

Mixcoin

Anonymity for Bitcoin with accountable mixes (Full version)

PRE-PROCEEDINGS DRAFT version 0.163

Joseph Bonneau¹, Arvind Narayanan¹, Andrew Miller², Jeremy Clark³, and
Joshua A. Kroll¹

¹ Princeton University
² University of Maryland
³ Concordia University

Abstract. We propose Mixcoin, a protocol to facilitate anonymous payments using the Bitcoin currency system. We build on the emergent phenomenon of currency mixes, adding an accountability mechanism to expose theft by mixes. Unlike other proposals to improve anonymity with Bitcoin our scheme can be deployed immediately with no changes to Bitcoin itself. We demonstrate that economic incentives of mixes and clients can be aligned to ensure that rational mixes will not steal from clients. We compare mixing for financial anonymity to the much-better studied problem mixing for anonymous communication, demonstrating important and subtle new attacks.

1 Introduction

Protecting the privacy of financial transactions has long been a goal of the cryptography community, dating at least to Chaum's work on anonymous digital cash using blind signatures [3]. Despite initial excitement, anonymous digital payments have not seen mass adoption. Perhaps one reason is that traditional electronic cash requires a centralized, trusted entity, a bank.

By contrast, Bitcoin is a relatively young decentralized currency that has rocketed to popularity with a current market capitalization of over USD \$10 billion. Bitcoin can be thought of as a public, distributed ledger that logs all transactions in order to prevent double spending. Using a proof-of-work system, the integrity of the ledger is maintained as long as the majority of the computing power is contributed by honest participants [16].

Bitcoin does not provide true anonymity: transactions involve pseudonymous addresses, meaning a user's transactions can often be easily linked. Further, if any one of those transactions is linked to the user's identity, all of her transactions may be exposed. A small but growing body of academic literature has found that Bitcoin offers only weak anonymity in practice (see Section 2.2).

The Bitcoin community is well aware of this issue, leading to much interest in the provision of stronger anonymity. One proposal, Zerocoin [14], provides

strong cryptographic anonymity but cannot be deployed without modification to Bitcoin miner’s transaction validation process. Mixing services enable a trusted entity to obfuscate the ownership of Bitcoin but there is no theft-protection. CoinJoin allows Bitcoin users to orchestrate a mix without trusting a third party with the temporary possession of their coins, however it becomes complex for participants to protect their anonymity from each other, and the protocol can be easily disrupted by a malicious entity.

We provide more detail of these solutions in Section 2.3, but for now note that the design of a robust protocol with strong anonymity properties that can be deployed immediately with no changes to Bitcoin is an open problem, and the one that we address. Our strategy is to build on the existing phenomenon of mixes, but to add an independent cryptographic accountability layer. Our contributions include the following main ones.

Accountability. Mixcoin mixes issue signed warranties (Section 3) to users which roughly state: “if v coins are sent to me from address κ_{in} by time t_1 , I will send v coins to address κ_{out} by time t_2 .” Here κ_{in} and κ_{out} are owned by the client. Given this warranty, a user can confidently send funds to the mix, knowing that if the mix misbehaves she can publish the warranty. This will damage the mix’s reputation and (presumably) its business model.

Randomized mixing fees. Our second major insight is the design of mixing fees. We show how rewarding mixes with a small percentage of the amount mixed provide incentives for rational mixes to behave honestly (Section 4). Yet simply collecting a fixed fee undermines building an effective mix network. Instead we apply randomized mixing fees [19], in which mixes are entitled to keep the entire amount of a small percentage of transactions (and nothing from others). We show how to generate the requisite randomness in a fair and accountable manner using the unpredictability of the Bitcoin blockchain itself.

Mix networks for Bitcoin. Our third innovation, inspired by anonymous communication networks, is to chain multiple mixes together. This both improves anonymity and guards against an adversary in control of some of the mixes. There are important differences from communication mixes, however, introducing subtle new attacks (Section 5). Mixcoin offers strong anonymity against an adversary who, in addition to being able to analyze the Bitcoin blockchain, controls some mixes and can engage in transactions with the honest mixes.

Our design philosophy is a very general core protocol, allowing clients and mixes to specify a variety of free parameters. Yet we expect that, because anonymity loves company, these parameters will converge to global values (Section 3.7). In particular, we expect mixing to complete in a few hours with mixing fees of less than 1% (Section 4). Given this modest overhead and the fact that Mixcoin can be deployed immediately with no changes to Bitcoin itself, it is our hope that Bitcoin users will mix their funds as a matter of course, finally making an anonymous, decentralized digital currency a reality.

2 Background and related work

In this section we provide a basic model of Bitcoin. We focus on the properties required for Mixcoin, which could be implemented on top of any distributed currency system similar to Bitcoin in these basic respects. We then model today’s nascent Bitcoin mixes and the attacks they are vulnerable to.

2.1 Bitcoin

Bitcoin can be thought of as a decentralized system which tracks a mapping between *addresses* and monetary value denominated in coins. An address, which we denote κ , is simply a public/private key pair. Addresses are pseudonymous: anybody can create an arbitrary number of addresses for free with no verification. Control of an address’s private key provides “ownership” of all coins mapped to that address. A Bitcoin *transaction* is essentially a statement that an address κ_{in} would like to transfer some value v to an address κ_{out} . This statement must be signed by κ_{in} and broadcast to the entire network.

A distributed consensus protocol prevents double spending and maintains a unified history of all transactions. Transactions are grouped into *blocks* for efficiency, which are chained in a linear structure called the *blockchain*. The chain represents (probabilistic) consensus; at present most Bitcoin users will consider a transaction confirmed if it appears in a block with at least $w = 6$ blocks following it. New blocks are generated roughly once every ten minutes.

2.2 Deanonymization in Bitcoin

Creating new addresses is trivial, but this does not make Bitcoin anonymous as all transfers are globally (and permanently) visible in the blockchain. Several recent papers have studied ways to link a user’s addresses to each other and to an external identity [13,18,20,2].

Four major techniques have been explored. First, in multi-input transactions, shared spending authority is evidence of joint control. The rationale is that different entities are unlikely to be co-spenders of the same transaction. Second is the use of taint analysis to track flows. The taint between addresses κ_{in} and κ_{out} is defined as the percentage of the balance of κ_{out} that came from κ_{in} . Taint analysis can overcome simple manual obfuscation techniques. More generally, one can analyze Bitcoin’s overall *transaction graph*, with each address as a node and each transaction as a weighted, directed edge between nodes. Various heuristics can be used for clustering addresses and tracking flows, such as heuristics for detecting the “change address” in a multi-output transaction (the rationale being that the change address is probably controlled by the same entity as the input address(es)). Third, there are various side channel attacks such as timing, precise amounts, and network-layer information. Finally, an attacker can use using auxiliary information (e.g., in forums) to connect pseudonyms to identities.

The first three techniques have proved quite powerful. The fourth technique (use of auxiliary information) is necessary to link a Bitcoin user to a real-world identity. However it is more difficult for researchers to carry out using public information than for a well-funded adversary who may compel disclosures from various service providers, so the academic literature has generally stopped short of this step. Nonetheless, the evidence suggests that the anonymity properties of Bitcoin cannot be relied upon in the face of a committed adversary.

2.3 Anonymity technologies for Bitcoin

Caution: Mixing services may themselves be operating with anonymity. As such, if the mixing output fails to be delivered or access to funds is denied there is no recourse. Use at your own discretion. — The Bitcoin Wiki (2013)

There have been two prominent attempts to provide stronger anonymity in Bitcoin. One is an academic proposal called Zerocoin [14] which uses cryptographic techniques (specifically, an accumulator with a zero-knowledge proof of inclusion) to break the link between individual Bitcoin transactions without adding trusted parties. Unfortunately, deployment of Zerocoin (or related variants, such as Pinocchio Coin [7]) would require modifications to Bitcoin which appears unlikely to happen due to the computational overhead. Additionally, the cryptographic constructions proposed require a trusted setup phase, where knowledge of the trapdoor created during setup would allow an attacker to forge coins (though this limitation may be removed using techniques like RSA-UFOs [22]).

