Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions

Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg

Boston University
{heilman, foteini}@bu.edu, goldbe@cs.bu.edu

Abstract. Although Bitcoin is often perceived to be an anonymous currency, research has shown that a user's Bitcoin transactions can be linked to compromise the user's anonymity. We present solutions to the anonymity problem for both transactions on Bitcoin's blockchain and off the blockchain (in so called micropayment channel networks). We use an untrusted third party to issue anonymous vouchers which users redeem for Bitcoin. Blind signatures and Bitcoin transaction contracts (aka smart contracts) ensure the anonymity and fairness during the *bitcoin* $\leftrightarrow$ *voucher* exchange. Our schemes are practical, secure and anonymous.

Keywords: Bitcoin, blockchain, smart contracts, blind signatures, anonymity.

1 Introduction

When Bitcoin was first introduced in 2008, one of its key selling points was anonymity—users should be able to spend bitcoins "without information linking the transaction to anyone" [15]. In the last few years, however, researchers have shown that Bitcoin offers much weaker anonymity than was initially expected [12, 17], by demonstrating that they could follow the movement of funds on the Bitcoin blockchain. The community has reacted to this by proposing two key approaches to improve the anonymity of Bitcoin: (1) new anonymity schemes that are compatible with Bitcoin [1, 7, 18, 23, 3, 11, 2, 19, 24], and (2) new anonymous cryptocurrencies that are independent of Bitcoin [14, 2]. In this paper we take the former approach by developing new anonymity schemes that are compatible with Bitcoin via a soft fork. Our schemes offer a new trade-off between practicality (*i.e.,* transaction speed), security (*i.e.,* resistance to double-spending, denial of service (DoS) and Sybil attacks) and anonymity (*i.e.,* unlinkable transactions). As we will see below, previous work either provided schemes that are efficient but achieve limited security or anonymity [7, 18, 23, 24, 19] or schemes that provide strong anonymity but are slow and require large numbers of transactions [11, 3, 1].

Our first scheme is an "on-blockchain" scheme providing anonymity at reasonable speed, *i.e.,* requiring four transactions to be confirmed in three blocks ($\approx$ 30 mins). Our protocol runs in epochs, and provides *set-anonymity within each epoch*. That is, while the blockchain publicly displays the *set* of payers and payees during an epoch, no one can tell which payer paid which payee. To do this, we introduce an untrusted (possibly malicious) intermediary $\mathcal{I}$ between all payers and payees.

Our second "off-blockchain" scheme uses a new payment technology called *micropayment channel networks* [16, 9]. Micropayment channel networks use Bitcoin as a platform to confirm transactions within seconds, rather than minutes, and already provide a degree of anonymity—most of the transactions are made outside of the blockchain, and thus not shown to the public—but this anonymity is incomplete. Critically, because micropayment channel networks chain payments through pre-established paths of connected users (explained in Section 5.1), these users that participate in the path learn transaction details, including the cryptographic identities of the sending and receiving party. We provide anonymity against malicious users by using an *honest-but-curious* intermediary $\mathcal{I}$ (Section 5.3); set-anonymity within an epoch is preserved as long as $\mathcal{I}$ does not abort or deny service to payers or payees.

Our technique, inspired by eCash [8], works as follows. For a user $\mathcal{A}$ to anonymously pay another user $\mathcal{B}$, she would first exchange a bitcoin for an *anonymous voucher* through intermediary $\mathcal{I}$. $\mathcal{B}$ could then redeem the anonymous voucher with $\mathcal{I}$ to receive a bitcoin back. Our scheme overcomes two main challenges: (i) Ensuring that the vouchers are unlinkable (*i.e.,* hiding the link between the issuance and the redemption of a voucher), and (ii) enforcing fair exchange between participants (*i.e.,* users can redeem issued vouchers even against an uncooperative or malicious $\mathcal{I}$, and no party can steal or double-spend vouchers and bitcoins). We use *blind signatures* to achieve unlinkability, and the *scripting* functionality of Bitcoin transactions to achieve fair exchange via transaction contracts (aka smart contracts [20]).

We provide an overview of our scheme in Section 2 and define the required properties. We discuss our use of transaction contracts in Section 3. Our scheme for on-blockchain anonymous transactions is in Section 4. Our off-blockchain scheme which uses micropayment channel networks is in Section 5. Finally, we analyze the anonymity of our schemes in Sections 4.2 and 5.3 and their security in Section 6.

1.1 Related Work

We now review some of the most representative related works in the literature.

Anonymous Payment Schemes. Zerocash [2] and Zerocoin [14] provide anonymous payments through the use of a novel type of cryptographic proofs (ZK-SNARKs). Unlike our schemes, they are "stand-alone" cryptocurrencies and can not be integrated with Bitcoin. Meanwhile, [19] is an anonymous payment scheme that can offer anonymity protections to Bitcoin that provides excellent blockchain privacy and is very fast. However, the parties entrusted to anonymize transactions in [19] can still violate users' anonymity, even if they are honest-but-curious.

Mixing Services. A bitcoin mixing service provides anonymity by transferring payments from an input set of bitcoin addresses to an output set of bitcoin addresses, such that is it hard to trace which input address paid which output address. Mixcoin [7] uses a trusted third party to mix Bitcoin addresses, but this third party can violate users privacy and steal users' bitcoins; theft is detected but not prevented. Blindcoin [23] improves on Mixcoin by preserving users privacy against the mixing service, as with Mixcoin, theft is still not prevented. CoinParty [24] is secure if $2/3$ of the mixing parties are honest. CoinJoin [10] and CoinShuffle [18] improve on prior work by preventing theft. [13] shows a rigorous proof of anonymity for a scheme "almost identical" to CoinShuffle.

CoinShuffle's anonymity set is thought to be small due to coordination costs [3, 6]; meanwhile, our schemes are not limited to small anonymity sets. Moreover, both CoinShuffle and CoinJoin run an entire mix in a single bitcoin transaction. Thus, a single aborting user disrupts the mix for all other users. Moreover, mix users cannot be forced to pay fees upfront, so that these schemes are vulnerable to DoS attacks [6, 22] (where users join the mix and then abort) and Sybil attacks (where an adversary deanonymizes a user by forcing it to mix with Sybil identities secretly under her control) [3].

XIM [3] is a decentralized protocol which builds on the fair-exchange mixer in [1] and prevents bitcoin theft and resists DoS and Sybil attacks via fees. We also prevent bitcoin theft resist DoS and Sybil attacks with fees (Section 4.1). One of XIM's key innovations is a secure method for partnering mix users. Unfortunately, this partnering method adds several hours to the protocol execution because users have to advertise themselves as mix partners on the blockchain. Our schemes are faster because they do not require a partnering service.

