ coin per aborted tx

Figure 1: In this comparison of various Bitcoin-compatible mixing protocols, only Xim simultaneously has all desired properties (distributed pairing, fair exchange, no globally-visible evidence, and per-Sybil and per-DoS fees).

system reacts to that possibility but is centralized nonetheless. If a centralized mix logs information about partnering, it could later be used, by the mix or another party, to reveal the information the mixing hid. If that revelation is undisclosed, the reputation system is ineffectual. As we show in Section 4, Xim does not suffer these vulnerabilities because its parter selection and exchange algorithms are both decentralized.

3.2 Vulnerabilities in Decentralized Pairing or Mixing

DoS Vulnerability. Protocols such as DarkWallet, SharedCoin, and CoinShuffle do not exact fees from their participants to join the mix pool, so they are all susceptible to a simple denial-of-service (DoS) attack. Suppose that an attacker would like to disrupt one of these mixes. He creates many different identities, each associated with a different IP address and Bitcoin address. The attacker inserts each of these identities into the pool of mix participants at no cost. Any time he is selected as a mix participant he refuses to sign the swapping transaction. If the attacker can create enough identities, it's possible for him to participate in the majority of transactions, which would significantly affect mix performance. Notice that the attacker loses no Bitcoin in any of the aborted swapping transactions.

In fact, precisely because CoinJoin and its derivatives (including CoinShuffle) perform the entire mix operation in a single transaction, it's not possible to force participants to pay a fee up front. Any fee would need to be collected earlier using a separate protocol, one that is unclear how to design, and has, to our knowledge, not been proposed. Similarly, Barber et al.'s fair exchange protocol does not include any fees for participating. Abandoning the exchange, resulting in denial-of-service for the victim, does not cost the attacker at all. As we show in Sections 4 and 4.5, Xim does not suffer these vulnerabilities because it charges a participation fee and aborting a round of mixing to cause a DoS surrenders the fee.

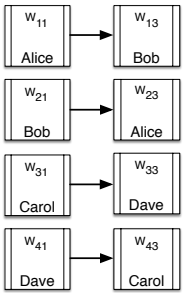
Intersection Attack. A practical consideration for any decentralized CoinJoin implementation is that the number of participants n in any single swapping transaction must remain relatively low. One reason is the exponentially increasing communication overhead, and another is increased susceptibility to the DoS attack outlined above. Additionally, there is a maximum transaction size.

On the other hand, when n is small, susceptibility to an intersection attack increases. The intersection attack is described in detail in Section 5.4, but we summarize its impact on CoinJoin here briefly. Suppose that a mix participant Alice tends to spend her mixed Bitcoin on a particular item like hatpins. At no cost, an attacker can easily identify CoinJoin's swapping transactions on the block chain, and therefore determine the set of n possible source addresses associated with the hatpin purchase. If Alice elects to mix over k rounds, then the attacker identifies a set of n^k possible source addresses. Over time, Alice might purchase more hatpins and she's always careful to mix her funds before completing the purchase. However, the attacker can observe that while many different source addresses are used over time, Alice's source address is always present. From this analysis, the attacker can surmise that Alice is the participant completing the purchase [9].

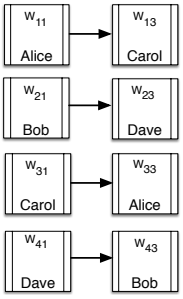
In CoinJoin (and CoinShuffle), the partnering peers are written to the public block chain, which makes the attack available *even to attackers that have not participated in mixing*; in analogy to anonymous communication systems, all attackers are global passive adversaries in these protocols. Xim does not allow for global passive adversaries because when using Barber et al., no evidence is written to the block chain; attackers must participate (and thus pay) to perform the intersection attack. Also, as argued in Section 5.4, the larger pool sizes used by Xim further lower the probability of attack success relative to CoinJoin implementations.

Sybil Attacks. One way to defeat any mixing protocol is to comprise a large fraction of the available participants. As the size of such a Sybil population increases, the chances increase that the parties selected to mix include the attacker. In a protocol that mixes pairwise, a Sybil attacker will know the destination of the funds in any address with which it is paired. DarkWallet, SharedCoin, and CoinShuffle are subject to a cost-free Sybil attack because none charge a fee to participants, and because addresses (i.e., Sybil identities) are free to create in Bitcoin and related VCs.

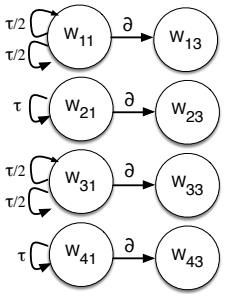
Participants in these protocols do pay standard transaction fees. These costs are practically negligible; but even if considered substantive, the cost to the attacker is only the sum of the instances where they are successful, rather than being proportional to the number of identities they created. As we detail in the next section, Xim does not suffer these vulnerabilities: Sybil attackers wishing to maintain a fixed success rate must pay for each participating address, and so their costs grow linearly with the total number of mix participants, while honest participants' costs are small, fixed, and constant with the number of participants.



Case I: A partners with B, C partners with D; This figure **does not depict the block chain**, rather who *controls* each address.



Case II: A partners with C, B partners with D; This figure **does not depict the block chain,** rather who *controls* each address.



Transaction graph for Xim for both case I and II.

Figure 2: Regardless of whether Xim executes case I or II above, the same transactions appear on the block chain *(c)* : an advertisement (two loops for the advertiser, one for the respondent) and a move to a new address. Even if an attacker knows who controls addresses $w_{11}, w_{21}, w_{31}, w_{41}$, it is of no help unless he is actually a mix partner.

