Obscuro: A Bitcoin Mixer using Trusted Execution Environments

Muoi Tran[1], Loi Luu[2], Min Suk Kang[1], Iddo Bentov[3], and Prateek Saxena[1]

[1] National University of Singapore
[2] Kyber Network
[3] Cornell University

Abstract. Bitcoin provides only pseudo-anonymous transactions, which can be exploited to link payers and payees – defeating the goal of anonymous payments. To thwart such attacks, several Bitcoin mixers have been proposed, with the objective of providing unlinkability between payers and payees. However, existing Bitcoin mixers can be regarded as either insecure or inefficient.
We present Obscuro, a highly efficient and secure Bitcoin mixer that utilizes *trusted execution environments* (TEEs). With the TEE's confidentiality and integrity guarantees for code and data, our mixer design ensures the correct mixing operations and the protection of sensitive data (i.e., private keys and mixing logs), ruling out coin theft and address linking attacks by a malicious service provider. Yet, the TEE-based implementation does not prevent the manipulation of inputs (e.g., deposit submissions, blockchain feeds) to the mixer, hence Obscuro is designed to overcome such limitations: it (1) offers an indirect deposit mechanism to prevent a malicious service provider from rejecting benign user deposits; and (2) scrutinizes blockchain feeds to prevent deposits from being mixed more than once (thus degrading anonymity) while being eclipsed from the main blockchain branch. In addition, Obscuro provides several unique anonymity features (e.g., minimum mixing set size guarantee, resistant to dropping user deposits) that are not available in existing centralized and decentralized mixers.
Our prototype of Obscuro is built using Intel SGX and we demonstrate its effectiveness in Bitcoin Testnet. Our implementation mixes 1000 inputs in just 6.49 seconds, which vastly outperforms all of the existing decentralized mixers.

1 Introduction

Bitcoin is the first widely-adopted cryptocurrency that allows users to transact digital coins without relying on any centralized, trusted party [36]. It allows users to have *pseudonymous* identities called *Bitcoin addresses*, which are generated from their public keys. Users send coins by creating transactions that include one or more *inputs* (i.e., references to outputs of previous transactions) and *outputs* (i.e., addresses and credits that go to these addresses).

Although each Bitcoin user can stay *pseudo-anonymous* by generating multiple cryptographic addresses that are used to receive funds, the users' transaction records are publicly available on the blockchain. In other words, Bitcoin's pseudo-anonymity can be regarded as publishing everyone's credit card statements, with the names redacted but the "account numbers" (i.e., cryptographic addresses) and payment amounts visible. Since all the Bitcoin transactions can always be linked (i.e., referenced) back to one or more previous transactions, the privacy of Bitcoin users can be violated by an adversary who is able to track the flow of bitcoins being transferred and further cluster Bitcoin addresses together [7, 29, 30, 35, 38, 40, 49].

Transaction traceability may be useful under some circumstances (e.g., tracking criminal activities); however, Bitcoin allows *anyone* to examine users transaction history (e.g., their incomes and spending habits), which may lead to large-scale privacy invasion or surveillance. Furthermore, Bitcoin's traceability is considered to be harmful to its fungibility because the coins in circulation can be considered tainted to certain degrees (see [39] for a well-known legal case).

Improving Bitcoin anonymity using mixers. In the past few years, there have been several proposals that aim to provide better privacy for Bitcoin users using Bitcoin *mixers*. These mixers take coins from multiple senders and output the coins to determined recipients in a shuffled order. Since all the recipients have the equal probability of being transacting with a given sender, it is difficult to identify which sending address is actually linked to which receiving address.

Unfortunately, existing Bitcoin mixers are either known to be *vulnerable* to a number of attacks that can be launched by malicious mixer service providers, or *inefficient* due to prohibitive algorithmic/communication overhead for large (e.g. hundreds or thousands) mixing set sizes. In general, some mixers require users to send coins to a centralized service and thus a malicious service provider can leak the links between senders and recipients, steal the coins or subtly reject some honest users from participating to reduce anonymity set (see Section 7 for how Mixcoin [14], BlindCoin [51] and TumbleBit [18] are vulnerable to these attacks). Alternatively, there are several decentralized Bitcoin mixing protocols that operate with individual peers [12,26,42,43,57]. However, decentralized mixers demonstrate limited scalability (e.g., only mix among 50 peers [26, 42, 43]), suffer from long waiting times for finding other mixing parties [12], or assume an unrealistic fraction of honest mixing parties (e.g., 2/3 of parties are honest [57]).

Our contributions. As a new design point in the space of Bitcoin mixers, we propose a centralized mixing system called OBSCURO, which can swiftly mix a thousand users while offering strong security and anonymity guarantees against malicious mixer service providers, which have not been achieved by any existing mixers.

OBSCURO utilizes modern hardware-based *trusted execution environments* (TEEs) to protect its mixing operations from a potentially malicious mixer service provider.[4] Specifically, OBSCURO (1) *isolates* its execution in a special memory region and prevents a malicious service provider from stealing users' deposits or leaking transaction links; (2) allows users to *verify* the correct mixing operations before they deposit their coins; and (3) maintains a TEE-based *simple* protocol architecture so that its mixing set size is limited only by the inherent Bitcoin block size. OBSCURO has a generic design that is compatible with various trusted execution environment techniques.

Moreover, OBSCURO addresses a new family of attacks that aim to weaken the anonymity guarantees of the mixer. Specifically, a malicious service provider may *block* connections between some users and the mixing service to effectively reduce the anonymity set. Also, a malicious provider can manipulate the Bitcoin blocks fed to the mixer and create blockchain *forks* to make the anonymity set reduction attack invisible to users; see Section 2.2 for detailed attack strategies. To handle these attacks, OBSCURO removes all direct network interactions between users and the mixer platform and guarantees benign users' participation. Also, OBSCURO employs a stateless design and malicious blockchain fork detection.

We implement OBSCURO using a recent trusted computing capability called Intel SGX [6,27]. The OBSCURO prototype demonstrates its effectiveness in terms of mixing times for various sizes of anonymity sets on both Bitcoin Regtest and Bitcoin Testnet. For example, OBSCURO takes only 6.49 seconds to mix 1000 transactions, showing that OBSCURO is efficient and ready to be deployed in practice.