CoinSwap [11] is a fair-exchange mixer that allows two parties to anonymously send Bitcoins through an intermediary. Like our schemes, the CoinSwap intermediary is prevented from stealing funds by the use of fair exchange. Unlike our schemes, however, CoinSwap does not provide anonymity against even a honest but curious intermediary. Our on-blockchain scheme takes $\approx$ 30 mins, slower than Coinshuffle's $\approx$ 10 mins. Off-blockchain however, our scheme is faster than CoinShuffle, since it only runs in seconds [16]; however, our off-blockchain only supports anonymity against a honest-but-curious intermediary[1].

2 Overview and Security Properties

We introduce two schemes: (a) on-blockchain anonymous payments and coin mixing, and (b) off-blockchain anonymous payments. By on-blockchain we denote the standard method of transferring bitcoins *i.e.,* using the Bitcoin blockchain, as opposed to the newly proposed "off-blockchain" methods that utilize micropayment channel networks.

On-blockchain Anonymous Payments. We first consider the scenario where a user $\mathcal{A}$, the *payer* wants to anonymously send 1 bitcoin, BTC, to another user $\mathcal{B}$, the *payee*[2] If $\mathcal{A}$ were to perform a standard Bitcoin transaction, sending 1 bitcoin from an address $Addr_A$ (owned by $\mathcal{A}$) to a fresh ephemeral address $Addr_B$ (owned by $\mathcal{B}$) there would be a record of this transaction on Bitcoin's blockchain linking $Addr_A$ to $Addr_B$. Even if $\mathcal{A}$ and $\mathcal{B}$ always create a fresh address for each payment they receive, the links between addresses can be used to de-anonymize users if, at some point, they "non-anonymously" spend a payment (*e.g.,* buying goods from third party that learns their mailing address) or receive a payment (*e.g.,* a Bitcoin payment processor like BitPay) [12].

One idea $\mathcal{A}$ and $\mathcal{B}$ could use to protect their privacy is to employ an intermediary party $\mathcal{I}$ that breaks the link between them. $\mathcal{A}$ would first send one bitcoin to $\mathcal{I}$, and then

[1] Our off-blockchain scheme is fast because it uses micropayment channel networks. It's unclear how to retrofit prior work onto these networks, *e.g.,* mapping Coinshuffle's single atomic transaction onto the arbitrary graph topology of a micropayment channel network.

[2] We assume that all transactions in our schemes are of 1 bitcoin value.

$\mathcal{I}$ would send a different bitcoin to $\mathcal{B}$. Assuming that a sufficient number of users make payments through $\mathcal{I}$, it becomes more difficult for an outsider to link $\mathcal{A}$ to $\mathcal{B}$ by looking at the blockchain (more on this below). The downside of this idea, however, is that the intermediary $\mathcal{I}$ knows everything about all users' payments, violating their anonymity.

We could apply techniques used in online anonymous eCash schemes [8] to prevent $\mathcal{I}$ from learning who $\mathcal{A}$ wants to pay. The protocol is in Figure 1. $\mathcal{A}$ pays one bitcoin to $\mathcal{I}$, and obtains an *anonymous voucher* $V = (sn, \sigma)$ in return. ($\mathcal{A}$ chooses a random serial number sn, blinds it to $\overline{sn}$ and asks $\mathcal{I}$ to compute a blind signature $\overline{\sigma}$ on $\overline{sn}$. $\mathcal{A}$ unblinds these values to obtain $V = (sn, \sigma)$. The blind signature requires only a minor change to Bitcoin and can be implemented using a soft fork (Section 3). Then $\mathcal{A}$ pays $\mathcal{B}$ using V, and finally $\mathcal{B}$ redeems V with $\mathcal{I}$ to obtain one bitcoin.

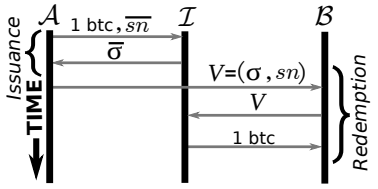

Fig. 1: Strawman eCash protocol.

How do we ensure that $\mathcal{I}$ does not know who $\mathcal{A}$ wants to pay? This follows from the *blindness* of blind signatures—namely, that the signer ($\mathcal{I}$) cannot read the blinded serial number $\overline{sn}$ that it signs, and also cannot link a message/signature (sn, σ) pair to its blinded value $(\overline{sn}, \overline{\sigma})$. Blindness therefore ensures that even a malicious $\mathcal{I}$ cannot link a voucher it redeems with a voucher it issues. Blind signatures are also *unforgeable*, which ensures that a malicious user cannot issue a valid voucher to itself.

While this eCash-based approach solves our anonymity problem, it fails when $\mathcal{I}$ is malicious since it could just refuse to issue a voucher to $\mathcal{A}$ after receiving her bitcoin. To solve this, we use Bitcoin transaction contracts to achieve blockchain-enforced *fair exchange* (as in prior work, fair-exchange denotes an atomic swap). The key idea is that $\mathcal{A}$ transfers a bitcoin to $\mathcal{I}$ *if and only if* it receives a valid voucher V in return. Figure 2 presents the high-level idea, and full description is in Section 4.

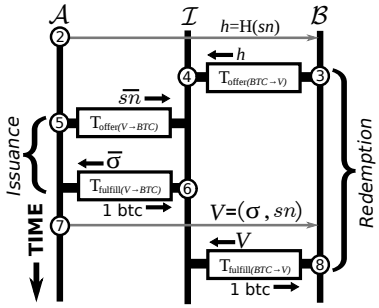

Fig. 2: Our protocol: Circles (step numbers from Section 4), black arrows (objects transfered via transaction), grey arrows (messages).