A second approach, arising organically from the Bitcoin community, is *mixing*, directly analogous to the concept in communication networks. The simplest implementation is a mixing address which receives coins from multiple clients and forwards them randomly to a fresh address for each client. Because Bitcoin transactions are irreversible, clients have little remedy if the mix operator simply steals their funds (*cf.* [13]). Several such services have arisen, typically charging commissions in the 1–3% range and requiring manual interaction through a website to arrange transactions.⁴ A small-scale study of three mixing services found that in one case (BitLaundry), taint analysis is immediately sufficient to link the input and output [15]. In the other two cases, Bitcoin Fog and blockchain.info, taint analysis did not succeed but the transaction graph showed rich structure, leaving open the question of more sophisticated linking attacks. Anecdotal evidence from user forums include complaints of stolen coins, slow mixing times of up to 48 hours, and low transaction volumes leading to users frequently receiving their own coins in return.⁵ The popular Bitcoin Wiki currently warns: *Caution: ... if the mixing output fails to be delivered or access to funds is denied there is no recourse. Use at your own discretion.*

In contrast to dedicated mixing services, some services with a preexisting high trust requirement have deployed implicit mixing successfully. For example,

⁴ Some mixing services are only accessible as Tor hidden services.

⁵ Note that receiving one's own coins is not necessarily a vulnerability. This will happen with probability $\frac{1}{N}$ with in a random permutation of N participant's coins.

the Silk Road anonymous marketplace mediated and mixed all transactions between buyers and sellers, while some “eWallet” services promise that when users withdraw they will receive random coins from the provider’s reserves.

Finally, a proposal called CoinJoin [12] enables n users to atomically transfer funds from their n input addresses to their n output addresses in a single transaction, signed by each key corresponding to the inputs, which is possible without modification to Bitcoin. Since no entity will sign the transaction if its own output address is not specified correctly, there is no risk of theft. However arranging the output addresses in a way that prevents the participants from learning the correspondence to input addresses (*i.e.*, providing anonymity) introduces complexity, with most designs relying on external cryptographic protocols [6] and/or a third-party facilitator. An attacker can easily block the transaction by participating initially to form the transaction, but failing to sign the finalized transaction. A simpler alternative adds a trusted “facilitator” to perform the mixing, though this re-introduces trust issues inherent in mixing services.

Despite these problems, enthusiasm and interest in mixing is high. The recent press on “dark wallets” for Bitcoin using Coinjoin is an example [11]. There has also been vigorous activity in the Bitcoin community on ideas for making mixing secure and viable. However, to the best of our knowledge, the ideas we introduce have not been explored previously.

2.4 Mix networks and accountability

Mix networks were introduced by Chaum in 1981 for anonymous communication [4]. Early work considered the relationship between design parameters, such as route selection and flushing policies, and the resulting anonymity (see [17] for a survey). One line of research on adding some form of integrity or accountability to mixing is verifiable mixing, beginning with Sako and Killian [21], where each mix issues a proof that its output is a permutation of its input—this is particularly important when the user cannot trace their own input through the mix. A second research direction is reputable mixing, beginning with [10], where each mix (in particular the last one) can prove that each output corresponds to some input, as opposed to the mix itself originating the message. Both lines of research address issues orthogonal to the one we consider. In our protocol, we are concerned that the mix will steal the coins in the input transaction, and once detected, it is too late to repeat the process with a different mix.

2.5 Mixing participants

Now we discuss mixing more formally, starting with a *client* Alice (A). While there are some differences in how current mixes operate, we describe the model on which Mixcoin’s design rests. A Bitcoin user owning some number of coins at an address κ_{in} which we assume is linkable to her real world identity. Alice wishes to transfer some of her funds to a fresh address κ_{out} in such a way that it is difficult to link κ_{out} to κ_{in} and hence Alice herself, in exchange for a mixing fee.

Alice will send some of her coins to a mix M , a for-profit entity which effectively hold Alice’s funds in escrow for an agreed time period and then send an equal value to κ_{out} (possibly minus a mixing fee). We don’t require M to have any real-world reputation or assets, only to consistently use the same cryptographic key K_M long enough to build a virtual reputation.

2.6 Mixing threat model

With simple mixes, Alice is exposed to two major threats:

Theft Because mixes routinely send funds to fresh addresses with no transaction history, it is possible for a malicious mix to send Alice’s funds to its own secret address κ_M instead of κ_{out} as requested. Though Alice can publicly complain about the theft and attempt to undermine M ’s reputation, there is no way for observers to determine which of A or M owns κ_M so Alice’s claim could be libelous.⁶ For-profit mixes may rationally attempt to undermine trust in their competitors through false accusations of theft. Because allegations of theft cannot be proven, it is difficult to determine which mixes are honest.

Deanonymization Because the mix learns that the same party owns both addresses ($\kappa_{\text{in}}, \kappa_{\text{out}}$), Alice’s anonymity depends on the mix keeping this pairing secret forever. A mix which is malicious, compromised or subpoenaed might share its records and destroy Alice’s anonymity. Alternately, the mix could send coins in a non-random manner which reveals the connection to observers.

3 The Mixcoin protocol

Our goal with Mixcoin is to provide *accountability*. Prior to mixing, the mix gives Alice a signed warranty enabling her to unambiguously prove if the mix has misbehaved. Dishonest mixes will quickly have their reputation destroyed and lose business. Security against theft thus reduces to properly aligning economic incentives of mixes and clients.

However, there is no way to prove that a mix is not storing records sufficient to deanonymize its clients. Similarly to mix networks providing anonymous communication, Alice can mitigate this risk by relaying coins through a series of mixes which must all collude in order to deanonymize her final output addresses.

3.1 Assumptions

We assume the availability of multiple mixes M_i , each represented by a warranty-signing key K_{M_i} . The mix’s reputation is strictly tied to K_{M_i} , changing it effectively makes the mix a new entity. As for-profit enterprises, mixes are motivated

⁶ It’s not even possible for observers to determine which specific transaction represents the theft of A ’s coins, since by design they are randomly sent to other clients.

to build and maintain a reputation for honest behavior. Unlike mixes, Alice does not need to maintain any long-term public key nor any public reputation.

Alice must be able to negotiate with the mix over an anonymous and confidential channel. In practice this will likely be realized by mixes running a dedicated Tor hidden service, but this is out of scope of the Mixcoin protocol itself. Ideally, this channel will also be deniable for clients, so that the mix cannot prove that any client contacted it about mixing funds.

3.2 Core protocol

We introduce the core Mixcoin protocol in Construction 1 which mixes a single “chunk” v of Alice’s funds in a single mixing round. For effective anonymity, chunk sizes should be standardized. While the core Mixcoin protocol can stand on its own, typically Alice will need to split her funds into multiple transactions of size v and perform multiple sequential rounds of mixing for each.

The key accountability mechanism in Mixcoin is Alice receiving a signed warranty prior to mixing. Alice specifies mixing parameters over an anonymous channel in Step 1, specifically a chunk size v , an input address κ_{in} , an output address κ_{out} , a deadline t_1 by which Alice will transfer funds, and a nonce n . The mix may accept the terms (Step 2a) by generating a fresh escrow address κ_{esc} , a deadline t_2 for transferring funds back to Alice, and a mixing fee rate ρ , and sending back a warranty consisting of the mix’s signature (using K_M) on all parameters. The mix may also reject Alice’s request for any reason (Step 2b), though in practice we expect that a reputable mix will abide by a public policy for acceptable terms.

Alice has no obligation to transfer funds after receiving a warranty. Alice may decline (or forget) to do so by the deadline t_1 in which case the mix may simply destroy its records and move on. Prior to transferring funds, Alice’s participation is also deniable because her request in Step 1 isn’t signed and doesn’t prove possession of κ_{in} . We will discuss security implications of this in Section 3.4.

Observe that Alice doesn’t specify t_2 or ρ . Mixes will specify these in a published policy (t_2 will be specified as an increment over t_1) and Alice may consider them when deciding to which mix to use. The mix places them in the warranty so that it contains all the parameters necessary to check whether the mix has breached it. Since it is the warranty in step 2a that is actually binding and not the mix’s stated policy, it is Alice’s responsibility to check these values before proceeding to transfer funds.