4. XIM: ANONYMOUS PARTNERING AND MULTI-ROUND MIXING

In this section, we detail Xim, a decentralized protocol for anonymously finding partners and mixing with them. Its advantages are *(i)* it mixes *without leaving evidence on the block chain* of the exchanged control of funds; *(ii)* it works *without trusting a third party* to select or learn the pairing of peers; *(iii)* it *thwarts Sybil attacks* because the costs for attackers increases linearly with the number of participants, but remaining constant for honest participants; and *(iv) dishonest participants can never profit*, and honest participants never risk more than a small amount, much less than the amount they are mixing.

4.1 Overview

Xim is described at a high level as follows. Alice wishes to mix coin stored in address A. To begin, Alice commits a transaction that advertises her willingness to mix with a partner, tipping $\tau/2$ coin from A to the miners; she uses Tor to hide her IP address. She is contacted by several willing partners (also using Tor), and she chooses one, who then tips τ coin to the miners; no other peers can recognize their partnership. Finally, Alice confirms the partnership by releasing another $\tau/2$ coin to the miners.

Once Alice has a partner (call him Bob with address B), they swap funds: all funds in A are transferred to an address B' controlled by Bob; all funds in B are transferred to A', controlled by Alice. The transfer of funds is performed in a single logical step using Barber et al.'s protocol for fair exchange (See Section 2.3). Once this round of mixing is complete, Alice begins anew, advertising and finding a new partner. We expect that Alice has a total of $m\delta$ coin that require mixing. Alice will thus repeat the protocol m times, but this can be performed in parallel if desired.

Completing one round for mixing δ coin is sufficient for defeating attackers that aren't participating in mixing because no evidence remains on the block chain. Transactions are structured to include no indication of who is partnered for mixing (cf., CoinJoin). However, to also defeat malicious Xim participants, mixing with only one partner is not sufficient. Each participant will require a variable number of rounds n; we detail how Xim manages multiple rounds in Section 4.5.

Evidence on the block chain. Figure 2 illustrates two cases where four peers pair up to mix, each resulting in the same record on the block chain. In Case I, A partners with B, and C partners with D. In Case II, A partners with C, and B with D. The transactions that are recorded are an advertisement (two loops for the advertiser, one for the respondent), and a final move to a new address. Even if an attacker knows who controls addresses $w_{11}, w_{21}, w_{31}, w_{41}$, it is of no help unless he is actually a mix partner. As we show in Section 4.5, attacks from mix participants are defeated with arbitrarily high probability when multiple rounds of mixing are used.

Transaction conventions and notation. If Bob transfers δ coin from address B to address X, we denote the transaction as $\sigma_{K_{b-}}\{B \xrightarrow{\delta} X\}$, where σ is a signing operator, and its subscript indicates that the transaction contents were signed with private key K_{b-} controlled by Bob. Note that the transaction is not actually valid until Bob *publishes* it and it is *committed* by miners to the block chain. For example, transferring δ Bitcoin from address A to address B' requires Alice to sign. In that case, we write the transaction as $\sigma_{K_{a-}}\{A \xrightarrow{\delta} B'\}$.

Communication. When a plaintext message msg is encrypted, we denote it as $E_{K_{a+}}(msg)$, where K_{a+} is a public key controlled by Alice. Our protocols make use of a TEXT field that can be signed. In Bitcoin, the comment field is not included in data that gets signed. However, the scripting language that controls the validity of transactions can include the extra values we require with ease. Mixers communicate out-of-band by depositing messages anonymously in a public bulletin board. In sum, we require a service where anyone can post or read messages, and no one can delete messages posted by others; see Section 4.2.2. Accordingly, Alice and Bob can pass messages without revealing personal information such as IP address. In our protocol, we denote the *location* where she can receive and post messages as α_A and the location corresponding to Bob as α_B.

4.2 One Round of Mixing

Xim operates in two phases: discovery of a mix partner, and the actual mixing with that partner. Participants Alice and Bob mix each unit of δ coins over $n \geq 1$ rounds where Alice's

current round is given by ρ_A and Bob's current round is given by ρ_B. Peers complete n rounds total. During any given round, δ coins are transferred from Alice's origin address A to her destination address A' as follows and likewise for Bob.

PROTOCOL 1: Xim(ρ_A, ρ_B, δ)

1: Alice:	$\alpha_B \leftarrow$ **Discover**(ρ_A)
2: Bob:	$\alpha_A \leftarrow$ **Discover**(ρ_B)
3: Alice, Bob:	**Barber et al.**(A, A', B, B', δ)

Protocol **Discover** is responsible for pairing Alice (listening for messages at DHT location α_A) with a randomly chosen participant, Bob, who is listening at DHT location α_B. By design, it is difficult for an adversary to partner herself with Alice in every round. Alice's objective is to move coin (indirectly) from address A into a freshly created and unlinkable address A'. Protocol **Barber et al.** performs the actual unlinking operation between Alice and Bob.

We introduce additional details in Section 4.5 that allow Xim to avoid timing attacks within a round, and allow it to operate over multiple rounds.

4.2.1 Peer Participation and Discovery

Peer discovery is a fundamental part of a distributed mix protocol and has its own set of challenges. It's important to ensure that a participant can easily find the full set of other participants without risk of being deceived into selecting from a pool of unscrupulous ones. Second, a single attacker must do approximately the same work as an honest participant for each Sybil-based identity she creates. We leverage the global consistency of the block chain itself to achieve these ends. Protocol **Discover** details the solution. Alice randomly alternates between acting as an advertiser and a peer that responds to advertisements.