At a high-level, our scheme consists of four blockchain transactions that are confirmed in three blocks on the blockchain, as shown in Figures 2-3. The protocol involves two blockchain-enforced fair exchanges. The first is $V \to BTC$, which exchanges a voucher from $\mathcal{B}$ for a bitcoin from $\mathcal{I}$, and is realized using the following two transaction contracts: (1) $T_{offer(V \to BTC)}$, which is created by $\mathcal{I}$, confirmed in the first block on the blockchain and offers a fair exchange of one bitcoin (from $\mathcal{I}$) for one voucher (from $\mathcal{B}$), and (2) $T_{fulfill(V \to BTC)}$, which is created by $\mathcal{B}$ to fulfill the offer by $\mathcal{I}$ and is confirmed in the third block on the blockchain. These transaction contracts ensure that a malicious $\mathcal{I}$ cannot redeem $\mathcal{B}$'s voucher without providing $\mathcal{B}$ with a bitcoin in return (see Sections 3,4). The second fair exchange is $BTC \to V$ and works in a similar fashion, fairly exchanging a bitcoin from $\mathcal{A}$ for a voucher from $\mathcal{I}$ via

two transaction contracts: (1) $T_{offer(BTC \to V)}$, created by $\mathcal{A}$ and confirmed on the second block, and (2) $T_{fulfill(BTC \to V)}$, created by $\mathcal{I}$ and confirmed in the third block. These two fair exchanges are arranged to realize the anonymity protocol shown on the previous page; the fair exchange $BTC \to V$ stands in for the interaction between $\mathcal{A}$ and $\mathcal{I}$, while the fair exchange $V \to BTC$ stands in for the interaction between $\mathcal{B}$ and $\mathcal{I}$.

Mixing Service. A mixing service allows a user to move bitcoins from one address it controls to a fresh ephemeral (thus anonymous) address, without directly linking the two addresses on the blockchain. To use our on-blockchain anonymous payments as a mixing service, users can just anonymously pay themselves from one address to another fresh ephemeral address, thus playing the role of both $\mathcal{A}$ and $\mathcal{B}$ in the protocol above.

Off-blockchain Payments. We also adapt our scheme to the recently proposed off-blockchain *micropayment channel networks*. Our off-blockchain scheme uses the same four transactions described above, but confirms them on a micropayment channel network. See Section 5 for details.

2.1 Anonymity Properties

In the strawman eCash protocol of Figure 1, the anonymity level of users depends on the total number of payments using $\mathcal{I}$ as users can obtain or redeem vouchers at arbitrary times. However, our anonymous fair-exchange protocol of Figure 2 provides anonymity only for payments starting and completing within an *epoch* (Figure 3) *i.e.,* a three block window.

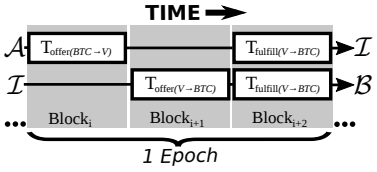

Fig. 3: Payment Epoch

Assumptions. We make the following assumptions for our schemes:

1. We assume that all users coordinate on epochs (by *e.g.,* choosing the starting block to have a block height that is divisible by three).
2. As with traditional eCash schemes, we assume that if $\mathcal{A}$ pays $\mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ trust each other. (A malicious $\mathcal{A}$ or $\mathcal{B}$ could easily conspire with $\mathcal{I}$ to reveal the other party of the transaction; for instance, $\mathcal{A}$ could just tell $\mathcal{I}$ the serial number the voucher she was issued, and then $\mathcal{I}$ can identity $\mathcal{B}$ when he redeems that voucher.) This is a reasonable assumption in cases where $\mathcal{A}$ is purchasing goods from $\mathcal{B}$, since $\mathcal{A}$ is likely already trusting $\mathcal{B}$ with far more personal and identifying information including *e.g.,* her shipping address or IP address.
3. For our on-blockchain scheme only, payees $\mathcal{B}$ always receive payments in a fresh ephemeral Bitcoin address $Addr_B$ controlled by them. Any communication between $Addr_B$ and $\mathcal{I}$ is done anonymously (*e.g.,* using Tor). The payee can transfer the payment from $Addr_B$ to his long-lived Bitcoin address if the protocol successfully completes.
4. Payers only make one anonymous payment per epoch. Similarly, payees only accept one payment per epoch (*i.e.,* we assume they do not create multiple ephemeral addresses to receive multiple payments in one epoch). [3]

[3] We could allow users to perform multiple payments (by using multiple Bitcoin addresses that belong to them) but this would reduce their anonymity and make our analysis more complex.

Given these assumptions, the anonymity properties of our on-blockchain scheme are:

Set-Anonymity within an Epoch. Our assumptions imply that in every epoch there are exactly n addresses making payments (playing the role of payer $\mathcal{A}$) and n receiving addresses (playing the role of $\mathcal{B}$). All these Bitcoin addresses should belong to different users. Anyone looking at the blockchain can see the participating addresses of payers and payees, but should not be able to distinguish which payer paid which payee within a specific epoch. Thus, for all successfully completed payments within an epoch, the offered anonymity set has size n. In other words, the probability of successfully linking any chosen payer $\mathcal{A}$ to a payee should not be more than $1/n$ plus some negligible function. This means that an adversary (or a potentially malicious $\mathcal{I}$) can do no better than randomly guessing who paid whom during an epoch.

Resilient Anonymity. All payments should be totally anonymous until the recipient, $\mathcal{B}$, chooses to transfer them to an address linkable to $\mathcal{B}$. Even if a party *aborts* our protocol before it completes in an epoch, the *intended* recipient of a payment should remain totally anonymous.

Transparency of Anonymity Set. Users in our on-blockchain scheme learn the membership of their anonymity set after a transaction completes, just like anyone else who might be looking at the blockchain. This property is unusual for eCash schemes, but quite common for bitcoin mixes. Thus, if a particular $\mathcal{B}$ feels his anonymity set is too small in one epoch, he can increase the size of his anonymity set by remixing in a subsequent epoch. For instance, if $Addr_B$ gets paid in an epoch with $n = 4$, he can create a fresh ephemeral address $Addr'_B$ and have $Addr_B$ pay $Addr'_B$ in a subsequent epoch. If the subsequent epoch has a $n = 100$, then $\mathcal{B}$ increases the size of his anonymity set.

Our on-blockchain protocol achieves all the above anonymity properties, which also generalize to our mixing service (Section 4.2). Our mixing service has the additional advantage that $\mathcal{A}$ does not need to trust $\mathcal{B}$ since they are the same user. Our off-blockchain scheme only offers set-anonymity against $\mathcal{I}$ when $\mathcal{I}$ is honest-but-curious, rather than malicious (Section 5.3). Additionally, our off-blockchain scheme does not achieve the anonymity-set transparency or the anonymity resilience property.

Remark: Intersection Attacks. Anyone observing the anonymity-set membership in each epoch can attempt *intersection attacks* that de-anonymize users across epochs (*e.g.,* frequency analysis). This follows because we are *composing* set-anonymity across multiple epochs, and is a downside of any mix-based service that composes across epochs. ([3] has a detailed description of intersection attacks.) By anonymity transparency, anyone looking at the blockchain can attempt an intersection attack on our on-blockchain scheme. Our off-blockchain scheme (roughly) only allows $\mathcal{I}$ to do this (Section 5.3).