If Alice transfers the agreed value v to κ_{esc} by the deadline t_1 (Step 3),⁷ then the mix is obligated to transfer an equal value to κ_{out} by time t_2 (unless if the funds are retained as a mixing fee—see Section 3.6). If the mix does so faithfully (step 4a), then after this transaction both parties should destroy their records to ensure forward anonymity against any future data breaches.

⁷ The deadline will be specified as a block number in the Bitcoin blockchain, rather than a clock time, to enable unambiguous auditing.

Note that the transfer to κ_{out} will typically not be from the same κ_{esc} , but from a different κ'_{esc} to hide the link between κ_{in} and κ_{out} . κ'_{esc} is the mix's escrow address from some previous transaction. This address is not included in the warranty, by design, since the pool of candidate addresses for κ'_{esc} , namely input addresses from all pending mixing runs at time t_2 , isn't known at the time of initial negotiation.

If the mix fails to transfer the value v to κ_{out} by time t_2 (Step 4b),⁸ then Alice publishes her warranty (Step 5). Because the warranty is signed by the mix's long-term key K_M and all Bitcoin transactions are publicly logged, anybody can verify that the mix cheated Alice by never sending funds to κ_{out} .

Publishing the warranty reveals the link between Alice's current address κ_{in} and the destination address κ_{out} . For this reason Alice must always generate a fresh κ_{out} for every run of the protocol, ensuring that she will only reveal her ownership κ_{out} if the mix has failed to transfer funds and the new address is worthless. Alice should similarly destroy κ_{out} if the mix rejects her request, because the mix will have learned of the connection between κ_{in} and κ_{out} .

3.3 Chunk sizes

Though we leave the chunk size v free to change in each protocol run, it is visible in the blockchain in both the transfers from κ_{in} to κ_{esc} and from κ'_{esc} to κ_{out} and hence the anonymity set of κ_{out} can only consist of other addresses sending an identical value v . Therefore in practice each mix should require a consistent v for all users. There is an inherent trade-off: setting v too high will exclude users owning less than v coins,⁹ while decreasing v will require proportionately more runs of the protocol and more transactions in the blockchain.¹⁰ An acceptable choice of v will require most users split their funds into a large number of chunks to be mixed in parallel. As we discuss in Section 3.7, we expect most mixes to converge to a single value or a small set of possible values of v .

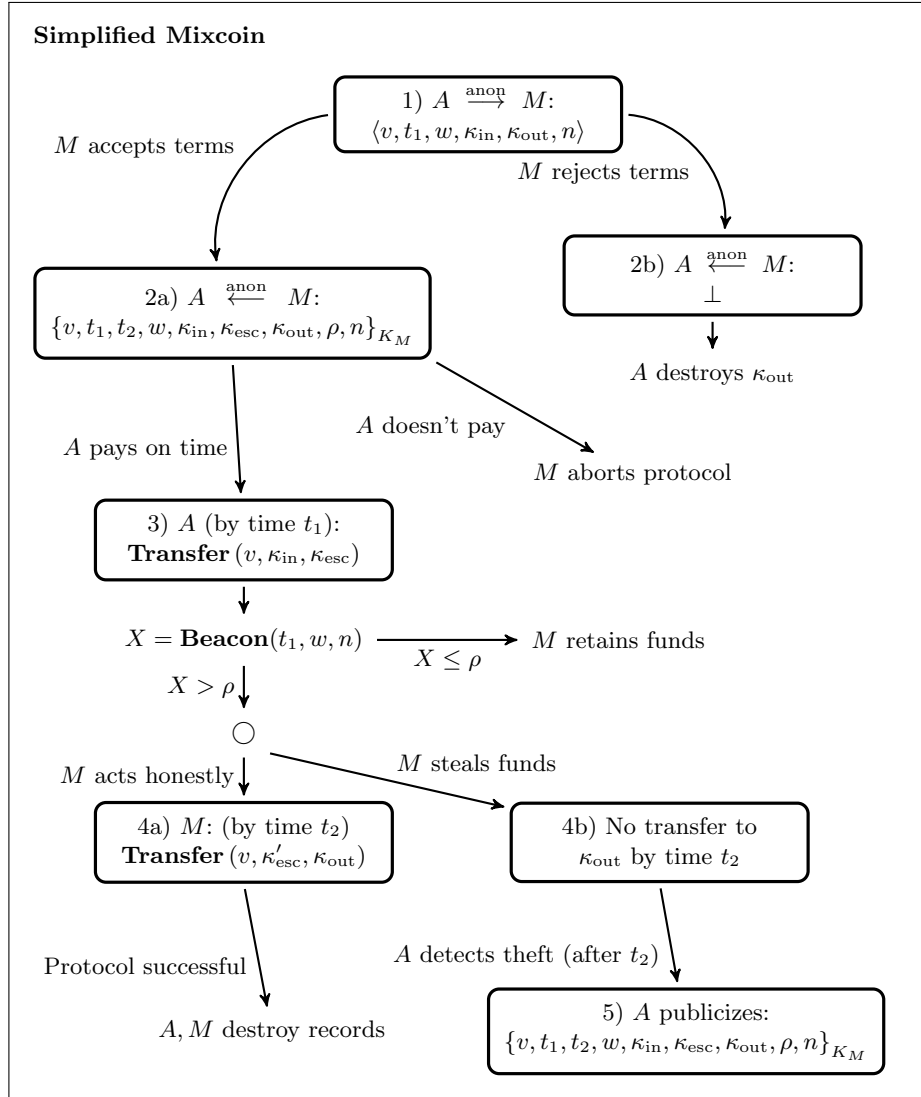
3.4 Duplicate transactions and input splitting

An important security consideration is for mixes to never issue more than one warranty with the same κ_{in} . Otherwise, a malicious user Mallory could guess that Alice will request mixing from input address κ_{in} and obtain a second warranty with the same κ_{in} but her own destination κ'_{out} , hoping that the mix will forward Alice's money to Mallory. Alternately, Mallory could obtain two warranties with the same input address κ_{in} but different output addresses κ_{out} and κ'_{out} and then

⁸ There is no way in Bitcoin to guarantee a transaction will be included in any specific block. Therefore in practice mixes will likely add a safety margin of several blocks to t_2 to ensure they can include the transaction before that time.

⁹ Additionally, with randomized mixing fees (see Section 3.6) users owning only a small multiple of v may face unacceptably high variance in their fee rate.

¹⁰ The Bitcoin community frowns on creating large numbers of low-value transactions (referred to as *dust*) because it places a higher verification burden on miners.



Construction 1: A single mixing round between client A and mix M . A owns the addresses κ_{in} and κ_{out} and M owns κ_{esc} and κ'_{esc} . The random value $X \in_R (0, 1)$ is computed using **Beacon**, a pseudorandom function using the Bitcoin block $t_1 + w$ plus the nonce n , and compared to the fee rate ρ . Times t_1 and t_2 are blocks in the blockchain. Curly brackets $\{\}$ indicate a digital signature.

only send one payment to the mix. This would leave the mix unable to fulfill both warranties, enabling Mallory to gainsay the mix's honesty.

We could address this by enabling multiplexed warranties which specify one input κ_{in} and a list of outputs of $\kappa_{\text{out}}^1, \kappa_{\text{out}}^2, \dots$ which should each receive one chunk of the input funds. We prefer a simpler solution though which leave the core protocol as lightweight and easy to verify as possible:

Therefore to mix multiple chunks simultaneously with the same mix, Alice must divide her funds before mixing and send each chunk j from κ_{in} to a fresh address κ_j^0 prior to mixing. While each κ_j^0 is trivially linkable to κ_{in} , this ensures each chunk sent to the mix has a separate warranty for auditing purposes. Because the transactions to each κ_j^0 are from Alice to herself, no confirmation period is necessary so this input splitting adds only one block of latency.¹¹

A subtle implication of the no-duplicate-warranty rule is that Mallory can perform a denial-of-service attack on any publicly known address¹² by flooding the mix with spurious mixing requests from that address. However only a digest of the full public key appears in the blockchain when funds are transferred to it. Therefore, to protect herself from this attack, Alice must always obtain a warranty for a new address κ_{fresh} prior to using the account in a transaction, as this requires disclosure of the full public key for signature verification.