2 Problem Definition

2.1 Preliminaries

Cryptography primitives. We denote $\pi(x_1, x_2, \cdots, x_n) \leftarrow (x_1, x_2, \cdots, x_n)$ as a permutation function which returns a random permutation of set $(x_1, x_2, \cdots, x_n)$. We utilize a digital signature DS consisting of three algorithms as follow: $(\mathsf{addr}, \mathsf{sk}^{\mathsf{DS}}) \leftarrow \mathsf{DS.GenKey}()$ is a key generation algorithm which generates a public key (correspond to a Bitcoin address addr) and a secret key $\mathsf{sk}^{\mathsf{DS}}$ associates with it, $\sigma \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, \mathsf{m})$ is an algorithm which signs a message m using the secret key $\mathsf{sk}^{\mathsf{DS}}$, and $\{0,1\} \leftarrow \mathsf{DS.Verify}(\mathsf{addr}, \mathsf{m}, \sigma)$ can be used to verify if the signature σ is correct on message m associated with the address addr. We also include public key encryption PE which includes three following algorithms: $(\mathsf{pk}, \mathsf{sk}^{\mathsf{PE}}) \leftarrow \mathsf{PE.GenKey}()$ generates a public and secret key pair, $\delta \leftarrow \mathsf{PE.Enc}(\mathsf{pk}, \mathsf{m})$ denotes the encrypting message m using public key pk, and $\mathsf{m} \leftarrow \mathsf{PE.Dec}(\delta, \mathsf{sk}^{\mathsf{PE}})$ returns message m in plaintext after decrypting δ using secret key $\mathsf{sk}^{\mathsf{PE}}$.
Mixing operations. Let us call a sender Alice and a recipient Bob. Alice, with the address $\mathsf{addr_A}$ participates in a mixing round by sending a denomination of deposit, along with Bob's withdrawal address $\mathsf{addr_B}$, to the mixer. After

[4] For simplicity, we assume that a mixer service provider runs its mixer software on its local machine. If a service provider outsources the mixer software to a cloud infrastructure, our threat model includes the infrastructure as well.

receiving n deposit transactions $(\mathsf{tx}_1, \mathsf{tx}_2, \cdots, \mathsf{tx}_n)$ from senders with addresses $(\mathsf{addr}_{\mathsf{A}_1}, \mathsf{addr}_{\mathsf{A}_2}, \cdots, \mathsf{addr}_{\mathsf{A}_n})$, the mixer permutes the corresponding recipient addresses, i.e., $\pi(\mathsf{addr}_{\mathsf{B}_1}, \mathsf{addr}_{\mathsf{B}_2}, \cdots, \mathsf{addr}_{\mathsf{B}_n})$ and returns coins to these addresses in that order. The final transaction TX written into the public blockchain contains the list of deposit transactions $(\mathsf{tx}_1, \mathsf{tx}_2, \cdots, \mathsf{tx}_n)$ as inputs and the shuffled recipient address list $\pi(\mathsf{addr}_{\mathsf{B}_1}, \mathsf{addr}_{\mathsf{B}_2}, \cdots, \mathsf{addr}_{\mathsf{B}_n})$ as outputs.
Privacy Definition. The purpose of a mixer is to provide *relationship anonymity* between any sender and recipient pair, as we define as follows [37]:

Definition 1. Relationship anonymity *of a sender address* addr_A *and a recipient address* addr_B *means that an outsider (i.e., being neither the sender nor the recipient) cannot distinguish whether the owners of these addresses are transacting through the mixer or not. In other words,* addr_A *and* addr_B *are unlinkable.*

2.2 Threat Model

We consider a strong adversary (e.g., a malicious mixing service provider) that controls the mixer operations as well as the underlying privileged software running on the hosting platform (e.g., operating system (OS)). Since the adversary can access any system resource of the mixer, she can actively modify or drop any message received by or sent from the mixer platform. Moreover, the adversary can observe all the transactions on the public blockchain. We also assume the adversary can make deposits under her control (or Sybil deposits).

We consider three types of attacks against Bitcoin mixers:
Coin stealing attacks. An adversary steals user-submitted coins. The adversary may trick participants to submit coins to adversary's address or obtain the secret key sk_M associated with the mixer's address addr_M.
Availability attacks. An adversary prevents some or all participants from using a mixing service. The adversary may disrupt a sender's access to the mixer or the mixer's access to the blockchain (i.e., the mixer cannot return coins to the intended recipient).
Anonymity attacks. An adversary aims to identify the recipient address addr_B of a benign recipient Bob who is transacting with a begin sender Alice (who owns addr_A) through a mixer, i.e., break the relationship anonymity between addr_B and addr_A (see Definition 1). When the adversary has access to the permutation π, de-anonymization is trivial. Without access to π, the adversary has to guess the correct recipient address within *the anonymity set*—i.e., the set of all benign recipient addresses.[5] Here, we present two attack strategies to *reduce* the anonymity set size:

1. *Participation rejection.* The adversary rejects deposits from any arbitrary benign senders to perform a mixing operation only with a few targeted benign deposits and some adversary-created deposits, effectively reducing the

[5] The anonymity set is defined over the recipient addresses of a mixing round. The same for the sender addresses can be defined and their set sizes are equivalent for a given mixing round.

anonymity set. In the extreme case, only one benign deposit can be included and its sender/recipient addresses are immediately linked.

2. *Blockchain forking.* By allowing indirect deposit submission via the Bitcoin blockchain (see Section 3.3 for details), Obscuro prevents anonymity set reduction in the valid blockchain; however, adversaries may still reduce anonymity set in a *stale* chain. To be specific, the adversary can feed the two different blockchain feeds—a stale chain and valid chain—to the mixer to have it mix twice with the two feeds. The stale chain is an adversary-generated one that contains only selected user deposits (thus reducing anonymity set) whereas the valid chain has all the user-submitted deposits. This attack makes the anonymity set reduction invisible to users. Two attack strategies are available: (1) the adversary can directly tamper with the blockchain data stored on the mixer platform, or (2) the adversary can feed a stale chain and a valid chain to the mixer sequentially so that the stale one becomes naturally abandoned thus invisible later.

2.3 Scope, Assumptions, and Limitations

Direct attacks against the confidentiality properties of TEE platforms are beyond the scope of this work. Particularly, we acknowledge that several side-channel attacks against Intel SGX have been discovered and also mitigations have been actively studied in the last few years [23, 33, 45–47, 53].

We assume that the mixer implementation has no malware or backdoor inserted by the mixer operator. The full implementation of the mixer (about additional 2.4K source lines of code (SLoC) to the trusted computing base; see Section 5.2) is open-sourced and available for public scrutiny. We leave a formal verification of our implementation for future work.

Similarly to all the centralized mixing services, our mixer proposal is susceptible to denial-of-service (DoS) attacks on the mixer's platform (though users will never lose their funds, due to a refund mechanism). Direct defenses to such attacks are out of scope.

We aim to provide anonymity for users in a single mixing round but do not offer special logic for better anonymity across multiple mixing rounds; e.g., intersection attacks [16, 17, 22, 32].