2.2 Security properties

Fair-exchange. There will always be a *fair-exchange* between $V \leftrightarrow BTC$. Our property ensures that: (i) malicious intermediary $\mathcal{I}$ cannot obtain a bitcoin from $\mathcal{A}$ unless it honestly creates a voucher for her, and (ii) malicious intermediary $\mathcal{I}$ cannot obtain a voucher V from $\mathcal{B}$ and refuse to pay a bitcoin back. This property is also true against

malicious users: (iii) malicious $\mathcal{A}$ cannot refuse to give a bitcoin to $\mathcal{I}$ when receiving V, and (iv) malicious $\mathcal{B}$ cannot receive a bitcoin from $\mathcal{I}$ without presenting a (valid) V.

Unforgeability. A user cannot create a valid V without interacting with $\mathcal{I}$.

Double-spending Security. A user can not redeem the same V more than once.

DoS Resistance. The intermediary $\mathcal{I}$ should be resistant to Denial of Service (DoS) attacks where a malicious user starts but never finishes many parallel fair exchanges (redemptions) of a V for a BTC.

Sybil Resistance. The protocol should be resistant to a Sybils (*i.e.,* identities that are under the control of single user) that attempt to de-anonymize a target user.

3 Implementing fair exchange via scripts and blind signatures

We explain how the transaction contracts $T_{offer(BTC \to V)}$ and $T_{fulfill(BTC \to V)}$ implement the fair exchange $BTC \to V$ used in our protocol ($V \to BTC$ is analogous).

We start with some background on transaction contracts. Recall that Bitcoin has no inherent notion of an "account"; instead, users merely move bitcoins from old transactions to new transactions, with the blockchain providing a public record of all valid moves. To do this, each transaction contains a list of *outputs*. These outputs hold a portion of that transaction's bitcoins and a set of rules describing the conditions under which the portioned bitcoins in that output can be transferred to a new transaction. The rules for spending outputs are written in a non-Turing-complete language called *Script*. One transaction *spends* another transaction when it successfully satisfies the rules in a script. Transaction contracts (aka smart contracts [20]) are written as scripts, *e.g.,* $\mathcal{A}$ will only pay $\mathcal{B}$ if some condition is met. Using the CHECKLOCKTIMEVERIFY feature [21] of scripts, we can *timelock* a transaction, so that funds can be reclaimed if a contract has not been spent within a given time window tw.

We use timelocking to implement the $BTC \to V$ fair exchange. The fair exchange begins when a user $\mathcal{A}$ generates (and the blockchain confirms) a transaction contract $T_{offer(BTC \to V)}$ which says that $\mathcal{A}$ offers one bitcoin to $\mathcal{I}$ under the condition "$\mathcal{I}$ must compute a valid blind signature on the blinded serial number $\overline{sn}$ within time window tw"; if the condition is not satisfied, the bitcoin reverts to $\mathcal{A}$. More precisely, $\mathcal{A}$ first chooses a random serial number sn, blinds it to $\overline{sn}$, and then uses $\overline{sn}$ to create a transaction contract $T_{offer(BTC \to V)}$ with an output of one bitcoin that is spendable in a future transaction T_f if one of the following conditions is satisfied:

1. T_f is signed by $\mathcal{I}$ and contains a valid blind signature $\overline{\sigma}$ on $\overline{sn}$[4], or
2. T_f is signed by $\mathcal{A}$ and the time window tw has expired.

The contract $T_{offer(BTC \to V)}$ is *fulfilled* if $\mathcal{I}$ posts a transaction $T_f = T_{fulfill(BTC \to V)}$ that contains a valid blind signature $\overline{\sigma}$ on $\overline{sn}$. This would satisfy the first condition of $T_{offer(BTC \to V)}$ and so the offered bitcoin is transferred from $\mathcal{A}$ to $\mathcal{I}$. If $\mathcal{I}$ does not fulfill the contract within the time window tw, then $\mathcal{A}$ signs and posts a transaction T_f that returns the offered bitcoin back to $\mathcal{A}$, thus satisfying the second condition of $T_{offer(BTC \to V)}$.

[4] $\mathcal{I}$ signs T_f to stop a malicious miner that learns $\overline{\sigma}$ from stealing the bitcoin $\mathcal{A}$ gives $\mathcal{I}$.

Blind Signature Scheme. Our fair exchange requires blind signatures with exactly two rounds of interaction. We use Boldyreva's [4] scheme, instantiated with elliptic curves for which the Weil or Tate pairing are efficiently computable and the computational Diffie-Hellman problem is sufficiently hard. While bitcoin supports elliptic curve operations, it uses a curve (Secp256k1) that does not support the required bilinear pairings. Thus, we need a soft fork to add an opcode that supports elliptic curves with efficient bilinear pairings (*i.e.,* supersingular curves of the type $y^2 = x^3 + 2x \pm 1$ over $\mathbb{F}_{e^\ell}$).

We use standard multiplicative notation and overlines to denote blinded values. Let $\mathbb{G}$ be a cyclic additive group of prime order p in which the gap Diffie-Hellman problem [5] is hard and $\mathbb{G}'$ a cyclic multiplicative group of prime order q. By e we denote the bilinear pairing map: $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}'$. Let g be a generator of the group and H be a hash function mapping arbitrary strings to elements of $\mathbb{G}\backslash\{1\}$. Let (p, g, H) be public parameters and $(sk, pk = g^{sk})$ be the signer's secret/public key pair.

- To blind sn, user $\mathcal{A}$ picks random $r \in \mathbb{Z}_p^*$ and sets $\overline{sn} = H(sn)g^r$.
- To sign $\overline{sn}$, signer $\mathcal{I}$ computes $\overline{\sigma} = \overline{sn}^{sk}$.
- To unblind the blind signature $\overline{\sigma}$, user $\mathcal{A}$ computes $\sigma = \overline{\sigma} pk^{-r}$.
- To verify the signature σ on sn, anyone holding pk checks that the bilinear pairing $e(pk, H(sn))$ is equal to $e(g, \sigma)$.
- To verify that the *blinded signature* $\overline{\sigma}$ on the blinded $\overline{sn}$, anyone holding pk can verify that this is valid (intermediate) signature by checking if $e(pk, \overline{m}) = e(g, \overline{\sigma})$.

4 On-Blockchain Anonymous Protocols

We now discuss the details of on-blockchain protocol depicted in Figure 2-3.