Similarly, Alice must ensure that κ_{out} has no potential source of income except funds returning from the mix, which means she can't run simultaneous mix rounds with the same κ_{out} . If she did, a clever mix could observe if v has been separately transferred to κ_{out} before the deadline t_2 , retain Alice's funds, and the warranty will still appear to an observer to have been fulfilled.

3.5 Sequential mixing

As with mix networks for communication, Alice can send her funds through multiple independent mixes to protect against the compromise of an individual mix at the expense of added latency and mixing fees. Unlike traditional mix networks, Alice needn't negotiate an end-to-end path in which each mix directly transfers funds to the next. Instead, for each chunk j Alice creates a series of addresses $\kappa_j^0, \kappa_j^1, \dots, \kappa_j^N$ for N rounds of mixing,¹³ with κ_j^i serving as the output address for the j^{th} chunk after the i^{th} round and the input address for the $(i + 1)^{\text{th}}$ round. Returning each chunk to Alice in between rounds keeps warranty negotiation and auditing as simple as possible, while only adding one block of latency per round of mixing.¹⁴ It is also possible to avoid this latency all together by negotiating contracts with each mix in sequence and pipelining the mixing process.

¹¹ Alice can broadcast her $\kappa_{\text{in}} \rightarrow \kappa_j^0$ and $\kappa_j^0 \rightarrow \kappa_{\text{esc}}^1$ transactions all at once, but miners' policies vary towards including simultaneous dependent transactions in a single block. Typically, without miner fees they incur an additional block of latency.

¹² For example during sequential mixing, described Section 3.5, Mallory may target any new address receiving v coins as a possible intermediate address during mixing.

¹³ Note that $N + 1$ address must be created for each chunk due to the input splitting.

¹⁴ Each mix will want to wait for a normal confirmation period before accepting a payment, but Alice can send funds in the following block after receiving them from the mix as her signed warranty will be breached if the mix double-spends.

It also enables Alice to dynamically change a chunk’s path while mixing is underway, something not possible in traditional mix networks. Note that to prevent denial-of-service, warranties should be obtained for round i before the protocol for round $i - 1$ is completed (and the addresses κ_j^i are visible).

3.6 Mixing fees

A simple approach is to specify a fixed mixing fee rate ρ and have the mix transfer $(1 - \rho) \cdot v$ instead of the full v to κ_{out} . However, this is problematic for sequential mixing, as the smaller output value $(1 - \rho) \cdot v$ cannot be the input to a subsequent round with the original v . This could be addressed with mixes using diminishing transaction sizes $v_i = (1 - \rho)^i \cdot v$ for each round i , but this would trivially leak which round i each transaction represents.

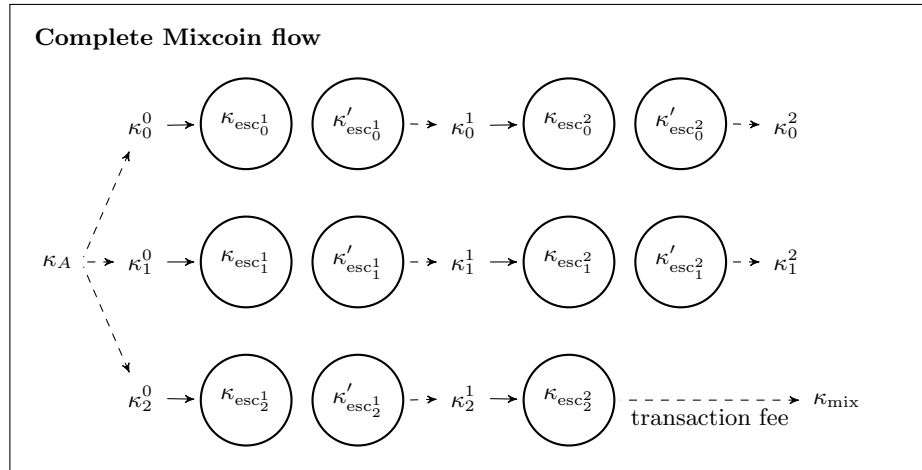
The solution is *randomized* mixing fees, whereby with probability ρ the mix retains the entire value v as a fee, and with probability $(1 - \rho)$ takes no fee at all. This produces an expected mixing fee rate of ρ and leaves κ_{out} with either nothing or a full v which can be directly re-mixed. This solution is inspired by the idea of electronic lottery tickets [19] used in some micropayment systems.

We require the mix to use a publicly verifiable mechanism to make the random choice of retaining Alice’s funds as a mixing fee. Specifically, the mix generates a $(\rho, 1 - \rho)$ -random bit such that neither Alice nor the mix can predict or influence it, and that it is apparent afterward that it was fair. A public source of randomness called a *beacon* can be used for this purpose. There are several options for generating such a bit. A coin-tossing protocol between Alice and the mix would introduce an extra round of communication, thus we concentrate on using a *beacon*, or public source of randomness.

If the beacon is external to Bitcoin (*e.g.*, NIST’s beacon [1] or financial data [5]), warranties would need to be synchronized to real-world time to enable auditing—a source of complexity. Alternatively, randomness could be generated from future Bitcoin blocks in the blockchain, assuming the exact set of future transactions included in each block (as well as the random nonce used by the miners to solve the proof of work) is unknown.¹⁵ Because each block includes the value of the previous block, every transaction during a confirmation delay period of w blocks adds randomness.¹⁶ We further utilize the fact that after Alice transfers v coins to M in block t_1 , the mix will wait w blocks as a confirmation delay. The value of the last block includes the value of each previous block (they form a hash chain), so each transaction in the span of these blocks adds randomness. The warranty also includes a nonce n specified by Alice to ensure that the mix will compute an independent value for all transactions it is managing.

¹⁵ A mix might also be a miner, in which case it may attempt to influence the block. We address the infeasibility this line of attack in Appendix A. However, such an attack is difficult and highly uneconomical given the high reward for mining a block compared to mixing fees.

¹⁶ Though in practice $w = 6$ is a common standard, we include w as a negotiable parameter in the warranty to enable flexibility.



Construction 2: A small Mixcoin flow. Alice sends three chunks worth (quantities of v) of coins through two rounds of mixing. This requires six runs of the core Mixcoin protocol described in Construction 1, all of which could be with independent mixes. The circled addresses are escrow addresses controlled by mixes. In this example one of Alice’s chunks is retained by a mix in the second round as a mixing fee. Alice is left with two output addresses holding a value of v .

Specifically, the mix in Construction 1 computes: $X = \text{Beacon}(t_1, w, n) = \text{PRNG}(n || B_{t_1+w})$, where B_i is the value of block i (*i.e.*, the Merkle root) in the Bitcoin blockchain, t_1 is the block index in which the mix receives Alice’s funds, n is Alice’s nonce, and w is the mix’s confirmation delay to guard against double spending. These are used to seed a pseudorandom number generator from which the mix can compute a uniform random value X in the range in the range $(0, 1)$, retaining Alice’s funds only if $X \leq \rho$. This computation can be performed by anybody with access to the blockchain and Alice’s warranty, which would be published in the event of a dispute. Note that this has the further advantage that in normal operation, where Alice’s warranty is kept secret, as long as n is randomly chosen no observer can tell which transactions were retained by the mix.

A drawback of randomized fees is that the variance in the effective mixing fee rate increases for users mixing a small number of chunks. To address this, v should be kept as low as possible so that most users can mixing at least $\frac{v}{\rho}$ coins.

3.7 Convergence of free parameters towards global values

Our design leaves many parameters free, such as the chunk size v , the time delay $t_2 - t_1$ and the number of rounds N . Our philosophy is to avoid embedding these into the protocol as the optimal choices may drift over time as the mixing

ecosystem evolves and the underlying parameters of Bitcoin drift. In practice however we expect most users of Mixcoin to use identical values for two reasons.