The advertising protocol is designed so that Advertiser ($\mathcal{A}$) and Respondent ($\mathcal{R}$) will each spend τ on an advertisement at its successful conclusion; we discuss how to set τ in Section 6. These recurring costs are a stronger proof of work against Sybil attacks than a flat, one-time fee [18]. Further, because they are charged for joining the pool of participants rather than for actual mixing, they ensure DoS attacks incur a non-zero cost. Although the ads are public, no one can prove that the two parties are linked. However, if $\mathcal{R}$ aborts before paying τ, $\mathcal{A}$ can reuse her advertisement without losing her investment. Conversely, if $\mathcal{A}$ aborts after $\mathcal{R}$ pays τ, then $\mathcal{R}$ can prove he committed to working with $\mathcal{A}$ who ultimately aborted. With his proof made public, others will ignore $\mathcal{A}$'s ad; $\mathcal{A}$ will have to advertise again at a loss of $\tau/2$ coin. Thus, $\mathcal{R}$ cannot cause $\mathcal{A}$ to lose any coin, and while $\mathcal{A}$ can cause $\mathcal{R}$ to lose τ coin, it will come at a cost of $\tau/2$.

- In Steps 1–5 of **Protocol 2 Discover**, a participant uniformly at random selects the role of advertiser or respondent; the choice is random to thwart inference attacks.
- In Step 6, as advertiser $\mathcal{A}$, a participant spends $\tau/2$ on an ad, and lists a location where others can leave messages for her (see Section 4.2.2). Messages left for her will be encrypted with the public key of the address she advertised with. The τ coin go to the miners; since no party can pre-select the winning miner, it's clear there can be no collusion. A nonce, N_a, uniquely identifies the ad, while optional parameter $\mathcal{P}$ gives the desired mix pool.
- In Step 7, some number of peers will each send her an encrypted message at the given location. The message contains the ad that they are responding to N_a, their own nonce N_r, and a location α_R at which they receive messages to set up the fair exchange. The three values are chosen at random for each instance of an ad or response.
- In Step 8, the advertiser selects one of the respondents $\mathcal{R}$, and she commits to his response by sending to location α_R a signed message including N_a and the hash of N_r. Other respondents will then seek other advertisers. Once $\mathcal{R}$ sees this commitment, he is encouraged to place his own response ad on the block chain in Step 9. His ad costs τ, and the included message, encrypted with $\mathcal{A}$'s public key, guarantees to her that the costs are in response to her ad only.
- If both parties are honest, then in Step 13, $\mathcal{A}$ will publish a response ad on the block chain costing $\tau/2$ coin that broadcasts she is pairing her ad with a partner obfuscated as $h(N_r)$. Since the nonce is included, $\mathcal{A}$'s response ad can satisfy exactly one respondent.
- The remaining steps serve for failure recovery. If $\mathcal{R}$'s ad (Step 9) does not appear in the block chain, then $\mathcal{A}$ can unpair and re-use her ad at no cost (Step 11). On the other hand, if it's $\mathcal{A}$ that does not publish a response, then $\mathcal{R}$ can prove $\mathcal{A}$ was dishonest by storing the following values to α_A for all to see: the id of $\mathcal{A}$'s ad N_a; the pairing message $h(N_r)$; and his knowledge of the true id N_r. Any third party can encrypt N_a and N_r, and match them to $\mathcal{R}$'s ad for verification. Thus, $\mathcal{R}$ can only make this claim if and only if he was $\mathcal{A}$'s respondent in Step 7 and he actually paid for $\mathcal{A}$'s ad.

It's important that peers take on both roles of advertising and responding between rounds. If Alice acts only as an advertiser, then an attacker could always request Alice's participation and make it very likely that she and Alice are partnered on every mixing round. If all peers act only as a respondent, then the protocol is not sustainable.

Fair exchange between the participants is handled by Barber et al. as described in Section 2.3.1 and the Appendix.

4.2.2 Anonymous Locations

Communication between Alice and Bob is facilitated through any publishing service that meets Xim's criteria for anonymous but public messaging. For example, a lightweight service could run by each participant as a Tor hidden service, and each round moved to a new .onion address. Run by Alice, this service would simply accept Xim protocol messages from anyone, and would allow anyone to read messages posted by Alice.

4.3 Mixing Fees

There are several fees associated with the mixing process. If Alice wishes to mix δ coin, then her origin address should start with a total of $n\tau + 5nf$ coin. The first $n\tau$ coin are required to pay advertisement fee τ for each of the n rounds. Every transaction also requires the payment of mining fee f so that it is committed to the block chain in timely fashion. Participants each publish either four or five transactions during the course of a round depending on whether they act as a respondent or advertiser respectively. Thus, $5nf$ coin must be reserved for mining fees (with the respondent adding an extra f to his tip). Following this scheme, at the end of n rounds, Alice will have an unlinked address holding exactly δ coin. Note that for reasons of clarity we ignored fees in protocol **Barber et al.**

PROTOCOL 2: Discover

Role Selection

1: **if** $rand(0,1) > 0.5$ **then**
2: Assume role of Advertiser $\mathcal{A}$ with DHT location α_A
3: **else**
4: Assume role of Respondent $\mathcal{R}$ with DHT location α_R
5: **end if**

Pairing

6: Advertiser: PUBLISHES $\sigma_{K_a-}\{A_\rho \xrightarrow{0} A_\rho, \text{tip} = \tau/2,\ \text{TEXT} : (\text{loc} = \alpha_a, \text{nonce} = N_a, \text{pool} = \mathcal{P})\}$
7: Respondent: Randomly selects advertiser; STORES $E_{K_a+}(N_a, N_r, \alpha_R)$ in location α_A
8: Advertiser: Selects respondent, STORES $\sigma_{K_a-}\{N_a$ paired to $h(N_r)\}$ in location α_A
9: Respondent: PUBLISHES $\sigma_{K_r-}\{R_\rho \xrightarrow{0} R_\rho, \text{tip} = \tau + f,\ \text{TEXT} : (\text{id} = E_{K_a+}(N_a, N_r))\}$