Moreover, we do not aim to address an exceedingly powerful adversary that can arbitrarily control the transactions of the public blockchain (e.g., via owning or colluding with a large fraction of miners).

3 Obscuro

3.1 Solution Overview

Obscuro maintains a simple centralized mixer architecture and protocol—that is, senders submit deposits to a single address of a mixer and then the funds are sent to the shuffled recipient addresses. Obscuro can be robust against all the three types of attacks while maintaining such simple mixing operations because it utilizes the confidentiality and integrity guarantees of trusted execution environments (TEE). Hardware-based TEEs (such as Intel SGX [6, 27], ARM TrustZone [8]) provide *isolated execution* (i.e., the mixer execution is isolated from all other operations in the platform including privileged software such as OS) and *remote attestation* (i.e., a third party can verify the correctness of the mixer's operations) properties [56]. The TEE-protected mixer design prevents adversaries from stealing participant's coins because users can verify the mixer's address $\mathsf{addr_M}$ via remote attestation and the secret key $\mathsf{sk_M}$ of the mixer is protected within TEE. Also, OBSCURO does not leak the permutation π because the shuffling operations are executed in the isolated TEE region and not accessible to adversaries. In addition, for higher availability, OBSCURO has a refund mechanism which is written in a simple script in the deposit transaction tx, allowing senders to claim back the sent coins when mixing service is being denied.

Although being powerful enough to rule out the coin stealing and availability attacks, TEE's confidentiality and integrity guarantees alone do not necessarily address the two anonymity reduction attacks (i.e., participation rejection and blockchain forking attacks) because these attacks involve the blockchain inputs manipulated *outside* of the TEE's protection. The participation rejection attack drops some user deposits before the TEE-based mixer reads them, and the blockchain forking attack presents an adversary-generated stale block to the mixer. First, to handle the participation rejection attacks, we propose an *indirect participation* mechanism, which enables any user to participate in a mixing round *without direct interaction* with the mixer. As we remove any direct interaction between users and the mixer, the adversary cannot reduce anonymity set by disrupting their interactions with the mixer. Second, OBSCURO avoids mixing user deposits twice in a stale chain and the valid chain by protecting the received blockchain data within TEE and ensuring any deposit transaction will contribute in at most one mixing round.

3.2 Obscuro Protocol

We describe the architecture of OBSCURO using Intel SGX [6, 27] capabilities in Figure 1. An application implemented in the SGX programming model includes two types of components: (1) trusted components that are loaded and executed inside an SGX *enclave*, a special memory region that is isolated from the untrusted functions including privileged software (e.g., OS), and (2) untrusted components that operate as a non-SGX application outside of the enclave boundary. While the code inside an enclave is able to read/write the application memory outside of the enclave as well as the enclave data in its unencrypted form, the non-enclave code (e.g., the OS) *cannot* access the enclave's memory. At a high level, OBSCURO protocol has four phases as follows:

1. *Bootstrapping and remote attestation.* OBSCURO starts its execution in TEE, generates keys, and publishes its identities (i.e., Bitcoin address and public key) so that users can verify them remotely.
2. *Participation via blockchain.* Users participate in a mixing round by submitting deposit transactions to the mixer's address. A transaction includes an encrypted recipient's address and a refund script.

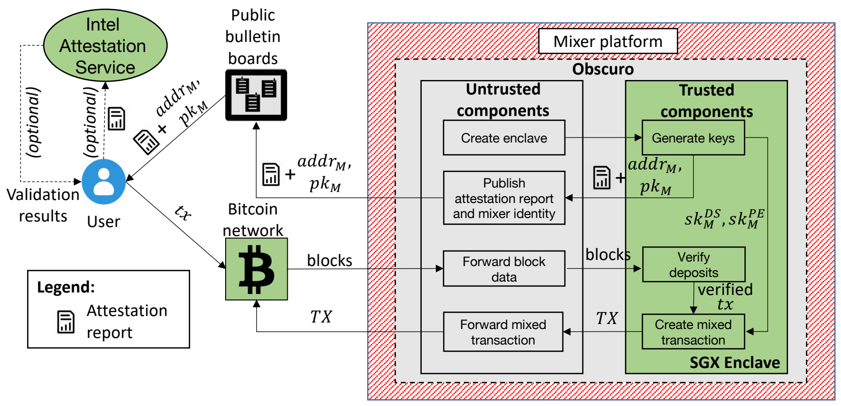


Fig. 1: The Obscuro architecture. Components in the green background are trusted, while the ones in the gray background are untrusted. The adversary controls components in the red stripe background.

3. *Deposit verification.* Obscuro downloads the blockchain data, extracts the deposit transactions, and decrypts the attached recipient addresses. Obscuro also checks if the collected deposits have been mixed in another blockchain fork to prevent blockchain tampering attacks.
4. *Mixing and returning coins.* Obscuro follows a set of rules to determine when a mixing round starts and then executes the mixing operations mentioned in Section 2.1.

We discuss the details of the first two phases, which realize our indirect participation mechanism against the participation rejection attacks, in Section 3.3. The mitigations for the blockchain forking attacks are presented in Section 3.4. Section 3.5 presents a set of parameters and rules for practical mixing operations of Obscuro.

3.3 Indirect Participation Mechanism

All-or-none availability for Obscuro's identity. We ensure that either *all or none* of the benign users can obtain the identity of Obscuro (i.e., $\mathsf{pk_M}$ and $\mathsf{addr_M}$) to participate a mixing round via the remote attestation process as follows.

1. An *initiator* first begins the remote attestation of the Obscuro instance. Ideally, any person or organization can be the initiator of a remote attestation.[6] When an Obscuro instance has been launched in a newly created SGX enclave, Obscuro involves $\mathsf{DS.GenKey}()$ and $\mathsf{PE.GenKey}()$ to generate fresh $(\mathsf{addr_M}, \mathsf{sk_M^{DS}})$ and $(\mathsf{pk_M}, \mathsf{sk_M^{PE}})$, respectively.
2. Next, the enclave provides an *attestation report*, which is cryptographically signed by the attestation key of the SGX hardware. The attestation report

[6] Intel suggests that the service provider of the SGX application to be the initiator [21].

contains the hash of the enclave's initial contents (i.e., the measurement of the application instance) and the hash of some *manifest data* computed inside the enclave. The manifest data includes the $\mathsf{pk_M}$ and $\mathsf{addr_M}$ and is sent along with the attestation report.

3. The attestation report and the manifest data are then distributed by the initiator through some *public bulletin boards* (e.g., IPFS [2], public blockchains).
4. Finally, users may forward the attestation report to the Intel Attestation Service (IAS) [21] to verify the report and the manifest data. Note, however, that the interaction with the IAS is *optional* for most of the users since any user or a third party can distribute the attestation validation result from the IAS so that users can verify the attestation report by themselves [19].