First, like with Bitcoin itself most clients will likely use one of a small number of software implementations which include reasonable parameters. Most users will avoid any configuration, except possibly parameters for the security/latency tradeoff for which default “low,” “medium” and “high” options can be presented. We also envision most users subscribing to a *mix reputation list* published by neutral evaluators which suggests a probability distribution of mixes to use which limits exposure to un reputable mixes without weighing too heavily towards a small set of mixes which could collude to break anonymity. We discuss this in more detail in Section 5.5.

Second and more importantly, all clients have an incentive to choose the most popular parameters in an application of the “anonymity loves company” principle. Unilateral variation in a user’s transaction sizes, for example, could leak information which would help Eve deanonymize Alice’s coins. Thus we expect Mixcoin users to relatively quickly converge on a global set of parameters, or possibly a small number of different “flavors” of Mixcoin, to maximize anonymity.

Mix selection is an illustrative example. We envision *mix reputation lists* (MRLs) published by neutral evaluators which estimate legitimate mix transaction volume via the methods outlined in Section C. Such lists would include a suggested probability distribution over mixes based on reputation scores, limiting exposure to un reputable mixes without weighing too heavily towards a small set of the most reputable mixes which could collude to break anonymity. Users will want to share a random path selection algorithm to avoid leaking information, and therefore we expect a few dominant MRL services to arise.

It is an open question how these parameters might change over time. For example, as Bitcoin experiences inflation or deflation the optimal chunk size will change. We expect that the ecosystem would react by introducing a new chunk size, with a transitional period where some mixes and clients use the old size and others use the new one, and the new chunk size gradually winning over.

4 Mix incentives and mixing fees

Unlike many other parameters in our system, the mixing fee rate ρ has no impact on the anonymity properties of Mixcoin and hence can be chosen independently by different mixes.¹⁷ Setting ρ requires considering the dual roles of mixing fees. First, they can cover overhead expenses for mixes such as electricity bills. Second and most importantly, they provide a mechanism for mixes to profit from indefinite honest behavior and disincentivize mixes from ceasing operations and absconding with users’ funds. Because higher fees more strongly incentivize honesty, an interesting property arises that users should avoid mixes charging less than some *minimum* acceptable value of ρ .

¹⁷ Mixes might even offer different rates to different users in an attempt to price-discriminate without compromising anonymity.

In a steady-state model, the mix has two choices at any given time: **continue** to operate honestly as a mix until the next round, or **abscond** and retain all user funds currently held. The expected value of either choice scales linearly with Q , the average amount of money flowing into (and out of) the mix during any one block. If \bar{t} is the average time period (in blocks) that the mix holds funds during a mixing round, then the expected payoff of absconding is $E[\text{abscond}] = Q \cdot \bar{t}$.

The expected payoff from choosing to **continue** would properly be defined recursively, since the mix is able to play the same game again. However, under steady state conditions the optimal decision will be the same in every round, so if the mix initially chooses to continue it will do so indefinitely. Assuming the mix is exponentially discounting future earnings¹⁸ at a rate r (per block), the net present value of indefinite honest behavior with a fee rate ρ is:

$$\begin{aligned} E[\text{continue}] &= \rho Q + (1 - r)E[\text{continue}] \\ &= \rho Q + (1 - r)\rho Q + (1 - r)^2\rho Q + \dots \\ &= \frac{\rho}{r}Q \end{aligned}$$

Incentivizing honest behavior therefore requires that $\frac{\rho}{r} > \bar{t}$. With the interpretation that r for a rational mix is equivalent to the highest available risk-free rate of return available to the mix, this condition is simply that the expected value of fees collected by a mix during the time it holds funds is greater than the amount those funds would yield during the same time period if invested.¹⁹ This can be explained by considering that we want an honest mix to continually decided to “invest” its potential earnings $Q \cdot \bar{t}$ from absconding into continuing to serve as a mix, earning a return of ρQ during every block.

Simple calculation lets us estimate that relatively low mixing fees should suffice to incentivize honest behavior. Assuming a very attractive rate of return of $r \approx 20\%$ annually is available to the mix, a mix time of $\bar{t} \approx 1$ hour gives a lower bound of $\rho_{\min} \approx 2^{-15.4}$. Even considering a chunk taking a path through 10 consecutive mixes, this still leaves only an effective fee rate of $\approx 2^{-12}$ necessary to discourage absconding. This suggests that very low mixing fees may be sufficient to cover the risk of theft. This analysis suggests that very low mixing fees may be sufficient, though absconding might be slightly more favorable in practice due to several factors missing from a simple analysis, such as super-exponential time discounting by the mix, the risk that business may decline, or the fact that the mix can choose to abscond during a period of unusually high transaction flow. Actual mixing fees will be dominated by operating costs of mixes, suggesting that any mix which has been operating for a non-trivial period of time is turning a profit and is unlikely to abscond. the additional benefit from stealing funds is negligible. Unfortunately Alice can't know what mixes' operating costs are, and

¹⁸ The exchange rate of bitcoins may of course be drastically different in the future. We assume mixes have no private information about the future value of bitcoins and therefore use its current market price in calculating the net present value.

¹⁹ This equivalence ignores the effects of compounding interest, though r and \bar{t} are both low enough that $(1 + r)^{\bar{t}} \sim 1 + r \cdot \bar{t}$.

therefore cannot verify that a rational mix will choose to stay in business at a given instant. Nevertheless, since absconding is a one-time action, she can gain a lot of comfort from the longevity of the mix.

5 Anonymity properties

Given the large number of negotiable parameters in Mixcoin, our anonymity analysis is theoretical. The actual anonymity guarantees will depend on how the Mixcoin ecosystem evolves. However, we can describe the threat model, quantify anonymity via a simulation under a simplified model, discuss optimal strategies for mixes and clients, enumerate some attacks and how to avoid them. We draw many connections to the extensive literature on mix networks for communication, dating to the initial proposal of communication mixes in 1981 [4]. That said, Mixcoin also differs from communication networks in important ways.

5.1 Threat model

Mixing literature typically analyzes the anonymity set of possible partners in a communications network. For unidirectional or message-based systems (as opposed to streams), the anonymity set may be considered separately for senders and receivers though the two often have symmetric properties. With Mixcoin the common case is a single linkable input address sending multiple chunks to distinct output addresses which never receive any other input. We focus on an attacker who wants to gain as much information as possible about the *anonymity set* of possible pre-mixing input addresses which may have been the source of the funds held by a final output address κ^N .

Because the Bitcoin blockchain is a permanent, public record of all transactions, every attacker is trivially a *global passive adversary*, a common attack model studied in communication mix networks²⁰ Communication mixes are also typically designed to withstand some number of malicious mixes, which are also possible in Mixcoin.

Mixing literature also considers extended attacker capabilities, such as the ability to compromise one or more mixes, delay or block messages from being sent, replay old messages, or flood the network with dummy messages [23]. Replay should be impossible in Mixcoin due to the double spending prevention in Bitcoin. Flooding is relatively easy and we should assume attackers have this capability. Delaying Bitcoin transactions is possible, but not trivial. Any miner can refuse to include transactions in blocks they mine, but this is only as effective as the amount of mining power the attacker can control. An attacker might also try to flood miners with larger value transactions, but there is no way to do so selectively and this would thus amount to a denial of service attack on all of Bitcoin. Assuming an attacker doesn't control a substantial majority of the mining pool, blocking transactions indefinitely should be effectively impossible.

²⁰ Tor is notably not designed to withstand attack by a global passive adversary, as Tor relays provide no mixing of traffic [9].

5.2 The passive adversary’s view and mix anonymity

We assume a passive adversary can determine with high probability which Bitcoin transactions are mix traffic, given their tell-tale size v and their use of one-time addresses. It will be easy to trace the path of a chunk originating from Alice through a series of escrow addresses and ending when the chunk is sent to a non-escrow address (identifiable because it is used more than once or holds more than v coins). The chunk’s mixing path will alternate addresses belonging to (probably) different clients and (probably) different mixes.²¹ If all clients use a constant N rounds of mixing and mixes choose chunks to send randomly, then the number of addresses in a given chunk’s path will be exponentially distributed with a mean of N , ignoring chunks retained as mixing fees.