Failure Recovery

10: **if** Respondent's transaction is not committed to block chain **then**
11: Advertiser: STORES $\sigma_{K_a-}\{N_a$ unpaired from $h(N_r)\}$ in DHT location α_A
goto line 7 (wait for new respondents)
12: **end if**

Pairing

13: Advertiser: PUBLISHES $\sigma_{K_r-}\{A_\rho \xrightarrow{0} A_\rho, \text{tip} = \tau/2,\ \text{TEXT: lock} = h(N_r), N_a\}$

Failure Recovery

14: **if** Advertiser's transaction is not committed to block chain **then**
15: Respondent: STORES "N_a aborted $h(N_r)$; proof: N_r" to DHT location α_a
16: **goto** line 7 (contact a new advertiser)
17: **end if**

End

18: **return** Address of the opposite party, α_A or α_R

4.4 Splitting Large Sums of Bitcoin

One round of mixing provides unlinkability for only δ coin, which is insufficient if the mixing partner is an attacker. To mix more than δ coin, Alice mixes the coin at address A across m *mix units*. Each unit can be processed in parallel. For each mix unit, Alice partners with a new participant in each of n rounds. To unlink all of her funds, Alice's goal is to find at least one honest partner out of those n participants for each mix unit. We analyze the security of this procedure against attackers with varying resources in Section 5.2; in short, a mixer's chances of success grow exponentially with n, but also decrease exponentially with m.

4.5 Mix Coordination

We call two participants that pair for the purpose of mixing *mix partners*, and we say they form a *mix pair*. If Alice and Bob form a mix pair, then they pass through seven sequential *stages* each corresponding to the publication of one or more Bitcoin transactions (see Table 1). If they exchange coin at a time different from all other mixers, then they will leave a distinct signature on the block chain. This is because a majority of Bitcoin transactions look alike — the use of contracts is not prevalent — so mix transaction will be obvious by examination of the block chain. For this reason, participants can loosely coordinate their mix times in large groups so that many participants will tend to be in the same stage simultaneously. This loose coordination avoids a timing attack (see Section 5.5).

Announcements. Any mix participant can post a *Mix Pool Commencement Announcement* to a public bulletin board. The announcement includes four pieces of information: the minimum number of participant pairs p, number of rounds n', a unique pool number $\mathcal{P}$, and a commencement time T. During Protocol **Discover**, any advertiser who wishes to participate in the mix will add $\mathcal{P}$ as his pool parameter. When time T has been reached, the mix proceeds for pool $\mathcal{P}$ provided that at least p advertisements list it as their desired pool. Otherwise, the mix for that pool is aborted and all associated advertisements (and responses) are nullified. Once a mix begins, participants commit to continuing for all n' rounds.

	Part.	Action	Balance
1	Alice	Ad Solicicitation	$\delta + 4f + \frac{\tau}{2}$
2	Bob	Ad Response	$\delta + 3f$
3	Alice	Ad Confirmation	$\delta + 3f$
4	Bob	TxCommB, TxRefundB	$\delta + f$
5	Alice	TxCommA, TxRefundA	$\delta + f$
6	Bob	TxClaimB	δ
7	Alice	TxClaimA	δ

Table 1: The seven sequential stages of a mix round.

Synchronization. Throughout the mix, participants are careful to remain synchronized with the rest of the pool. Specifically, they all wish to pass each stage of each round at approximately the same time. But this is difficult to accomplish because completing a stage requires that all transactions from that stage are confirmed on the block chain. Section 7 demonstrates that two transactions published at the same time may each see confirmations at very different times. For example, 30% of the participants will see those confirmations in fewer than 40 minutes while another 30% will see them after more than 60 minutes. For this reason, each mix pair will wait a fixed number of minutes before proceeding to the next stage regardless of when their transactions happen to be confirmed.

Intra-round delay. Let t^{90} and t^{99} denote the number of minutes required to ensure that 90% and 99%, respectively, of mix pairs have completed a single stage. If all pairs wait a minimum of t^{90} minutes between stages, then even if mix pairs never catch up once they've fallen behind, approxi-

mately half of all participant pairs are expected to remain in the same stage throughout the round. Therefore, participants can be confident that by inspection of the block chain, their mix pair is indistinguishable from $0.5p$ other pairs. Any pairs that failed to complete a stage in time t^{90} should continue to mix by proceeding immediately to the next stage. They should be aware, however, that an attacker will have an easier time linking them with their partner for that round.

Inter-round delay. At the end of a round, participants will again wait until a total of $5t^{90} + 2t^{99}$ minutes have passed since the beginning of the round before proceeding to the next. This will ensure that all participants have completed seven stages with high likelihood. We choose this time for the following reason. The probability that a given mix pair completes at least 5 of 7 stages within t^{90} minutes each is $1 - \binom{6}{3}0.1^3 = 0.98$. For the stages that the mix pair does not complete in t^{90} minutes, the probability that they *do* complete within t^{99} minutes is at least $0.99^2 > 0.98$. Therefore, the probability that any given mix pair completes the seven stages in less than $5t^{90} + 2t^{99}$ minutes is at least $0.98^2 > 0.96$.