Deposit Submission via Blockchain. Obscuro allows users to participate in a mixing round indirectly by *embedding* the participation information to the deposit transaction written on the public blockchain. In particular, a recipient (say Bob) chooses the recipient address $\mathsf{addr_B}$, encrypts it with the $\mathsf{pk_M}$ of Obscuro (i.e., $\delta \leftarrow \mathsf{PE.Enc}(\mathsf{pk_M}, \mathsf{addr_B})$), and then forwards δ to a sender (say Alice). Next, Alice constructs the deposit transaction tx that follows the predefined format shown in the Appendix A. The encrypted address δ is included in the `OP_RETURN` field (which can carry up to 80 bytes) of the deposit transaction. The encryption must be *CCA secure* because the ciphertexts will be available on the blockchain and adversaries can modify and submit them to the oracle (i.e., the mixer); see Section 5 for details.

We also utilize the opcode `OP_CHECKLOCKTIMEVERIFY`, which allows the mixer to spend the deposit, and allows the user to claim back the deposit after a locked time (if it has not been spent yet). Alice then broadcasts the deposit transaction tx to the Bitcoin network.

3.4 Detection of Malicious Blockchain Forks

Obscuro detects malicious blockchain forks by employing the following design policies. First, Obscuro does *not* store blockchain data outside of the protected TEE, hence prevents an adversary from directly tampering with the blockchain data stored on the mixer platform. Note that sealed storage [6] cannot be used to store the blockchain data securely because the adversary can roll back the state of the sealed storage [25,50]. Second, to prevent an adversary from loading different blockchain data to different Obscuro instances, Obscuro generates and uses a new address to receive deposits for every instance (see Section 5.2), ensuring that each deposit is mixed on only one blockchain. Third, Obscuro checks the received blockchain data to see if: (1) the blockchain contains forks; (2) there is a deposit transaction appears on one or more forks; and (3) the deposit transaction has been mixed before. If all the conditions are met, we consider it as an attempt to tamper with the blockchain fed to Obscuro. Then Obscuro abandons its current address, generates new keys, and starts a new mixing round.

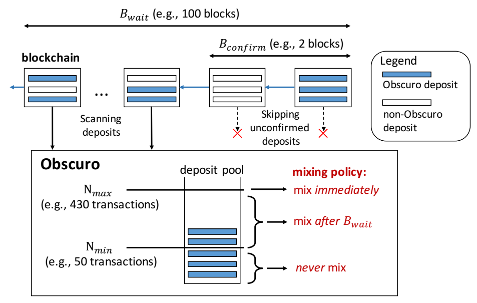

B_{wait} (e.g., 100 blocks)
$B_{confirm}$ (e.g., 2 blocks)
blockchain
Scanning deposits
Skipping unconfirmed deposits
Legend
Obscuro deposit
non-Obscuro deposit
Obscuro
deposit pool
N_{max} (e.g., 430 transactions)
N_{min} (e.g., 50 transactions)
mixing policy:
mix *immediately*
mix *after* B_{wait}
never mix

Fig. 2: Obscuro's mixing policy.

3.5 Collecting Deposits

To specify precisely how the deposits should be collected and when a mixing operation is executed, we introduce four system parameters: $\mathtt{N_{min}}$, $\mathtt{N_{max}}$, $\mathtt{B_{wait}}$, and $\mathtt{B_{confirm}}$. In Figure 2, we illustrate how these parameters are used to determine the mixing set for each round. In particular, Obscuro continuously monitors the blockchain and collects deposits from the maximum $\mathtt{B_{wait}}$ blocks (excluding the $\mathtt{B_{confirm}}$ most recent blocks) since its last mixing operation, to be maintained in the deposit pool. Obscuro decides whether to mix the deposits in the pool based on the size of the pool and the number of blocks since the last mixing operation as follows.

1. If there are less than $\mathtt{N_{min}}$ deposits in the pool after scanning $\mathtt{B_{wait}}$ blocks (since the last mix operation), no mixing is done and the deposits are refunded.
2. When the pool reaches $\mathtt{N_{max}}$ deposits, the mixer immediately starts mixing with the $\mathtt{N_{max}}$ deposits.
3. If there exist at least $\mathtt{N_{min}}$ deposits after reaching $\mathtt{B_{wait}}$ blocks since the last mix operation, the mixer performs the mix with all the available deposits.

The practical value of $\mathtt{B_{wait}}$ can be the locked time of the deposits (plus the few $\mathtt{B_{confirm}}$ blocks for confirmation), so that the user can get refunded immediately in the case that her deposit is not mixed. $\mathtt{N_{max}}$ denotes the maximum capacity of participants in a mixing round due to the underlying blockchain's constraints (e.g., maximum transaction size). Having a minimum mixing set size parameter $\mathtt{N_{min}}$ is desirable since it provides a lower bound on the quality of the mix. This assures the users an in-protocol guarantee regarding the mixing set size even *before* they send coins to the mixer, which is not supported by any centralized mixers. Since Obscuro may finish a mixing round within 2 blocks, we use the block confirmation parameter $\mathtt{B_{confirm}}$ to prevent it from strictly abandoning its address whenever there is an orphaned fork on the main blockchain.

4 Security Analysis

In this section, we present how Obscuro is secure against coin stealing, availability, and anonymity attacks described in our threat model (see Section 2.2). We assume that all the cryptographic primitives used are secure.

Coin stealing attacks. With the execution isolation guarantee of TEE, an adversary cannot access to the mixer's secret keys and thus cannot steal the user-submitted coins. Moreover, an untrusted service provider or bulletin board also cannot trick participants to deposit coins to a wrong address because any tampering with $\mathsf{pk_M}$, $\mathsf{addr_M}$, or the attestation report will lead to a failed verification by the IAS (see Section 3.3).

Availability attacks. Since the mixing operations are executed within the protected TEE, we ensure that Obscuro returns coins to the recipient addresses at the end of each mixing round. If the mixer transaction is not submitted to the Bitcoin network, the senders can get their coins back after a determined period thanks to the refund script in the deposit transaction. Thus, the user-submitted coins are always available to be withdrawn from Obscuro.

Anonymity attacks. We remark that the permutation is performed within a TEE which is not accessible to the adversary. However, with the capability of manipulating the inputs of the mixer, an adversary can reject some benign users by either (1) selectively disclosing the mixer's identity (i.e., $\mathsf{addr_M}$ and $\mathsf{pk_M}$) to only an arbitrary set of benign users; or (2) selectively accepting only some benign users' deposit submissions.[7] Our Obscuro design removes these adversary's capabilities. First, Obscuro's all-or-none availability mechanism makes it extremely hard for an adversary to selectively prevent benign users from learning the identity of the mixer unless she controls all the public bulletin boards the users can use, which is practically impossible. Second, Obscuro's blockchain-based deposit submission requires an adversary to own (or collude with) a significant portion of Bitcoin mining power to prevent arbitrary benign deposits from being accepted by Obscuro on the main blockchain, which is also impractical.