As shown in Figure 2, our protocol interleaves two fair exchanges $BTC \to V$ (implemented using $T_{offer(V \to BTC)}$ and $T_{fulfill(V \to BTC)}$) and $V \to BTC$ (implemented using $T_{offer(BTC \to V)}$ and $T_{fulfill(BTC \to V)}$). The interleaving is designed to ensure that a malicious $\mathcal{I}$ cannot issue a voucher V to $\mathcal{A}$ and then subsequently refuse to redeem V from $\mathcal{B}$. The key idea is that it is in the interest of both $\mathcal{A}$ and $\mathcal{B}$ to force $\mathcal{I}$ to *commit* to redeeming the voucher $V = (sn, \sigma)$. To do this $\mathcal{A}$ starts by choosing the serial number sn for the voucher and sending its hash $h = H(sn)$ to $\mathcal{B}$; notice that h hides the value of sn and thus does not harm anonymity. Then, $\mathcal{B}$ uses h to force $\mathcal{I}$ to commit to redeeming a voucher with serial number sn. Specifically, $\mathcal{B}$ asks $\mathcal{I}$ to create the transaction contract $T_{offer(V \to BTC)}$ that offers one bitcoin to $\mathcal{B}$ under the condition "$\mathcal{B}$ must provide a *valid* voucher V with serial number sn such that $h = H(sn)$ within time window tw". To prevent double-spending, $\mathcal{I}$ agrees to create $T_{offer(V \to BTC)}$ iff the hash value h does *not* match the h of any prior transaction contract that $\mathcal{I}$ has signed. Once $T_{offer(V \to BTC)}$ is on the blockchain, committing that $\mathcal{I}$ will redeem the voucher with serial number sn, our two fair exchanges proceed as in Figure 2.

The details of the scheme are as follows. Let k be the security parameter. We assume that $\mathcal{I}$ performs a one-time setup by posting public parameters on the blockchain. These parameters include the public parameters for the blind signature scheme, the fee value f and reward value w, and the time windows (tw_1, tw_2). (We define f, w below.)

1. $\mathcal{B}$ creates a fresh ephemeral Bitcoin address to receive the payment.

2. $\mathcal{A}$ randomly chooses $sn \xleftarrow{r} \{0,1\}^k$, computes $h \leftarrow H(sn)$ and sends h to $\mathcal{B}$.
3. $\mathcal{B}$ sends h to $\mathcal{I}$ and asks $\mathcal{I}$ to create transaction contract $T_{offer(V \rightarrow BTC)}$ offering one bitcoin to $\mathcal{B}$ under condition: "$\mathcal{B}$ must provide a *valid* voucher V with serial number whose hash is equal to h within time window tw_2".
4. If h does *not* match any h from prior transaction contracts signed by $\mathcal{I}$, then $\mathcal{I}$ creates the requested contract $T_{offer(V \rightarrow BTC)}$ and posts it to the blockchain.
5. $\mathcal{A}$ blinds sn to obtain $\overline{sn}$ and waits for $T_{offer(V \rightarrow BTC)}$ to be confirmed on the blockchain. Then $\mathcal{A}$ creates transaction $T_{offer(BTC \rightarrow V)}$, offering a $1 + w$ bitcoins to $\mathcal{I}$ under the condition "$\mathcal{I}$ must provide a valid blind signature on the blinded serial number $\overline{sn}$ within time window tw_1" (where $tw_1 > tw_2$ so that $\mathcal{I}$ cannot cheat by waiting until $T_{offer(BTC \rightarrow V)}$ expires but $T_{offer(V \rightarrow BTC)}$ has not).
6. To prevent $\mathcal{A}$ from double-spending the bitcoin offered in $T_{offer(BTC \rightarrow V)}$, $\mathcal{I}$ waits until the blockchain confirms $T_{offer(BTC \rightarrow V)}$. $\mathcal{I}$ then fulfills the $BTC \rightarrow V$ fair exchange by creating transaction $T_{fulfill(BTC \rightarrow V)}$ which contains the blinded signature $\overline{\sigma}$ on $\overline{sn}$. $T_{fulfill(BTC \rightarrow V)}$ is posted to the blockchain, and transfers $(1 + w)$ bitcoins from $T_{offer(BTC \rightarrow V)}$ to $\mathcal{I}$.
7. $\mathcal{A}$ learns $\overline{\sigma}$ from $T_{fulfill(BTC \rightarrow V)}$, unblinds $\overline{\sigma}$ to σ and sends $V = (sn, \sigma)$ to $\mathcal{B}$.
8. $\mathcal{B}$ creates a transaction $T_{fulfill(V \rightarrow BTC)}$ which contains the voucher $V = (sn, \sigma)$, and thus transfers the bitcoin in $T_{offer(V \rightarrow BTC)}$ to $\mathcal{B}$. $T_{fulfill(V \rightarrow BTC)}$ is posted to the blockchain and confirmed in the same block as $T_{fulfill(BTC \rightarrow V)}$.

Rewards. $\mathcal{A}$ offers $1+w$ bitcoins to $\mathcal{I}$ in $T_{offer(BTC \rightarrow V)}$, but $\mathcal{I}$ only offers $\mathcal{B}$ 1 bitcoin in $T_{offer(V \rightarrow BTC)}$. The remaining w bitcoin is kept by $\mathcal{I}$ as a "reward" for completing its role in the protocol. $\mathcal{I}$ cannot steal w because w is paid via a fair exchange.

4.1 Anonymous Fee Vouchers

Bitcoin transactions include a transaction fee that is paid to the miner who confirms the transaction in the blockchain; if this transaction fee is not paid or is too low, it is extremely unlikely that this transaction will be confirmed. Since $\mathcal{I}$ can not trust $\mathcal{B}$ or $\mathcal{A}$, $\mathcal{I}$ should not be required to cover the cost of the transaction fee for first transaction contract $T_{offer(V \rightarrow BTC)}$ that $\mathcal{I}$ posts to the blockchain.

Following ideas from [3], we have $\mathcal{A}$ buy a special *anonymous fee voucher* V' of value f bitcoin from $\mathcal{I}$. The value $f \ll 1$ should be very small and is set as a public parameter. Since fee vouchers are anonymous and have low value, $\mathcal{A}$ should buy them out-of-band in bulk with cash, credit or bitcoin. Then, whenever $\mathcal{A}$ wishes to mix or make an anonymous payment, $\mathcal{A}$ sends an anonymous fee voucher V' to $\mathcal{B}$, who in turn sends it to $\mathcal{I}$ with a request that $\mathcal{I}$ initiate the protocol. All this happens out-of-band. Note though, that the fee voucher V' is *not* created with a fair exchange, and thus $\mathcal{I}$ could steal f bitcoin by accepting V' but refusing to initiate the protocol. However, we argue that $\mathcal{I}$ has very little incentive to do this if, upon completing the protocol, $\mathcal{I}$ obtains a reward w that is significantly larger than f.