Jilted mix participants. Even though mix pairs will likely finish in time to begin the next round, there is always a possibility that they will fail to complete all seven stages in time. It's also possible that one partner will go offline during the round. In such cases, jilted mix participants must withdraw from the current mix pool because the protocol requires uniform Bitcoin balances for all participants. There are, however, several opportunities for recourse. First, the participant may decide to mix no further. Second, he may choose to combine the funds in the jilted address with another address and enter a later mix. Third, he may enter a later mix that requires fewer rounds for completion and avoid recombining funds with a different address.

5. ATTACKS ON XIM

Having specified a distributed mix protocol appropriate for the virtual currencies, we next turn our attention to several attack vectors. The key questions that we address in this section are: What are the costs and effectiveness of attacks that defraud or gather information about participants? How does the advertising fee increase the costs to attackers? How does mixing decrease their ability to infer linkability?

5.1 Damaging and Defrauding Participants

For the advertiser who is an attacker, the best opportunity to cause maximum loss to others given minimal cost to herself is to attack during Protocol **Discover**: she can abort before line 13, causing the respondent to lose $\tau + f$ on his advertisement, though she must spend $\tau/2$ herself. Because the respondent will publish an abort message (line 15), the attacker also risks being unable to find a new partner for her address.

An attacker can also attempt to exploit a race-condition-based vulnerability in Barber et al., to perform an unfair exchange; as we show in Section 7, this vulnerability can be mitigated with careful parameter choices.

5.2 Cost of Sybil-based Linking

Xim is susceptible to variations of the *Sybil Attack* [12]. It is possible for one entity to present as multiple identities, which can lead a victim to believe they are mixing with an inflated number of entities.

The following theorem relates the resources an attacker Eve is willing to apply to this attack to the probability of her success for the synchronous mixing scheme.

THEOREM 1: It will cost an attacker Eve a minimum of $(\frac{\tau}{2})X(1-(1-p)^{1/m})^{2/n}$ coin to partner with Alice for every round of an n-round mix with probability of success p.

The proof appears in the Appendix.

As a concrete example, suppose that there are $X = 1000$ mix advertisements and that τ is the equivalent of \$1. If Alice mixes $m = 10$ units over $n = 10$ rounds each, it will cost Eve about \$341 to succeed in tracking at least one address with 80% probability.

The theorem demonstrates the robustness of our fee-based design: the costs to the attacker are $O(X\tau)$, linear with the fee and the number of participants mixing. Without an advertising fee or without a fee per advertisement, the costs for attackers would not scale with the number of participants [18].

5.3 Bribes for Proof of Mixing

Barber et al.'s protocol [7] is susceptible to bribes because Alice and Bob exchange secrets in order to carry out an exchange. Either of them can choose to sell their secrets to a third party once they have mixed.

Currently there is no market for acquiring nonces in the Bitcoin economy. But even if such a market emerged, the existence of a cryptographic receipt does not necessarily make bribes more commonplace. These receipts provide security to mix participants when interacting with an untrusted second party and we expect the benefits to outweigh the risks. Even without cryptographic proof, a third party still has the ability to obtain information about mix participants simply by asking them for information in return for money; although the evidence may not be cryptographic, it is still useful and informative.

5.4 Intersection Attacks

Xim is susceptible to the intersection attack described in Section 3.2. Recall that we're concerned with the case where Alice exhibits unique spending behavior of her mixed coins such as purchasing hatpins. Imagine that an attacker is able to determine the entity controlling every input address, which could be accomplished for example by using the heuristics developed by Meiklejohn et al. [19]. He defines the *entity* E^A associated with Alice by the set of addresses she controlls $E^A \equiv \{A_1, A_2, \ldots\}$ where A_i is part of the ith mix. The mixes may be distributed over a period of days, months, or even years. He does not know the associated output addresses $A'_1, A'_2, \ldots$ a priori, but can observe where these addresses are eventually spent. To launch the attack, he forms a sequence of entity sets $\mathcal{E} = \{E_1, E_2, \ldots\}$ and output address sets $\mathcal{O} = \{O_1, O_2 \ldots\}$, where E_i and O_i are the sets associated with mix i. Suppose that by analyzing the block chain, the attacker identifies a unique destination address D that belongs to every set in the subsequence $\mathcal{O}' \in \mathcal{O}$. From the corresponding subsequence of entities $\mathcal{E}'$ he notes that $E^A \in \cap_{E \in \mathcal{E}'} E$. The attacker can now surmise that the probability that Alice is the entity making purchases from D is $1 \;/\; |\cap_{E \in \mathcal{E}'} E|$.

There is no simple defense for this attack, but note that its efficacy depends crucially on two factors. First, the entity

churn rate between mixes. Churn is defined as the number of new entities joining each mix plus the number of old entities leaving. If churn is zero, then the intersection set $\cap_{E \in \mathcal{E}'} E$ will always be the same size no matter how long the attacker waits. Unfortunately churn is entirely dependent on participant behavior, and thus it is difficult to predict what it will look like once Xim is deployed. The second factor is the number of entities participating in any given mix. Even when churn is low, the overall probability of attack success depends on the number of mixing entities. While existing mix protocols such as CoinShuffle and DarkWallet are limited to a small number of mix participants, Xim enables participants to refrain from mixing until the mix pool reaches a desired size (see Section 4.5). This greatly decreases our protocol's vulnerability an to inference attack relative to existing solutions.

5.5 Timing Attacks

Section 4.5 illustrated how an attacker could potentially infer that two participants were mixing together by observing the delay in publication of their mix transactions or the proximity of those transaction on the block chain. To combat such inference, we introduced coordinated mix times for all participants in a particular mix pool. Under these conditions, all participants pass through each mix stage at approximately the same time with high probability. Specifically, about half of all participants will remain synchronized throughout all seven mix stages. This means that by observing transaction publication and confirmation timing, an attacker cannot distinguish the majority of participants from the others. Any participants that do not stay synchronized for a particular round will be made aware of this fact and will be encouraged to adjust their security expectations accordingly.