Here Mixcoin benefits greatly here from a fortuitous property with no parallel in communication mixes: unless Alice publishes her warranty, a passive adversary cannot link any of the addresses in a chunk’s path to the identity of the mixes or users involved! All addresses will appear indistinguishable in the blockchain. If all clients use a constant N rounds of mixing, then the number of rounds in a given chunk’s journey (ignoring mixing fees) is an exponential distribution with a mean of N . After just a single round of mixing Alice’s chunk can’t be distinguished by a passive adversary from any other chunk mixed at the same time t_1 . If different mixes have different mix delays or randomize their delays, then her anonymity set is even greater. Mix anonymity is also useful to protect mixes from subpoenas.

On the downside, the difficulty of observers to learn the identity of mixes involved in a transaction means there is no exact way to determine mix transaction volumes. We will discuss statistical estimation methods in Section C.5.

5.3 Mix delay

We leave it to the mix to select t_2 given t_1 and hence the period of time Alice’s funds will remain in escrow. Mixes will require that $t_2 \geq t_1 + w$ to protect against double spending. Picking the smallest possible $t_2 = t_1 + w$ allows Alice to afford more rounds of mixing in a given time period. But this also means that Alice’s anonymity set for the round consists only of other chunks that were mixed at time exactly t_1 . Assuming the mix chooses delays uniformly $\delta \in_R [w, \delta_{\max}]$,²² increasing δ_{\max} will increase Alice’s anonymity set size by a factor of $\delta_{\max} - w + 1$.

Assuming steady mix transaction volumes, with Q chunks being mixed in a given round. Then Alice’s anonymity set size is $Q \cdot (\delta_{\max} - w + 1)$, and this process takes on average $w + \frac{\delta_{\max} - w}{2}$ blocks. In other words, her the entropy of her anonymity set grows by $1 + \frac{\log_2 Q \cdot (\delta_{\max} - w + 1)}{1 + w + \frac{\delta_{\max} - w}{2}}$ per block. With $w = 6$ and $Q = 100$, this expression is maximized for $\delta_{\max} = 7$. In fact, the optimal value of

²¹ Even this alternation can be obfuscated by a client or a mix by sending a chunk to their own address before sending it along.

²² Non-uniform distributions such as an exponential distribution are possible, but they make it difficult to provide a firm bound on the delay as required by the warranty.

δ_{\max} turns out to be 7 for the broad range $98 \leq Q \leq 4096$. $\delta_{\max} = 7$ means that the mix should enforce a delay of either 6 or 7 blocks, with equal probability.

The above calculation is simplistic because Alice’s anonymity set after multiple rounds of mixing will very quickly reach the set of all other chunks that are being mixed contemporaneously, and so will not accrue $\log_2 c \cdot (\delta_{\max} - w + 1)$ bits of entropy with each round. This suggests that we may want bigger delays than calculated above. Heuristically, $\delta_{\max} = 2w$ may be a good choice.

Whatever the mix’s published policy is, Alice can easily verify that the random distribution of δ matches the mix’s policy since we expect that over time she will send a large number of chunks to the mix. The mix cannot selectively cheat Alice since clients are (naturally) anonymous. If the mix cheats and picks delays that don’t match the distribution specified in its policy, it is not a violation of the warranty (as long as the mix adheres to its maximum delay). However, individual clients can detect this cheating and take their business elsewhere.

Some mixes may “free load” and publish policies with $\delta_{\max} = w$: since other mixes still introduce a delay, a passive adversary can’t tell which is which. We expect the market to reach an equilibrium between these two types of mixes.

Finally, the mix should be careful to ensure that the choice of which escrowed chunk to return to Alice does not leak information about the link between κ_{in} and κ_{out} . The safest strategy is to pick a random chunk from the set of escrowed chunks.

5.4 Active adversaries and mix deanonymization

An active attacker can (probabilistically) determine which mixes Alice is using. Observe that when Alice sends a chunk from κ_{in} to M via κ_{esc} , the client who ultimately receives this chunk will learn that κ_{in} interacted with M . Similarly, the client who sends a chunk to κ'_{esc} which is eventually sent to κ_{out} will learn that Alice interacted with M . The adversary can exploit this to launch an active attack: by sending numerous chunks to various mixes, he can identify two other addresses interacting with the same mix from each of those mix transactions.

This is a very strong attack model; to have a nontrivial chance of being able to learn the mix identity for a random mixcoin transaction, the adversary must generate a significant fraction of all Mixcoin transactions, and incur the concomitant mixing fees. Nevertheless, even against an implausibly strong attacker who can link every escrow address to its originating mix, the anonymity set still snowballs quickly with each mixing round as in traditional communication mixes. We demonstrate this via simulation in Appendix D.

5.5 Mixing multiple chunks

So far we have considered each chunk individually. However, if Alice combines many mixed chunks to make a payment, her anonymity set will be reduced to the intersection of the anonymity sets for the chunks. As long as she mixed those chunks at the same time, then those chunks will have the same anonymity sets,

and her payment is still unlinkable. If she didn't, it potentially allows an attack that we discuss in the next section.

Another issue is that if even one of the chunks travels through a path consisting (entirely) of compromised mixes, Alice is linkable to her payment and completely loses anonymity. If each chunk is routed independently, then with say 25% of mixes compromised, there is a 2^{-20} chance of routing a chunk through a chain of 10 compromised mixes, which seems acceptably low. However this probability increases rapidly with a greater fraction of mixes compromised. One way to avoid this would be to randomly pick a set of mixes for each contemporaneous batch of funds, and use a random permutation of that set for each chunks in the batch. In Tor, there is a similar argument for why building circuits improves anonymity as opposed to routing each packet independently.

It seems tempting to simply use the N most popular mixes permanently, since using more popular mixes improves the anonymity set. However, using popular mixes improve anonymity only against an unrealistically powerful active adversary, and Against a passive adversary, the popularity of the mix does not matter much. popular mixes may be more attractive targets for the adversary to compromise leading to a *greater* probability of picking a completely compromised path. This question also impacts how many mixes we expect to see. If there is a strong preference for popular mixes, then the barrier to entry will be high and the first entrants will corner the market. If not, there will be a diversity of mixes. It is impossible to definitively answer this question, but our best guess is that there will be perhaps a dozen popular mixes followed by a long tail, and clients will use a combination of the two types in the paths they choose.

6 Conclusion

Despite significant interest in providing strong anonymity for Bitcoin, the design of a robust protocol with that can be deployed without modifications to Bitcoin has remained an open question. In this paper we described how Mixcoin meets these goals. Our key innovations are cryptographic accountability, randomized mixing fees, and an adaptation of mix networks to Bitcoin. We look forward to engaging with the academic community and the Bitcoin community to further refine the design and to progress toward implementation and deployment.

Acknowledgements. We thank the several people who read drafts of this work and contributed suggestions, especially Ed Felten and Aaron Johnson, as well as the anonymous reviewers for their invaluable insights.