DoS Resistance. Fees raise the cost of an DoS attack where $\mathcal{B}$ starts and aborts many parallel sessions, locking $\mathcal{I}$'s bitcoins in many $T_{offer(V \rightarrow BTC)}$ transaction contracts. This is because $\mathcal{B}$ must forward an anonymous fee voucher V' from $\mathcal{A}$ to $\mathcal{I}$ every time $\mathcal{B}$ wishes to initiate our protocol. This method also works for our mixing service, where

$\mathcal{A}$ and $\mathcal{B}$ are the same user. Moreover, if a party aborts a run of our protocol during an epoch, this has no affect on other runs in that epoch. This is in contrast to [18, 10] where a single aborting player terminates the protocol for all parties in that mix.

Sybil Resistance. In a sybil attack, the adversary creates many sybil identities secretly under her control, and deanonymizes a target user by forcing the target to mix only with sybils [3, 22]. To launch this attack on our protocol, the attacker could create m runs of our protocol (*i.e.,* m payers and payees) that occupy most of the intermediary $\mathcal{I}$'s resources, leaving only a single slot available for the targeted payer and payee. Again, we use fees to raise the cost of this attack, by requiring each of the m Sybil runs to pay a fee voucher of value f. If $\mathcal{I}$ performs a sybil attack, $\mathcal{I}$ avoids paying f but must pay all four transaction fees.

4.2 Anonymity Analysis

Before discussing the anonymity properties of our scheme we start by noting that in the first step of our protocol, $\mathcal{B}$ is always required to create a fresh ephemeral Bitcoin address $Addr_0$. Upon creation, this address is *completely anonymous*, in the sense that there is no way to link it to $\mathcal{B}$'s identity; this is a much stronger notion than the set anonymity defined in Section 2.1. Now suppose that $\mathcal{A}$ uses our protocol to pay a bitcoin to $Addr_0$. Then, as we will argue below, $Addr_0$ is now linkable to $\mathcal{A}$ with probability $1/n$ (if n payments happened in that epoch). However $Addr_0$ is still completely anonymous with respect to $\mathcal{B}$'s "Bitcoin identity", *i.e.,* the long-lived Bitcoin address that $\mathcal{B}$ uses to send and receive payments. If the funds from $Addr_0$ were paid into another fresh ephemeral Bitcoin address $Addr_1$ controlled by $\mathcal{B}$, these funds would still be unlinkable to $\mathcal{B}$. Indeed, the funds in $Addr_0$ only become linkable to $\mathcal{B}$ if they are transferred to an address controlled by $\mathcal{B}$ that already contains some bitcoins.

Set-Anonymity within an Epoch. Our on-blockchain payment scheme achieves an anonymity set of size n within an epoch, as defined in Section 2.1. Suppose that n payments successfully complete during an epoch, and recall that each payer may only perform one payment per epoch and each payment is made to a fresh ephemeral address. It follows that there are n payers and n payees during the epoch. Any adversary (including $\mathcal{I}$) observing the blockchain can see the following: n payers' addresses, n payees' addresses, and n sets of transactions of the type $T_{offer(BTC\rightarrow V)}$, $T_{fulfill(BTC\rightarrow V)}$, $T_{offer(V\rightarrow BTC)}$, $T_{fulfill(V\rightarrow BTC)}$. For the adversary to link a payer to a payee, it would need to link a $T_{offer(BTC\rightarrow V)}$, $T_{fulfill(BTC\rightarrow V)}$ pair ($BTC\rightarrow V$) to a $T_{offer(V\rightarrow BTC)}$, $T_{fulfill(V\rightarrow BTC)}$ pair ($V\rightarrow BTC$). Let us first examine what do these pairs of transaction contracts reveal on the blockchain. The $BTC\rightarrow V$ pair reveals a blinded serial number $\overline{sn}$ and the corresponding intermediate (blinded) blind signature $\overline{\sigma}$. Meanwhile, the $V\rightarrow BTC$ pair reveals a serial number sn and the corresponding signature σ. As long as the blinding factor of sn is not revealed, the blind signature ensures that no one can link an sn to an $\overline{sn}$. The signatures $\overline{\sigma}$ and σ are similarly unlinkable (except with some negligible probability $\nu(k)$). Thus, the adversary's best strategy is to randomly link a payer to a payee, which succeeds with probability $1/n + \nu(k)$.

The same analysis applies to our mixing service. Moreover, mix users can repeatedly rerun the mix over several epochs, thus boosting the size of their anonymity set beyond what could be provided during a single epoch.

Resilient Anonymity and Transparency of Anonymity Set. Ephemeral addresses prevent $\mathcal{I}$ from de-anonymizing a payment from $\mathcal{A}$ to $\mathcal{B}$ by aborting or denying service. Suppose $\mathcal{I}$ aborts by refusing to issue $T_{fulfill(BTC \to V)}$ to $\mathcal{A}$ (Figure 2). If this happens, $\mathcal{A}$ does not obtain voucher $V = (sn, \sigma)$ and cannot pass V on to $\mathcal{B}$. By the unforgeability of vouchers, it follows that $\mathcal{B}$ will not be able to issue a valid $T_{fulfill(V \to BTC)}$ that fulfills $T_{offer(V \to BTC)}$. Thus, $\mathcal{I}$ can de-anonymize the payment between $\mathcal{A}$ and $\mathcal{B}$ by matching the aborted exchange with $\mathcal{A}$ with the incomplete exchange with $\mathcal{B}$. As another possible attack, malicious $\mathcal{I}$ could instead refuse service to all payers apart from a target $\mathcal{A}$, and then identify $\mathcal{B}$ by finding the single $V \to BTC$ exchange that completes during the epoch. Fortunately, however, anonymity-set transparency allows $\mathcal{B}$ to detect these attacks. $\mathcal{B}$ can recover by discarding the ephemeral address it used in the attacked epoch, and chose a fresh ephemeral address in a subsequent epoch.

Note that for both our payment and mixing service one could attempt an intersection attack as discussed in Section 2.1.

5 Off-Blockchain Anonymous Payments over Micropayment Channel Networks

We start by reviewing off-blockchain transactions via micropayment channel networks and then describe how to make our protocol faster by adapting it to work with them.