6. PARAMETER SETTINGS

Selecting n. Alice mixes $m\delta$ coin, separated into m mix units, each mixed over n rounds. To set an appropriate value for n, we derive the fraction of the m units she can expect to successfully mix in the presence of an attacker that controls a fraction of all advertisements.

For each unit of δ, Alice must make sure that she has successfully avoided an attacker, Eve, at least once. Alice alternates between contacting a mixer participant and being contacted. Therefore, in the best case, she has control over only half of the rounds ($\frac{n}{2}$), where she can freely choose her mix partner. It's possible that the rest of the rounds can be controlled solely by the attacker.

Suppose that the attacker represents a fraction x of the X advertisers. Alice successfully mixes a unit if Eve does not observe at least 1 of the $\frac{n}{2}$ rounds controlled by Alice per given δ. The probability that any given unit goes unobserved at least once by the attacker is 1 minus the probability that none are, which is

$$1 - x^{n/2} \tag{1}$$

A plot of Eq. 1 appears in Fig. 3 as dashed lines for three Sybil populations: $\frac{1}{9}$, $\frac{2}{9}$, $\frac{3}{9}$, and $\frac{1}{2}$. The solid lines represent a simulation of 1,000 trials of each point. In the simulation, peers randomly select the role of advertiser or responder in each round. This strategy avoids an inference attack, but performs slightly worse than the simplified assumptions of the equation. (Confidence intervals are not shown since they are negligible in size.) Even with half the mix represented by a Sybil, mixing success reaches 95% after 10 rounds.

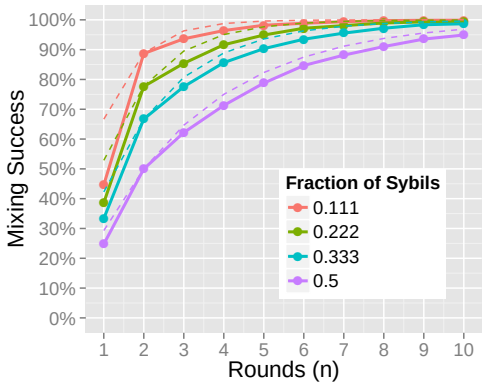


Figure 3: The chance of successful mixing of an address over n rounds, given a fraction of Sybil attackers in the mix pool. The dashed lines are based on the simple model of Eq. 1. The solid lines are the results of a simulation.(Confidence intervals are not shown and negligible.)

Setting τ. The overhead of our scheme is $n(\tau + 5f)$ coin per mixing of δ over n rounds. Transaction fees are small and set by the network, whereas τ is in our control and ideally quite small. It's ideal to keep δ relatively large so that a participant with a large amount of coin to mix is not overburdened by the advertising fee. On the other hand, δ is the smallest mix unit, so it needs to be small enough to attract a maximum number of participants. Satisfying the needs of the majority is crucial to creating a thriving mix.

Different mixes could serve different communities of interest, but the values for τ, δ, and to a very large degree n, must be consistent within each community.

7. DELAY ANALYSIS

In this section, we analyze the delay overhead incurred by Xim if used on the Bitcoin network. We measured commit delays on Bitcoin for 21-days and characterize the delay distribution both for commits and for blocks to be settled up to 5 places back into the block chain. In sum, we observed 890,467 unique transactions posted to 2,517 unique blocks[2]. In addition to using the data to determine the delay overhead of Xim, we are also able to determine the correct setting for timelocks in the refund transactions of Barber et al.'s fair exchange protocol; we find there is a race condition if set incorrectly.

7.1 Xim Delay Overhead

Using five listeners on the Bitcoin broadcast network, we collected transaction times between 2014-04-09 to 2014-04-29. We recorded each transaction's earliest broadcast time and the time when it appeared on the block chain. In the case of block chain forks, we recorded the time of the block after the fork was resolved.

Figure 4 shows the distribution of the delay between when a transaction is first broadcast and when it appears as the last (i.e., newest) block in the block chain. We also calculated the delay until the transaction appears 1–5 blocks behind

[2]An average of 353 transactions per block; See `http://blockchain.info/charts/n-transactions-per-block`.

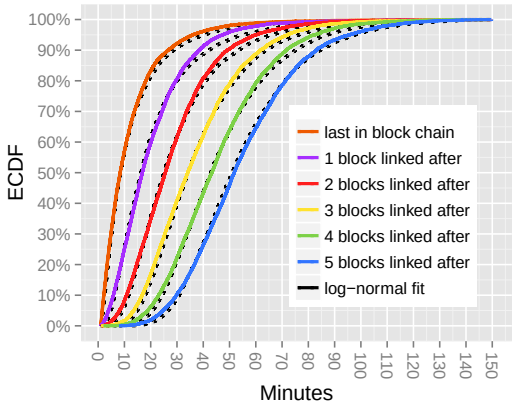


Figure 4: The (empirical) cumulative distribution function (or ECDF) of the delay until transactions appear in the block chain. The black dotted lines represent a fit to a log-normal distribution. This skewed distribution of when transactions are committed to the block chain stands in contrast the managed 10-minute average delay between the addition of blocks to the chain itself. (Data from April 4–29, 2014.)