We now show that the blockchain forking attack is also ineffective. First, an adversary cannot directly tamper with the blockchain data that Obscuro is processing since Obscuro does not store it outside TEE and data within TEE is protected with integrity guarantee. If an adversary restarts Obscuro and feeds a blockchain to it, Obscuro will not be able to collect the deposits from that blockchain because its secret keys are destroyed when the previous execution is terminated. Similarly, because the Obscuro uses a new address to receive coins when it detects malicious blockchain forks, any deposit transaction will be used to mix at most once, regardless on a local branch or the main branch of the blockchain. In both attack attempts, the adversary may break the unlinkability of some senders and their recipients in the fake blockchain; however, the recipient addresses will never be used in the main blockchain. Instead, the affected users will get their deposits back via our refund mechanism.

[7] Note that rejecting individual users can be also considered as an availability attack; however, its ultimate attack goal is to reduce anonymity set.

5 Implementation and Evaluation

We implement a proof of concept of Obscuro on a commodity platform with the full Intel SGX support. Our evaluation of mixing large numbers of deposits shows that our SGX-based Bitcoin mixer is practical and incurs only negligible overhead.

5.1 Implementation

We utilize Panoply framework [48] to port Bitcoin Core's codebase [8] into an Intel SGX application. We use OpenSSL library to implement our public key encryption scheme PE, which is the Elliptic Curve Integrated Encryption Scheme (ECIES) over the `secp256k1` elliptic curve with AES counter mode and 16-byte HMAC tag. ECIES is CCA secure [9, 15] and its ciphertexts are quite compact (i.e., only 69 bytes) to fit in the `OP_RETURN` field. We also port the constant-time ECDSA library libsecp256k1 [52] into the enclave for the implementation of the digital signature DS due to its compatibility with current Bitcoin protocol. We implemented the shuffling function with the linear time complexity Fisher-Yates shuffle algorithm [13] using the trusted randomness generator.

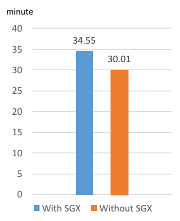
Because it is crucial to use a reliable randomness source for the cryptographic keys and the permutation function, we increase the entropy of our random seed in order to reduce the trust in the hardware provider (i.e., Intel in our current implementation). Thus, in addition to the trusted hardware-based randomness provided by the RDRAND to `sgx_read_rand`(), we concatenate extra sources of randomness: OS provided randomness, the SGX trusted clock and the latest block hash from the Bitcoin blockchain. To predict the random seed that we feed to the key generator, the adversary will need to control all the components that contribute to the seed.

5.2 Evaluation

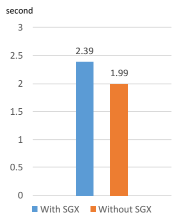
We evaluate Obscuro on a Dell Latitude E5570 laptop that is SGX-enabled with the 6th Generation Intel® Core™ i7-6820HQ CPU and 8GB of memory. We configure the laptop's BIOS to allocate 128 MB memory for each SGX enclave. We use the Linux 1.6 Open Source Beta version of Intel Software Guard Extensions SDK, Intel SGX Platform Software (PSW), and a driver on Ubuntu Desktop-14.04-LTS 64-bits with Linux kernel version 3.13. Obscuro is compiled with GCC v4.8 and built for SGX hardware pre-release mode `HW_PRERELEASE` with default optimization flags.

Here, we evaluate the performance of several steps in Obscuro protocol. We also measure the overhead caused by SGX operations by comparing two versions of Obscuro, with and without SGX equipped. All experiments are done 20 times in Bitcoin Regression Testing environment. Furthermore, we also measure the transaction fees via an on-chain evaluation. Finally, we describe the trusted computing base of Obscuro implementation.

[8] Version v0.13.1: `https://bitcoin.org/en/release/v0.13.1`



(a) Fetching and verifying 200,000 blocks.



(b) Scanning transactions in 100 blocks.

Fig. 3: Measured time for fetching blocks and scanning transactions with-/without SGX.

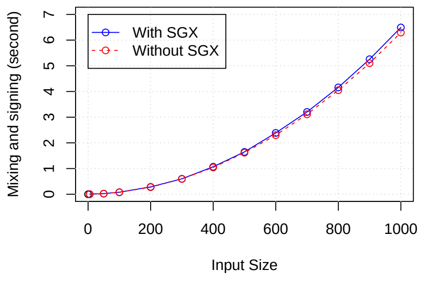


Fig. 4: Obscuro signing and mixing time with/without SGX.

Bootstrapping Bitcoin blockchain. We measure the time taken to fetch and verify the Bitcoin blockchain since the latest checkpoint. As of this writing, the latest block is approximately 200,000 blocks ahead of the most recent blockchain checkpoint.[9] Figure 3a shows that it takes approximately 35 minutes. This is easily acceptable in practice since the bootstrapping happens only once when an Obscuro instance is launched.

Scanning for Obscuro deposits. We measure the time taken to scan Bitcoin blocks and find valid Obscuro deposits, which also involves ECIES decryption operations. In particular, we assume a conservative scenario in which Obscuro should scan 100 blocks to search for 2000 Obscuro deposits among a total of 4000 transactions. Figure 3b shows that our implementation scans the deposits very fast (2.39 seconds with SGX) and the overhead incurred by the use of SGX is acceptable (only 0.4 seconds).

Mixing and Signing transactions. We measure the running time of the mixing and signing operations and show that Obscuro is scalable and efficient in mixing a large set of transactions and operating Obscuro with trusted hardware causes negligible overhead. We test the shuffling and transaction signing with different sizes of the mixing set, ranging from 5 to 1000 transactions. Figure 4 shows that the operation time increases as the size of the mixing set increases and SGX programming model causes a very small extra execution time (approximately $3\% - 5\%$). Furthermore, Obscuro can mix one thousand input transactions within seconds (specifically, 1000 inputs in 6.49 seconds). This means that a practical deployment of Obscuro can handle thousands of deposits in a mixing round. Note that the transaction signing operation is the major contributor to the mixer performance and it is known to scale quadratically due to its re-hashing mechanism.