5.1 Micropayment Channel Networks

Micropayment Channels. To establish a pairwise micropayment channel, $\mathcal{A}$ and $\mathcal{B}$ each pay some amount of bitcoins into an *escrow transaction* T_e which is posted to the blockchain. This escrow transaction is on-blockchain and therefore slow ($\approx$ 10 minutes), but all subsequent transactions are off-blockchain and therefore fast ($\approx$ seconds). T_e ensures that no party reneges on an off-blockchain transaction. Suppose x bitcoins are paid into T_e. T_e offers these x bitcoins to be spent under condition: "The spending transaction is signed by both $\mathcal{A}$ and $\mathcal{B}$". Then, the spending transaction T_r has the form: "a bitcoins are paid to $\mathcal{A}$ and b bitcoins are paid to $\mathcal{B}$" where a and b reflect the agreed-upon balance of bitcoins between $\mathcal{A}$ and $\mathcal{B}$.

Once T_e is confirmed on the blockchain, $\mathcal{A}$ and $\mathcal{B}$ can transfer funds between themselves off-blockchain by signing a spending transaction T_r. Importantly, T_r is *not* posted to the blockchain. Instead, the existence of T_r creates a credible threat that either party can claim their allocated bitcoins by posting T_r to the blockchain; this prevents either party from reneging on the allocation reflected in T_r. To continue to make off-blockchain payments, $\mathcal{A}$ and $\mathcal{B}$ just need to sign a new transaction T_r' that reflects the new balance of bitcoins a' and b'. Micropayment channels have mechanisms that ensure that this later transaction T_r' always supersedes an earlier transaction T_r. Our protocol applies generically to any micropayment channel with such a mechanism, *e.g.,* Lightning Network [16], Duplex Micropayment Channels (DMC) [9].

Micropayment Channel Networks. Micropayment channel networks are designed to avoid requiring each *pair* of parties to pre-establish a *pairwise* micropayment channel

between them. Indeed, such a requirement would be infeasible, since it requires each pair of users to lock funds into many different escrow transactions T_e on the blockchain. Instead, suppose a pair of users $\mathcal{A}$ and $\mathcal{B}$ are connected by a path of users with established pairwise micropayment channels (*i.e.,* $\mathcal{A}$ has a channel with $\mathcal{A}_1$, $\mathcal{A}_1$ has a channel with $\mathcal{A}_2$, ..., $\mathcal{A}_{m-1}$ has a channel with $\mathcal{A}_m$, $\mathcal{A}_m$ has a channel with $\mathcal{B}$). Then, the path of users can run a protocol to transfer funds from $\mathcal{A}$ to $\mathcal{B}$. However, it will not suffice to simply have each user $\mathcal{A}_i$ create a transaction paying the next user $\mathcal{A}_{i+1}$ in the path, since a malicious user $\mathcal{A}_k$ could steal funds by failing to create a a transaction for $\mathcal{A}_{k+1}$. Instead, the Lightning Network and DMC use a protocol based on *hash time-locked contracts* or HTLCs. A transaction T is an HTLC if it offers bitcoins under the condition: "The spending transaction must contain the preimage of y and be confirmed within timewindow tw", where $y = H(x)$ and x is a random value, *i.e.,* the preimage. We say that T is *locked under the preimage of* y.

Micropayment channels use HTLCs as follows. Suppose the existing balance between $\mathcal{A}$ and $\mathcal{B}$ is a bitcoin for $\mathcal{A}$ and b bitcoin for $\mathcal{B}$. Now suppose that $\mathcal{A}$ wants to transfer ϵ bitcoin to $\mathcal{B}$, updating the balances to $a - \epsilon$ and $b + \epsilon$. First, $\mathcal{B}$ chooses a random value x, computes $y = H(x)$, and announces y to everyone in the path. Then, $\mathcal{A}$ asks each pair of parties $(\mathcal{A}_i, \mathcal{A}_{i+1})$ on the path to transfer ϵ bitcoin *locked under the preimage of* y using the micropayment channel from $\mathcal{A}_i$ to $\mathcal{A}_{i+1}$. The mechanics of the transfer between $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ are as follows. Suppose the existing balance between $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ is c bitcoin for $\mathcal{A}_i$ and d bitcoin for $\mathcal{A}_{i+1}$. Then $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ jointly sign a new spending transaction T'_r of the form "$c - \epsilon$ bitcoins are paid to $\mathcal{A}_i$ and $d + \epsilon$ bitcoins are paid to $\mathcal{A}_{i+1}$" under the condition "the spending transaction contains the preimage of y within timewindow tw". Once $\mathcal{A}$ sees that all the transactions on the path have been signed, it releases the preimage x to the path and the funds flow from $\mathcal{A}_i$ to $\mathcal{A}_j$. If any user refuses to sign a transaction, the timelock tw allows all signing users to reclaim their funds. The timelock is decremented along the path to prevent race conditions. This entire protocol occurs off-blockchain, with x and the HTLCs creating a credible threat that users can reclaim their funds if they are posted to the blockchain.

5.2 Anonymizing Micropayment Channel Networks

As a strawman for anonymous transactions in micropayment channel networks, we can replace the hash lock with the transaction contracts conditions in that we use in $T_{offer(V \to BTC)}$ and $T_{offer(BTC \to V)}$ (see Section 4). The protocol assumes paths of intermediate channels $(path_1, path_2)$ connecting $\mathcal{A}$ to $\mathcal{I}$ and $\mathcal{I}$ to $\mathcal{B}$ respectively, and has *Setup* phase as in our original on-blockchain protocol.

1. $\mathcal{A}$ chooses a random serial number sn, hashes it to $h = H(sn)$ and sends h to $\mathcal{B}$.
2. $\mathcal{B}$ uses h to lock a path of micropayment channels $(path_2)$ to $\mathcal{I}$ under the condition: "The spending transaction must provide a *valid* voucher V with serial number whose hash is equal to h within time window tw_2."
3. $\mathcal{A}$ blinds the serial number sn to obtain $\overline{sn}$. $\mathcal{A}$ asks $\mathcal{B}$ to confirm that each party on $path_2$ from $\mathcal{A}$ to $\mathcal{I}$ has properly locked the path. Then, $\mathcal{A}$ asks $\mathcal{I}$ to lock a path $path_1$ of micropayment channels between $\mathcal{A}_i$ and $\mathcal{I}$ under the condition: "The spending transaction must provide a valid blind signature on the blinded serial number $\overline{sn}$ within time window tw_1" where $tw_1 > tw_2$.