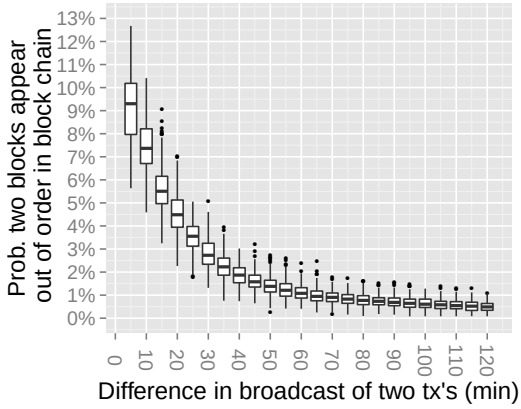


Figure 5: The chances that two transactions appear out of order on the block chain given the difference in times they are broadcast. Data at each interval is displayed as a boxplot; the median is the center horizontal line of each box. (Data from April 4–29, 2014.)

the newest block. As Figure 4 shows, these delays are highly skewed. We fit the data to several distributions, and the heavily skewed log-normal resulted in the best fit. Bitcoin's miners are calibrated to post a new block to the block chain once every 10 minutes; but they are not calibrated to ensure that transactions are posted according to any policy. Miners do not post the largest number of transactions possible in each block for fear of losing the race to post a winning result (which requires all bytes posted and not just the resulting hash value). Accordingly, while the 50% of transactions are committed to the block chain in at most 8 minutes, 25% take at least 15 minutes, and 5% take at least 35 minutes. Delays for waiting until a block is further into the chain are also log-normally distributed, seeing heavy-tail delays as well.

As a result, we can provide estimates of the delay of Xim rounds based on the inter-round delay analysis from Section 4.5. In the case that peers would wait only until a transaction appears last on the block chain, the inter-round delay is $5t^{90} + 2t^{99} = 5(27) + 2(64) = 263$ minutes. To wait until transactions are second from the last block, the delay is $5(50) + 2(88) = 426$ minutes. Most conservatively, a requirement that each transaction is sixth from the last block incurs an inter-round delay of $5(84) + 2(122) = 664$ minutes. While these delays of between 4.4–11.1 hours per round are certainly lengthy, note that there is no bound on δ, and that a participant can mix many addresses in parallel. Additionally, Xim's mix success rate is very high after just 4 rounds: at 90% for a Sybil population of 1/3, and 99% for a Sybil population of 1/9.

7.2 Bitcoin Transaction Reordering

Our data also shows that, when configured incorrectly, Barber et al. is subject to a race condition that can make it unfair, allowing one participant to both retain their coin and gain control of the other participant's coin as well.This race condition exists because the protocol assumes that transactions are processed in the order they are broadcast (plus the additional required delay, or *time lock*, that the protocol enforces through the Bitcoin protocol). If the time lock on TxRefundA is not long enough, it's possible that Alice's refund appears on the block chain before Bob can claim his money. Therefore, if TxRefundA is committed on the block chain before TxClaimB, and TxClaimA is also committed, Alice would receive both her refund and Bob's money.

Fig. 5 shows the likelihood of a pair of transactions that were broadcast in a certain order to appear in reverse order on the block chain. On average there is a 7.5% chance of two transactions appearing out of order on the block chain if they are less than 20 minutes apart. Fortunately, the timelock value (l in the Appendix) can be set to start very distantly in the future without affecting the performance of honest peers. Our data indicates that the value should be at least 120 minutes to ensure a 99% chance of a correct ordering. The tradeoff is that by aborting dishonest peers might cause honest peers to skip the current mix round and wait until they recover their funds.

8. CONCLUSION

We analyzed current mixing schemes used in VCs, and shown that they are susceptible to Sybil, denial-of-service, and timing attacks. As a solution, we proposed and analyzed Xim, the first complete solution for two-party mixing that is compatible with Bitcoin and related VCs. Our protocol includes a decentralized solution for anonymously finding partners and is designed to be a multi-round system with arbitrarily increasing security. It is the first protocol to simultaneously address Sybil attackers, denial-of-service attacks, and timing-based inference attacks. Xim uses fee-based advertisements to pair partners for mixing, and provides evidence of the agreement that can be leveraged if a party aborts.

We demonstrated why the specific properties of our protocol increase economic privacy. Specifically, we showed that fees and multiple rounds make the protocol more robust against Sybil and cost-free denial-of-service attacks. Loosely coordinating the actions of participants thwarts timing attacks and linkability of addresses. Finally, using data we collected from Bitcoin, we showed that the heavy-tail delay distribution of transaction commitments impacts the efficiency of our protocol. Fortunately, because Xim can be operated in parallel and there is no restriction on the amount of coin that can be mixed per address, these delays do not increase with the amount of coin mixed.

9. REFERENCES

[1] Coinjoin. https://bitcointalk.org/index.php?topic=279249.0, May 2014.

[2] Coinshuffle. http://crypsys.mmci.uni-saarland.de/projects/CoinShuffle/coinshuffle.pdf, May 2014.

[3] Darkwallet. https://wiki.unsystem.net/index.php/DarkWallet/Alpha#Mixing, May 2014.

[4] Send shared. https://blockchain.info/wallet/send-shared, May 2014.

[5] Sharedcoin. https://sharedcoin.com/, May 2014.

[6] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating User Privacy in Bitcoin. In *Proc. Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 34–51. 2013.

[7] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to Better — How to Make Bitcoin a Better Currency. In *Proc. Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 399–414. 2012.

[8] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *To appear in Proc. Financial Cryptography*, 2014.

[9] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, May 2004.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proc. USENIX Security Symposium*, pages 303—320, August 2004.

[11] J. Douceur. The Sybil Attack. In *Proc. Intl Wkshp on Peer-to-Peer Systems (IPTPS)*, March 2002.

[12] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

[13] Shimon Even and Yacov Yacobi. Relations among public key signature systems. Technical report, Technical Report 175, Technion, Haifa, Israel, 1980.