On-blockchain evaluation. We also deploy Obscuro on Bitcoin Testnet, a global testing environment mimicking the mainnet, having each user sends 0.01 Testnet Bitcoin through Obscuro. We have successfully mixed 430 users in a

[9] Latest checkpoint is at block 295000, see `chainparams.cpp` in Bitcoin Core's codebase.

standard transaction[10] and 1000 users in a non-standard transaction[11]. We set the transaction fee to be around 22 satoshi/bytes, which make the transactions likely to be mined instantly. Each user will need to pay the transaction fee in the deposit transaction and partly the mixed transaction, which takes account in total only 0.0001 BTC (0.8 USD as in April 2018).

Trusted Computing Base (TCB). We measure the size of the TCB of our prototype. OBSCURO's trusted functions contribute 1,150 source lines of code (SLoC). OBSCURO also requires some changes in the Bitcoin Core implementation which contribute 1,292 SLoC. Thus, OBSCURO contributes a total of 2,442 SloC to the TCB. The entire TCB includes the Bitcoin Core implementation, two widely used cryptographic libraries (i.e., libsecp256k1 and OpenSSL), and the Panoply implementation, in which users can audit to verify whether they deviate from their public codebases. [12]

6 Discussion

6.1 Recipient of the Mixing Fees

Like many other mixing services, OBSCURO may enforce some participation fees (e.g., 1–3 percent of the mixing value) on top of the transaction fees to deter the DoS and Sybil attacks [12,14,18,51,57]. However, if the recipient of the mixing fees happens to be malicious, the deterrence does not work. For example, the recipient of the mixing fees can generate a large number of Sybil deposits *without* any mixing fees because the mixing fees will be paid to herself eventually. Deciding who receives the mixing fees is challenging and has been rarely addressed in previous works.

The most secure defense against this subtle attack is to *burn* the mixing fees by sending the fees to an *unspendable* address (similar to PeerCoin [4]). Note that burning fee can cause a small deflation on the total supply of Bitcoin but completely prevent the aforementioned risk. As a more economically viable solution that is less ideal in terms of security, some reputable charity organizations or privacy advocacy organizations (e.g., EFF [1] or Tor [5]), which are believed to be honest, can be set as the recipients of the mixing fee. It is even possible to allow each user to choose the recipient of his mixing fee from a list of several reputable organizations (e.g., using different identifiers in the `OP_RETURN` output) or the burning option.

6.2 Multiple Obscuro Instances

OBSCURO is necessarily an open source project (for public scrutiny of its code) and thus any third party can spawn an OBSCURO instance simply by getting the open-source implementation and running it on an Intel SGX platform. Consequently, users may see many OBSCURO instances, which have different identities

[10] https://www.blocktrail.com/tBTC/tx/59e1f4ffe3e6b735f279f340a088597af45f545e6bab4542c82a24d0014b59b9

[11] https://www.blocktrail.com/tBTC/tx/f5230965145ef06eb65595e41ecb701af6c128802a174f34a7b65ac7d44dc9b8

[12] https://github.com/BitObscuro/Obscuro

Bitcoin mixers	Theft prevention	Relationship anonymity	Participation guarantee	Large mixing set guarantee	Join-then-abort resistance	On-chain transactions
Decentralized						
CoinJoin [26]	✓	✗[3]	✓	small set[5]	✗	1
CoinShuffle [42,43]	✓	✓	✓	small set[5]	✗	1
CoinParty [57]	✓[1]	✓	✓	✓	✗	2
Xim [12]	✓	✗[3]	✓	small set[5]	✓	7
Centralized						
MixCoin [14]	✗[2]	✗[4]	✗	✗	✓	2
BlindCoin [51]	✗[2]	✓	✗	✗	✓	2
TumbleBit [18]	✓	✓	✗	✗	✓	4
Obscuro	✓	✓	✓	✓	✓	2

Table 1: Comparison between existing Bitcoin mixer proposals and Obscuro. [1] CoinParty only achieves theft prevention if 2/3 users are honest. [2] MixCoin and BlindCoin only provide accountability.[3] Users can link all senders and recipients. [4] Mixer operator can link all senders and recipients. [5] Users know the mixing set size before mixing, but the set size is small.

(i.e., address and public key), are successfully verified via remote attestation. In fact, this does not provide any economic gain to the third-party service provider, since the recipients of the mixing fees are hard-coded in Obscuro's codebase, which is owned and maintained by the original service provider. However, having many Obscuro instances may cause users to become confused and deposit their coins to third-party Obscuro instances, resulting in the shortage of deposits at the original one.

To thwart this concern, the original operator can provide a signature that certifies the Obscuro instance that it is operating, along with the attestation data that is published in the public bulletin boards. Then, users are advised to deposit coins only to the address of the Obscuro instance that is run by the original operator, who takes the responsibility to maintain the Obscuro codebase.

7 Related Work

Privacy of Bitcoin and altcoins have been actively studied in the past few years. In this section, we first summarize Bitcoin-based mixer proposals and then discuss other non-Bitcoin proposals. Moreover, we outline a recent trend in utilizing trusted hardware in cryptocurrency or, in general, blockchain applications.

7.1 Existing Bitcoin Mixer Solutions

Existing mixer proposals can be classified into two main groups based on their design, namely centralized and decentralized mixers. We compare our scheme Obscuro with existing proposals in various aspects (see Table 1).

Decentralized mixers. In decentralized mixing protocols, participants communicate among themselves to privately permute the ownership of their coins. In CoinJoin, users mutually sign on a single transaction where each user controls an input and an output addresses [26]. However, CoinJoin allows every participant to learn all the between any sender and recipient addresses. CoinShuffle [42] and its successor CoinShuffle++ [43] use an additional overlay cryptographic mixing protocol on top of CoinJoin to provide relationship anonymity for all sender and recipient pairs. CoinParty [57] is another decentralized mixing protocol. However, it relies on an assumption that 2/3 of the peers are honest, which could easily be violated in practice. In these decentralized protocols, users are allowed to refuse to agree on the transaction, thus coin stealing are prevented; however, a malicious user can initially participate in the execution of the protocol, but aborts before the end of the execution in order to disrupt the mixing of the honest users (i.e.,*join-then-abort attack*). CoinShuffle and CoinShuffle++ can identify which users have been aborted but the rest has to mix again. As a consequence, They only mix among a relatively small set of users (e.g., 50 participants). To discourage the join-then-abort attack, XIM [12] proposes a two-party mixing protocol where the users need to pay a participation fee and advertise themselves on the blockchain. Nevertheless, XIM requires multiple on-chain transactions and thus may take hours to finish a mixing round between only two participants.