References

1. Nist randomness beacon. http://www.nist.gov/itl/csd/ct/nist_beacon.cfm
2. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating User Privacy in Bitcoin. In: Proceedings of Financial Cryptography (2013)
3. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology: Proceedings of Crypto. vol. 82, pp. 199–203 (1982)
4. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2), 84–90 (1981)
5. Clark, J., Hengartner, U.: On the use of financial data as a random beacon. *USENIX EVT/WOTE* (2010)
6. Coutu, O.: Decentralized mixers in Bitcoin. In: Bitcoin Conference (2013)
7. Danezis, G., Fournet, C., Kohlweiss, M., Parno, B.: Pinocchio coin: Building zero-coin from a succinct pairing-based proof system. In: Proceedings of the First ACM Workshop on Language Support for Privacy-enhancing Technologies. pp. 27–30. *PETShop '13*, ACM, New York, NY, USA (2013), <http://doi.acm.org/10.1145/2517872.2517878>
8. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: *Privacy Enhancing Technologies*. pp. 54–68. Springer (2003)
9. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. Tech. rep., DTIC Document (2004)
10. Golle, P.: Reputable mix networks. In: *PET* (2004)
11. Greenberg, A.: Darkwallet aims to be the anarchist's bitcoin app of choice. <http://www.forbes.com/sites/andygreenberg/2013/10/31/darkwallet-aims-to-be-the-anarchists-bitcoin-app-of-choice/> (2013)
12. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world. <https://bitcointalk.org/index.php?topic=279249.0> (August 2013)
13. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 127–140. ACM (2013)
14. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous Distributed E-Cash from Bitcoin. *IEEE Symposium on Security and Privacy* (2013)
15. Möser, M.: Anonymity of bitcoin transactions: An analysis of mixing services. In: *Proceedings of Münster Bitcoin Conference* (2013)
16. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
17. Raymond, J.: Traffic analysis: protocols, attacks, design issues, and open problems. In: *PET* (2001)
18. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: *Security and Privacy in Social Networks*, pp. 197–223. Springer (2013)
19. Rivest, R.: Electronic lottery tickets as micropayments. In: *FC* (1997)
20. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. *IACR Cryptology ePrint Archive* 2012, 584 (2012)
21. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: *EUROCRYPT*. pp. 393–403 (1995)
22. Sander, T.: Efficient accumulators without trapdoor. In: *In Second International Conference on Information and Communication Security ICICS 99*. pp. 252–262. Springer (1999)
23. Serjantov, A., Dingledine, R., Syverson, P.: From a trickle to a flood: Active attacks on several mix types. In: *Information Hiding*. pp. 36–52. Springer (2003)
24. Todd, P.: Fidelity bonds. https://en.bitcoin.it/wiki/Fidelity_bonds (2013)

A Manipulating the common reference string generated by the blockchain

In Section 3.6, we assumed that values in the blockchain are unpredictable for both Alice and the mix, and therefore can be safely used to determine which transactions the mix is able to retain as a fee. In reality, the mix might also control significant mining resources and therefore attempt to influence the value of blocks in the chain to collect a higher effective mixing fee rate than the negotiated rate ρ .

This attack faces two major difficulties. First, it is very costly to influence blocks. Because the value of block hashes is the output of a proof-of-work scheme, the only effective means for influencing their value is to discard some valid blocks which M finds in hopes of finding an alternate block which increases the number of mixing fees M collects. Discarding otherwise valid blocks makes the attack very expensive, given the high value of mined blocks (currently over US\$2,500) and expense of computing them.

Second, even given the ability to discard some blocks in the hope of finding a block which allows M to collect a higher mixing fee than usual, it is difficult to find a block which will significantly increase the number of mixing fees retained. Assuming the mix is holding Q transactions and is able to choose from a small number n of potential blocks, each potential block enables the mix to retain an independently random subset of the Q transactions with probability ρ each. The number of transactions retained is therefore a multinomial distribution with expected value $\rho \cdot Q$ and variance $Q(1 - \rho)\rho$. Under reasonable assumptions for Q and ρ , for example $Q = 100$ and $\rho = 2\%$, even generating 1000 valid blocks (which would be absurdly expensive computationally) produces less than a 40% chance of finding a block which even double's the mixes fee rate.

B Side channels

Despite the strong anonymity properties described in the previous section, there are side channels that may compromise her anonymity. In this section we discuss protocol-level side channels and defenses against them.²³

The most obvious side channel is payment amounts. If Alice receives a very specific amount of Bitcoins at her long-term address, is observed mixing them, and a day later an equal quantity of mixed chunks are combined to make a payment, the adversary might plausibly infer that Alice made the payment. This problem is not specific to Mixcoin. To mitigate it, we suggest that clients interested in unlinkable transactions mix their funds as soon as they receive them. This works as long as Alice makes payments that are subsets of her mixed assets and not her entire balance of mixed assets.

Recall that a chunk's anonymity set is not the set of all other chunks in Mixcoin, but rather the chunks that were mixed contemporaneously. This can

²³ Network-level side channels are out of scope. As noted earlier, we assume that Mixcoin clients always communicate using a secure anonymity network such as Tor.

lead to the following subtle side channel. It is based on the fact that each mixed chunk bears a timestamp of when it was last mixed. This is, of course, an inherent property of Bitcoin.

Suppose that Alice received three large, equal-sized quantities of income (of say 1,000 BTC) on three specific dates, each of which she mixed immediately upon receiving. Assume further that this is known to Eve (for instance, Eve might be the one paying Alice a regular salary). Since 10 rounds of mixing completes in a few hours in our system, each of Alice's mixed chunks will bear a timestamp that is very close to the time she received her payment. Finally, assume that Alice's policy for making payments is to pick a random set of mixed chunks. This seems sensible, since it does not require keeping state on the provenance of chunks.

Now, say Eve observes Bob receiving a payment of 1,000 BTC worth of mixed chunks, and would like to discover who made this payment. Eve observes the provenance of the chunks, and notices that they fall into three roughly equal-sized categories, with the set of timestamps known to be associated with Alice. This gives her strong evidence that the payer is Alice, which may be sufficient for deanonymization in practice.

Admittedly, this is a strong attack model. The adversary must have considerable insight into Alice's financials, and it only applies for large payments. Nevertheless, when these conditions to hold, the attack can be very effective. So let us consider some mitigation techniques. Which technique(s) are applicable depends on the user.

First, Alice could make each payment using chunks of the same provenance (*i.e.*, chunks that were mixed contemporaneously). This works if the payments are small enough. Second, Alice could re-mix all of her chunks every time she receives income. This destroys the timing information. Naturally, re-mixing incurs additional mixing fees. The above two methods can easily be combined. For example, at the end of each month Alice could mix her incomes from the previous month. When she needs to make a payment she will pick from a specific month's funds. These techniques require Alice to keep track of the provenance of her mixed chunks.

Third, if Alice has advance notice (a few hours) of needing to make a payment, there is a simple strategy that we call *input/output mixing* that will completely stop this side channel. Alice mixes her funds as soon as she receives income. When she needs to make a payment, she mixes a set of (already mixed) chunks totalling to the amount she owes. It introduces a delay in payment equivalent to mixing time, which is why Alice must have advance notice. If she did only output mixing, the payment amount could be used to link it to her.

Finally, we propose *continual mixing*, which does not require advance notice of payments. In addition to avoiding the timing side channel, it actually increases Alice's anonymity set. The core idea is that Alice continues mixing her coins until she is ready to spend them, but at a greatly reduced rate (*e.g.*, one round per month). Let Δ_A be a time period such that Alice is prepared to keep her coins for time Δ_A between receiving them and spending them. Then the continual

mixing algorithm for a chunk c for which initial mixing completes at time t_0 is as follows:

- generate $\Delta_{A,c} = U[0, \Delta_A]$
- mix c at time $\Delta_{A,c}$ and thereafter at Δ_A intervals
- mark c as spendable after the first continual mix round

It is easy to verify that regardless of the timings of the payments received by Alice, the distribution last mixing times for each of her spendable chunks is always $U[0, \Delta_A]$. This nullifies the timing channel, except for the matter of picking Δ_A . If Alice makes a payment with a random subset of her spendable chunks, Eve can infer Δ_A with high accuracy.²⁴

Picking Δ involves a trade-off. From the point of view of a business, if Δ is too high, it adds latency to the operating cycle and decreases cash flow. If Δ is too low, it leads to a higher depreciation rate of long-term assets due to the mixing fees incurred by continual mixing. Further, clients must consider each others' choices in picking Δ , since anonymity loves company and highly unusual values of Δ will help Eve.

Given these constraints, we propose several globally fixed values of Δ : for instance, a day, a week, a month, and a quarter; each client is free to pick the value that best suits their operating patterns. Alice can now expect her anonymity set to be the set of all Mixcoin clients who have the same value of Δ .

We must point out that some inference attacks cannot be prevented by any mixing system. For example, if Alice owes Bob a highly unique amount of money, and neither Alice nor Bob transacts with any other users, this information is sufficient to link Alice's outflow with Bob's inflow. Unlikely as such situations are for most users in the real world, they pose a problem for analysis of anonymity of our system. We are currently working on a formal model of anonymity that allows quantifying side-channel leaks; we do not attempt such a quantification in this paper.