[14] Elizabeth A. Harris and Nicole Perlroth. For Target, the Breach Numbers Grow. *The New York Times*, page B1, Jan. 11 2014.

[15] Phillip Koshy, Diana Koshy, and Patrick McDaniel. An Analysis of Anonymity in Bitcoin Using P2P Network Traffic. In *To appear in Proc. Financial Cryptography*, February 2014.

[16] Watson Ladd. Blind signatures for bitcoin transaction anonymity. http://wbl.github.io/bitcoinanon.pdf, March 2012.

[17] Litecoin. http://litecoin.org/.

[18] N. Boris Margolin and Brian Neil Levine. Quantifying Resistance to the Sybil Attack. In *Proc. Financial Cryptography and Data Security (FC)*, pages 1–15, January 2008.

[19] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Proc. Conference on Internet Measurement Conference (IMC)*, pages 127–140, 2013.

[20] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 397–411, Washington, DC, USA, 2013. IEEE Computer Society.

[21] Malte Möser. Anonymity of bitcoin transactions: An analysis of mixing services. *Münster Bitcoin Conference*, 2013.

[22] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http://www.bitcoin.org/bitcoin.pdf, 2009.

[23] H. Pagnia, H. Vogt, and F. C. Gaertner. Fair Exchange. *The Computer Journal, vol. 46, num. 1, p. 55, 2003.*, 46(1):55–78, 2003.

[24] Nicole Perlroth. Target struck in the cat-and-mouse game of credit theft. *The New York Times*, page B1, 20 December 2013.

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

APPENDIX

A. PROOF OF THEOREM 1

THEOREM 1 It will cost an attacker Eve a minimum of $(\frac{\tau}{2})X(1-(1-p)^{1/m})^{2/n}$ coin to partner with Alice for every round of an n-round mix with probability of success p.

PROOF: Suppose there are X mixer advertisements and that Eve has submitted a fraction x of them. Since there are an even number of rounds we know that exactly $n/2$ of those rounds will have Alice choosing her mix partner. The probability that Alice chooses a mix partner controlled by Eve in one of these rounds is x. We assume the worst case for the other rounds, that Eve always partners with Alice. The probability that Eve pairs with Alice for all n rounds is $x^{n/2}$. It follows that the probability that all of Alice's m mix units go unobserved is $(1-x^{n/2})^m$. Therefore, the probability that Eve can partner with Alice on every round for at least one mix unit is $p \equiv 1-(1-x^{n/2})^m$; and so $x = (1-(1-p)^{1/m})^{2/n}$. Eve's cost for all the mix participation advertisements is $(\frac{\tau}{2})Xx$. (We ignore the additional small cost of $\tau/2$ each time the attacker successfully mixes with Alice.). Substituting for x, the result follows.

□

B. BARBER ET AL.'S FAIR EXCHANGE

A description of Barber et al. is presented in Figure 6.

PROTOCOL 3: Barber et al.(A, A', B, B', δ)

1: Alice, Bob: perform a cut-and-choose resulting in Bob knowing values x and y, and Alice knowing x, y', and z, such that (with arbitrarily high probability) $y' = H(y)$ and $z = H(x + y)$. Alice generates two key pairs with private keys K_{A1} and K_{A2}; Bob generates two key pairs with private keys K_{B1} and K_{B2};

2: Bob: GENERATES TxCommB: ($B \xrightarrow{\delta} script$, having an output script redeemable by a subsequent transaction having either: *(i)* An input script signed by K_{A2} and K_{B2}; or *(ii)* An input script signed by K_{A2} with a value whose hash is y').

3: Bob: GENERATES TxRefundB: ($script \xrightarrow{\delta} B$, having an input script signed by K_{B2}; timelock$= l + 3$).

4: Alice: Adds her signature K_{B2} to the input script in TxRefundB.

5: Bob: **publishes** TxCommB and TxRefundB.

Once Alice is satisfied that TxCommB is committed to the block chain, she proceeds.

6: Alice: GENERATES TxCommA: ($A \xrightarrow{\delta} script$, having an output script redeemable by a subsequent transaction having either: *(i)* An input script signed by K_{A1} and K_{B1}; or *(ii)* An input script signed by K_{B1} with a value whose hash is z).

7: Alice: GENERATES TxRefundA: ($script \xrightarrow{\delta} A$, having an input script signed by K_{A2}; timelock$= l + 1$).

8: Bob: Adds his signature K_{B1} to the input script in TxRefundA.

9: Alice: **publishes** TxCommA and TxRefundA.

Once Bob is satisfied that TxCommA is committed to the block chain, he proceeds.

10: Bob: **publishes** TxClaimB: ($script \xrightarrow{\delta} B'$, having an input script signed by K_{B1} with a value $x + y$ whose hash is z). Alice is able to deduce the value of $y = x + y - x$ from the above transaction.

11: Alice: **publishes** TxClaimA: ($script \xrightarrow{\delta} A'$, having an input script by K_{A2} with a value whose hash is y').

If either party aborts after Step 5, the already-published timelocked transactions return the funds to their original addresses.

Figure 6: Barber et al.'s protocol. This type of exchange is possible because Bitcoin allows cryptographic contracts to be written into transactions, specifically TxCommA and TxCommB. In this case, TxRefundB and TxRefundA require signatures from both Alice and Bob, providing guarantee that money can be refunded if one party aborts. Once Bob publishes TxClaimB, Alice learns the missing values to satisfy condition *(ii)* of TxCommB. Accordingly, she writes the missing values into TxClaimA and publishes shortly after Bob. In other words, Bob can't receive his money unless he enables Alice to receive hers. The value of l is discussed in Section 7.2.