Centralized mixers. In Mixcoin, users send coins to a centralized party and receive back the mixed coins [14]. Mixcoin is invulnerable to join-then-abort attack since the users join the mix independently. Mixcoin does not achieve relationship anonymity despite users cannot identify the recipient addresses of others because the mixer operator knows all the links between senders and their recipients. Blindcoin modifies Mixcoin using blind signature scheme so that the mixer operator cannot learn the links [51]. Since the mixer operators of MixCoin and Blindcoin may steal users' coins, they give users signed certificates to provide accountability, which can damage the reputation of a malicious mixer operator. The accountability property is far from ideal as coin theft cannot be prevented. TumbleBit presents an untrusted intermediate payment hub between the payer and payee, involving a cryptographic puzzle promise and solver protocol among them to prevent coin theft [18]. While these centralized protocols can mix participants in a large set (e.g., 800 users in TumbleBit), they fail to defend against the participation rejection attack (see Section 2.2) because the malicious mixer service providers can selectively reject to interact with some benign users. Also, there is no in-protocol guarantee regarding the mixing set size and thus users may need to participate in subsequent rounds until they are satisfied with large enough mixing set size.

Comparison with Obscuro. From Table 1, we can see that OBSCURO outperforms all other mixers in most of the properties. In particular, OBSCURO protects the coins and ensures the unlinkability between all senders and their recipients. We also guarantee participation for benign users and allow them to verify the minimal size of the mixing set they will be included *before* they send their deposits (see Section 3.5). Similar to other centralized mixers, there is no coordination among users in OBSCURO and a malicious participant cannot disrupt others during mixing. In fact, a sender and recipient pair is only involved in two transactions, a deposit transaction and a mixed transaction, which is slightly higher than it is in decentralized proposals.

7.2 Privacy Improvements in other Cryptocurrencies

Several privacy-enhancing cryptocurrencies have been proposed and built recently such as Monero [3], ZCash [31, 44] and MimbleWimble [20]. In Monero, ring signature is used to hide a sender's address within a group of others. Monero users can also use stealth addresses to hide their actual outputs while transacting, and ring confidential transaction to hide the transferring amount of coins. Unfortunately, recent studies demonstrate various anonymity attacks against these properties [22,32]. ZCash is a cryptocurrency developed from ZeroCoin [31] and ZeroCash [44] proposals. ZCash protocol is based on a zero-knowledge proof called SNARKs [10] where transactions reveal no information about the transacting amount or recipients. However, ZCash requires a trusted setup, limiting its large-scale adoption. MimbleWimble is an extension of Bitcoin protocol and offers confidential transactions; however, its functionalities lack script support. There is also a confidential transaction proposal for Bitcoin, which motivates ValueShuffle to mix transactions with different transferring amounts [41].

In some other cryptocurrencies with no built-in privacy such as Ethereum and Ripple, there are also some mixer proposals that aim to improve the transaction anonymity. For instance, Möbius replaces a central mixer with an Ethereum smart contract employed with ring signatures and stealth addresses so that a recipient can withdraw coins without being linked with any sender [28]. PathShuffle proposes a mixing protocol for path-based transactions in credit networks and demonstrates mixing in Ripple [34].

While these alternative cryptocurrencies are promising, Bitcoin still remains as the most popular cryptocurrency with the largest market capitalization. Obscuro aims to provide a secure and anonymous mixing service for Bitcoin transactions, without any need to modify the current protocol.

7.3 TEE for Cryptocurrency Applications

The trusted hardware has opened a new range of research problems — including ones in cryptocurrency research. For instance, a recently proposed off-chain micropayment channel, named Teechan, utilizes the trusted execution environment (e.g., Intel SGX) to scale up transaction throughput of Bitcoin transactions to thousands per second [24]. Intel SGX is utilized in Town Crier to provide authenticated data to the Ethereum smart contracts system [54]. A recent proposal named Tesseract uses Intel SGX to build a real-time cryptocurrency exchange [11]. The TEE is also used to securely report CPU cycles, or a Proof-of-Useful-Work, which is the foundation of a blockchain mining framework called REM [55].

8 Conclusion

Mixing Bitcoin transactions significantly improves the anonymity of Bitcoin by providing relationship anonymity to transactions. Bitcoin mixers must guarantee protection against strong adversaries (e.g., malicious service providers), provide strong anonymity guarantees, and support large-size anonymity set and short mixing time, which have not been completely achieved by prior work. We exploit a new security capability in emerging CPUs, trusted execution environment (TEE), to design a secure and anonymous Bitcoin centralized mixer. Our OBSCURO mixer demonstrates that the strong security and anonymity guarantees are achievable for fast and large-size mixing services.

References

1. Electronic Frontier Foundation (EFF). `https://www.eff.org/`
2. IPFS. `https://ipfs.io/`
3. Monero. `https://getmonero.org/`
4. PeerCoin. `https://peercoin.net`
5. Tor. `https://www.torproject.org/`
6. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative technology for CPU based attestation and sealing. In: Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy. vol. 13. ACM New York, NY, USA (2013)
7. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 34–51. Springer (2013)
8. ARM, A.: Security technology building a secure system using trustzone technology (white paper). ARM Limited (2009)
9. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. J. Cryptology 21(4), 469–491 (2008)
10. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: Advances in Cryptology–CRYPTO 2013, pp. 90–108. Springer (2013)
11. Bentov, I., Ji, Y., Zhang, F., Li, Y., Zhao, X., Breidenbach, L., Daian, P., Juels, A.: Tesseract: Real-time cryptocurrency exchange using trusted hardware (2017)
12. Bissias, G., Ozisik, A.P., Levine, B.N., Liberatore, M.: Sybil-resistant mixing for bitcoin. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. pp. 149–158. ACM (2014)
13. Black, P.E.: Fisher-yates shuffle. Dictionary of algorithms and data structures 19 (2005)
14. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: Anonymity for bitcoin with accountable mixes. In: International Conference on Financial Cryptography and Data Security. pp. 486–504. Springer (2014)
15. Brown, D.: Standards for efficient cryptography, SEC 1: elliptic curve cryptography. Released Standard Version 1 (2009)
16. Chandrasekaran, K., Karp, R., Moreno-Centeno, E., Vempala, S.: Algorithms for implicit hitting set problems. In: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms. pp. 614–629. Society for Industrial and Applied Mathematics (2011)