C Improving mix trustworthiness

If a mix cheats, the cheated client can ensure that the mix gets a poor reputation. But how can a mix build a reputation for trustworthiness? Even if there are no theft reports against it, it might simply be because the mix doesn't have much volume yet. Further, to the extent that more popular mixes offer more anonymity (Section 5), clients would like to estimate mix transaction volumes.

In this section we discuss ways to better measure, as well as prove, mix trustworthiness, and even a mechanism for recourse against cheating mixes. These are all "out-of-band" and do not require modifications to the Mixcoin protocol.

C.1 External reputation

While some mix operators may choose to be anonymous, others may be comfortable revealing their real-world identity. A bank or trusted community member could leverage their external reputation to increase trust in their mix service.

²⁴ Describe the best way for Eve to infer Δ_A .

C.2 Throttling

Throttling, or rate limiting by the client, lets Alice limit her exposure to a given mix at any given time. If Alice wants her maximum exposure to M to be E , she transacts with M at the average rate of $\frac{E}{\delta_{A,M}}$ per block, where $\delta_{A,M}$ is the maximum mix delay that she picks for M . If she stops transacting with M as soon as she detects misbehavior, then M can steal at most E of her coins.

C.3 Aggregating theft reports

Essential to the success of Mixcoin is an external mechanism for client software to report theft by mixes. These reports will be aggregated and re-distributed to clients. Since it is impossible to falsely allege theft, this mechanism need not be trusted. Further, this aggregator could combine theft report data with estimates of mix volumes to publish *mix reputation lists* which are probability distributions over mixes. Estimating mix volumes, however, is tricky, so let's turn to that next.

C.4 User reports

To estimate volume, client users could publish through out-of-band channels, such as forums, logs containing aggregate statistics about their usage of various mixes (*e.g.*, "Alice mixed 10,000 chunks through mix M_1 in August"). If these are reputable members of the community (for example, with longstanding active accounts), observers can be reasonably confident that they are not sybils. Such reports provide lower bounds on mix volume.

C.5 Mark and recapture

The mark-and-recapture method for estimating wildlife populations could be used to estimate a mix's escrow reserves and hence its volume. The method involves engaging the mix in n transactions over a short period, and observing what fraction of these get forwarded among the set of corresponding return transactions. If the transaction volume of the mix is Q , then at any time the escrow pool contains Q transactions, and the expected number of corresponding returns is approximately n/Q when n is much smaller than Q . The mix may attempt to inflate this measurement by simulating transactions of sybil clients and contributing its own funds to the escrow pool. To defeat sybil detection by transacting with other mixes would incur fees proportional to the inflated volume. Thus, to inflate the apparent volume to twice the actual amount, the mix would have to forego its entire profits.

A more rigorous version of this is for a reputable entity, such as Consumer Reports, to perform a survey of publicly known mixes by periodically carrying out small transactions and report the results. The success of this approach depends on the (network-level) anonymity of the surveyor's actions.²⁵

²⁵ As an aside, anonymity is an issue for survey organizations in physical-world transactions as well. Consumer reports had a policy of purchasing vehicles with cash to

C.6 Fidelity bonds

As we’ve seen earlier, once a mix is in “steady state,” it is rational to stay in business. But how can a mix “bootstrap” trust before getting to this steady state?

The challenge faced by a new, unknown mix service in establishing trust has many parallels in business and in life. In fact, the earliest modern banks faced the issue of getting clients to entrust their money. How could clients be sure the bank wouldn’t disappear overnight? The banks solved this problem through signaling — they built huge, expensive buildings with gleaming facades as a way of showing that they were “in it for the long haul.”²⁶

Effort and money expended by the operator of a new mix in advertising the service could serve as such a costly signal. But Bitcoin also has a mechanism for explicit signaling: *fidelity bonds* [24]. Somewhat different from real-world fidelity bonds, Bitcoin fidelity bonds are a protocol for the owner of a Bitcoin address to sacrifice coins, *i.e.*, make them provably unspendable. We envision that mixes might optionally put up a fidelity bond to gain trust initially.

One possible game-theoretic explanation for why signaling by banks worked, and why fidelity bonds by mixes might work, involves *asymmetric information*. Let’s say a business (a bank or mix) has done the research and concluded that the market is ripe for a new entrant. It therefore sets up shop with honest intentions. But consider clients who typically do not have access to this market research or other reasons for the business’s belief in its long-term viability. To such a client, the business may very well have the intention of never reaching the steady state, but instead absconding as soon as a few tentative clients entrust them with money money. By undertaking a costly action, the business signals its belief that it can be viable. Since the sunk cost exceeds what the business might gain by absconding before reaching the steady state, a business that does not believe in its own viability will not undertake the costly action. Via a Bayesian argument, the client can infer that the business does not intend to abscond.

D Growth in anonymity against a strong attacker

As claimed in Section 5.4, even against an attacker able to identify which mixes correspond to escrow addresses in the block chain the probability distribution over the anonymity set of a chunk rapidly approaches the uniform distribution after multiple rounds of mixing. Assume that there are m mixes, and Q chunks are being mixed in N concurrent, synchronous rounds. In other words, in each round, each chunk is assigned to a given mix, and its anonymity set for that round is the set of all other chunks assigned to that mix in that round. We can quantify this by measuring the statistical distance (L_1 distance) from the uniform distribution. Figure 1 summarizes the results of simulation.

protect their identity, but had to abandon the practice when car dealers realized that they were the only entity to pay with cash.

²⁶ Other examples of costly actions undertaken for the purpose of signaling commitment are gang initiations and engagement rings.

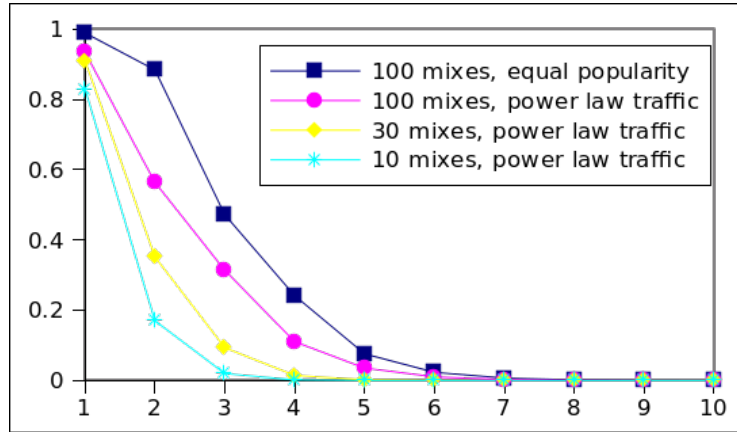


Fig. 1. Simulation. Statistical distance (L_1 distance) between PDF of a chunk’s anonymity set and the uniform distribution, as a function of the number of rounds. $Q = 1000$ chunks in all experiments.

The first line shows $m = 100$ mixes, with each mix being equally popular (*i.e.*, in each round, each chunk is independently equally likely to be sent to each mix). We observe that the statistical distance of the PDF from uniform drops exponentially with the number of rounds. After 7 rounds it is under 0.1, and after 10 rounds it is about $2.4 \cdot 10^{-4}$.

The other lines show the situation when the mix traffic is skewed according to a power law distribution, *i.e.*, in each round, each chunk is independently sent to mix i with probability proportional to $\frac{1}{i}$. Here the distribution converges to uniform even faster. Decreasing the number of mixes has a similar effect. These are both illustrations of the fact that anonymity loves company. The *degree of anonymity* [8] converges to 1 similarly. With 100 mixes with power-law traffic, after 4 rounds it exceeds 0.99 and after 10 rounds it is about $1 - 2 \cdot 10^{-9}$.

While this is a simplified model, we can infer a lot about the anonymity properties of Mixcoin. Qualitatively, after a few rounds Alice’s chunk is uniformly mixed with all the other chunks that are being mixed contemporaneously. The chunk’s anonymity set does not include chunks that were mixed at a different time, for example a month ago. If the attacker has compromised some of the mixes, as long as some of the rounds of mixing (say N') went through honest mixes, the anonymity properties are identical to using N' rounds of mixing with all honest mixes.