17. Danezis, G., Serjantov, A.: Statistical disclosure or intersection attacks on anonymity systems. In: International Workshop on Information Hiding. pp. 293–308. Springer (2004)
18. Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. Proceedings of NDSS 2017 (2017)
19. Intel: Attestation Service for Intel® Software Guard Extensions: API Documentation. `https://software.intel.com/sites/default/files/managed/7e/3b/ias-api-spec.pdf` (2017)
20. Jedusor, T.E.: Mimblewimble (2016)
21. Johnson, S., Scarlata, V., Rozas, C., Brickell, E., Mckeen, F.: Intel® Software Guard Extensions: EPID Provisioning and Attestation Services. White Paper 1, 1–10 (2016)
22. Kumar, A., Fischer, C., Tople, S., Saxena, P.: A traceability analysis of Monero's blockchain. In: European Symposium on Research in Computer Security. pp. 153–173. Springer (2017)
23. Lee, S., Shih, M.W., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside sgx enclaves with branch shadowing. In: 26th USENIX Security Symposium, USENIX Security. pp. 16–18 (2017)
24. Lind, J., Eyal, I., Pietzuch, P., Sirer, E.G.: Teechan: Payment Channels Using Trusted Execution Environments. In: 4th Workshop on Bitcoin and Blockchain Research (2017)
25. Matetic, S., Ahmed, M., Kostiainen, K., Dhar, A., Sommer, D., Gervais, A., Juels, A., Capkun, S.: ROTE: Rollback Protection for Trusted Execution. In: 26th USENIX Security Symposium, USENIX Security. pp. 1289–1306 (2017)
26. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world. In: Post on Bitcoin forum (2013)
27. McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative instructions and software model for isolated execution. In: HASP@ ISCA. p. 10 (2013)
28. Meiklejohn, S., Mercer, R.: Möbius: Trustless tumbling for transaction privacy. Proceedings on Privacy Enhancing Technologies 2018(2), 105–121 (2018)
29. Meiklejohn, S., Orlandi, C.: Privacy-enhancing overlays in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 127–141. Springer (2015)
30. 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)
31. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from bitcoin. In: Security and Privacy (SP), 2013 IEEE Symposium on. pp. 397–411. IEEE (2013)
32. Miller, A., Möser, M., Lee, K., Narayanan, A.: An empirical analysis of linkability in the Monero blockchain. arXiv preprint arXiv:1704.04299 (2017)
33. Moghimi, A., Irazoqui, G., Eisenbarth, T.: Cachezoom: How sgx amplifies the power of cache attacks. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 69–90. Springer (2017)
34. Moreno-Sanchez, P., Ruffing, T., Kate, A.: Pathshuffle: Credit mixing and anonymous payments for ripple. Proceedings on Privacy Enhancing Technologies 2017(3), 110–129 (2017)

35. Moser, M., Bohme, R., Breuker, D.: An inquiry into money laundering tools in the bitcoin ecosystem. In: eCrime Researchers Summit (eCRS), 2013. pp. 1–14. IEEE (2013)
36. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. bitcoin.org (2009)
37. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management (2010)
38. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Security and privacy in social networks, pp. 197–223. Springer (2013)
39. Reid, K.: Banknotes and their vindication in eighteenth-century scotland. David Fox and Wolfgang Ernst (eds), Money in the Western Legal Tradition (Oxford University Press, 2014, Forthcoming); Edinburgh School of Law Research Paper No. 2013/19 (2013)
40. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: International Conference on Financial Cryptography and Data Security. pp. 6–24. Springer (2013)
41. Ruffing, T., Moreno-Sanchez, P.: Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 133–154. Springer (2017)
42. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Coinshuffle: Practical decentralized coin mixing for bitcoin. In: European Symposium on Research in Computer Security. pp. 345–364. Springer (2014)
43. Ruffing, T., Moreno-Sanchez, P., Kate, A.: P2P Mixing and Unlinkable Bitcoin Transactions. IACR Cryptology ePrint Archive 2016, 824 (2016)
44. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014)
45. Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard, S.: Malware guard extension: Using SGX to conceal cache attacks. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 3–24. Springer (2017)
46. Shih, M.W., Lee, S., Kim, T., Peinado, M.: T-SGX: Eradicating controlled-channel attacks against enclave programs. In: Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA (2017)
47. Shinde, S., Chua, Z.L., Narayanan, V., Saxena, P.: Preventing page faults from telling your secrets. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. pp. 317–328. ACM (2016)
48. Shinde, S., Tien, D., Tople, S., Saxena, P.: Panoply: Low-TCB Linux applications with SGX enclaves. In: Proceedings of the Annual Network and Distributed System Security Symposium (NDSS). p. 12 (2017)
49. Spagnuolo, M., Maggi, F., Zanero, S.: Bitiodine: Extracting intelligence from the bitcoin network. In: International Conference on Financial Cryptography and Data Security. pp. 457–468. Springer (2014)
50. Strackx, R., Piessens, F.: Ariadne: A minimal approach to state continuity. In: USENIX Security (2016)
51. Valenta, L., Rowan, B.: Blindcoin: Blinded, accountable mixes for bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 112–126. Springer (2015)
52. Wuille, P., et al.: libsecp256k1: Optimized C library for EC operations on curve secp256k1 (2015)

53. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 640–656. IEEE (2015)
54. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town Crier: An authenticated data feed for smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 270–282. ACM (2016)
55. Zhang, F., Eyal, I., Escriva, R., Juels, A., Van Renesse, R.: REM: Resource-Efficient Mining for Blockchains. IACR Cryptology ePrint Archive 2017, 179 (2017)
56. Zhang, F., Zhang, H.: Sok: A study of using hardware-assisted isolated execution environments for security. In: Proceedings of the Hardware and Architectural Support for Security and Privacy 2016. p. 3. ACM (2016)
57. Ziegeldorf, J.H., Grossmann, F., Henze, M., Inden, N., Wehrle, K.: Coinparty: Secure multi-party mixing of bitcoins. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy. pp. 75–86. ACM (2015)

A Structure of the Deposit Transaction

```
OP_IF
<pubkey_M> OP_CHECKSIG %public key associated with addr_M
OP_ELSE
<time-lock> OP_CHECKLOCKTIMEVERIFY OP_DROP
<pubkey_A> OP_CHECKSIG %public key associated with addr_A
OP_ENDIF
```

Fig. 5: Structure of the redeem script.

```
Input:
  scriptSig: <signature_A> <pubkey_A>
Output:
  Index: 0
  Value: 0
  scriptPubKey:
    OP_RETURN <identifier> <Encrypted(addr_B)>

  Index: 1
  Value: 1000000 %The denomination is 0.01 bitcoin
  scriptPubKey:
    OP_HASH160 <Hash160(redeem_script)> OP_EQUAL
```

Fig. 6: Structure of the deposit transaction.

Here, we describe the format of the deposit transaction that the users submit to Obscuro. Particularly, the recipient address addr_B is a Pay-To-Script-Hash (P2SH) address and is encrypted with the Obscuro's pk_M. The user then needs to construct a redeem script that follows the format shown in Figure 5. Essentially, this script allows both the mixer and the user spend the deposit transaction but the user can only do that after $\langle time - lock\rangle$ value (e.g., 100 blocks). Next, the user then hashes the redeem script, puts the hash in the `scriptPubKey`, and broadcasts the deposit transaction to the Bitcoin network (See Figure 6).