4. $\mathcal{I}$ then reveals $\overline{\sigma}$ to every party on $path_1$, unlocking the path from $\mathcal{I}$ to $\mathcal{A}$. $\mathcal{A}$ obtains $\overline{\sigma}$, unblinds it to σ and thus obtains the voucher $V = (\sigma, sn)$. $\mathcal{A}$ sends $V = (\sigma, sn)$ to $\mathcal{B}$ who releases it to every party on path $path2$, unlocking the path from $\mathcal{I}$ to $\mathcal{B}$.

We again need the notion of an epoch. Since we do not have blocks to coordinate these epochs, we instead use synchronized clocks. We break an epoch of q seconds into three equal divisions of $\frac{q}{3}$ seconds long. $path_2$ is set up in first division, $path_1$ is set up second and $path_1$ and $path_2$ are resolved in third division. Also, we can add anonymous fee vouchers to this protocol, since fee vouchers are redeemed out of band (Section 4.1).

5.3 Anonymity Analysis

Of the properties in Section 2.1, our off-blockchain scheme only supports set anonymity within an epoch (as discussed in Section 4.2) when $\mathcal{I}$ is *honest-but-curious*. That is, $\mathcal{I}$ follows the protocol without aborting or denying services to other payers and payees, but is still curious to learn which payer is paying which payee. However, we still support fair-exchange against a malicious $\mathcal{I}$ (Section 6) as well as set-anonymity within an epoch against malicious third parties.

We only support anonymity against honest-but-curious $\mathcal{I}$ because we cannot use fresh ephemeral addresses in this off-blockchain context. This follows because choosing a fresh address amounts to establishing a fresh micropayment channel. Because this requires a fresh escrow transaction T_e to be posted on the blockchain (taking $\approx$ 10 minutes), it obviates the speed benefits of the off-blockchain scheme. Recall that $\mathcal{B}$ discards its ephemeral address in order to recover from an epoch where a malicious $\mathcal{I}$ de-anonymized the payment from $\mathcal{A}$ to $\mathcal{B}$ by aborting or denying service (Section 4.2).

We have also given up on anonymity transparency. Because transactions are no longer posted on the blockchain, even users that participate in the protocol cannot learn the size or membership of their anonymity set.

Proxy Addresses. We need the notion of *proxy addresses* to ensure that no parties (other than $\mathcal{I}$) can break anonymity by behaving maliciously. Notice that in a micropayment channel network, a malicious user $\mathcal{A}_i$ along the path $path_1$ from $\mathcal{A}$ to $\mathcal{I}$ can abort the protocol by refusing to create the appropriate transactions. Now if $\mathcal{A}_i$ is also on the path $path_2$ from $\mathcal{I}$ to $\mathcal{B}$, then $\mathcal{A}_i$ can abort the protocol and de-anonymize $\mathcal{A}$ and $\mathcal{B}$ in the same way that $\mathcal{I}$ can. To prevent this attack, we need to make sure that $\mathcal{I}$ is the only party that is on both $path_1$ and $path_2$. The idea is that every user $\mathcal{B}$ of our system has an additional proxy address $Addr_B^{px}$, and uses this address to establish, just once, a (reusable) micropayment channel directly to $\mathcal{I}$. This ensures that $path_2$ consists of only $\mathcal{I}$ and $Addr_B^{px}$. Then, $\mathcal{B}$ will receive payments to its proxy address $Addr_B^{px}$ using the strawman protocol of Section 5.2 in a one epoch. In the subsequent epoch, $\mathcal{B}$ will rerun the strawman protocol to transfer funds from $Addr_B^{px}$ (acting as user $\mathcal{A}$) to its long-lived address $Addr_B$ (acting as user $\mathcal{B}$).

Intersection Attacks. The lack of anonymity set transparency and the use of proxy addresses implies that only $\mathcal{I}$ can observe the full membership of the anonymity set during each epoch. As payments between proxy $Addr_B^{px}$ and identity $Addr_B$ addresses occur in contiguous epochs, $\mathcal{I}$ could use an intersection attack [3] to infer their relationship. Other adversaries only observe off-blackchain transactions flowing through them.

6 Security Analysis

Fair-Exchange Our schemes prevent parties from stealing from each other.

1. The $BTC \rightarrow V$ fair exchange (Section 3) ensures that (1) $\mathcal{I}$ cannot steal $\mathcal{A}$'s bitcoin without issuing her a valid voucher V, and (2) $\mathcal{A}$ cannot refuse to pay $\mathcal{I}$ a bitcoin upon receiving a V. (Fair exchange properties (i) and (iii) from Section 2.2.)
2. The $V \rightarrow BTC$ fair exchange ensures that $\mathcal{B}$ cannot steal $\mathcal{I}$'s bitcoins without actually redeeming V. Also, $\mathcal{I}$ cannot refuse to redeem a $V = (sn, \sigma)$ that it issued to $\mathcal{A}$. This follows because, as discussed in Section 4, $\mathcal{I}$ commits to the redemption of V when it posts $T_{offer(V \rightarrow BTC)}$ (which contains h, where $h = H(sn)$). Moreover, recall that $T_{fulfill(BTC \rightarrow V)}$ is transaction where (a) $1 + w$ bitcoin are transferred from $\mathcal{A}$ to $\mathcal{I}$, and (b) $\mathcal{I}$ issues V by providing the blind signature $\overline{\sigma}$. Since $T_{offer(V \rightarrow BTC)}$ is posted to the blockchain before $T_{offer(V \rightarrow BTC)}$, it follows that $\mathcal{A}$ does not pay $\mathcal{I}$ for V until $\mathcal{I}$ has committed to redeeming V. (Fair exchange properties (ii) and (iv) from Section 2.2.)
3. $\mathcal{I}$ cannot prevent $\mathcal{B}$ from redeeming V by issuing V just before $T_{offer(V \rightarrow BTC)}$ expires. This follows because $\mathcal{A}$ choose tw_1 such that $tw_1 > tw_2$ which ensures that $T_{offer(BTC \rightarrow V)}$ expires earlier that $T_{offer(V \rightarrow BTC)}$. This way, if $\mathcal{I}$ takes too long to issue $T_{offer(BTC \rightarrow V)}$, $\mathcal{A}$ will have already reclaimed her refunds. (Fair exchange property (ii) from Section 2.2.)

Unforgeability and Double-spending. Unforgeability follows from the underlying blind signature scheme, which ensures that only the intermediary $\mathcal{I}$ can issue vouchers $V = (sn, \sigma)$. Moreover, vouchers cannot be double-spent because if $\mathcal{I}$ has previously seen $h = H(sn)$, $\mathcal{I}$ will refuse to post $T_{offer(V \rightarrow BTC)}$.

DoS and Sybil Resistance. Both our on- and off-blockchain schemes support anonymous fee vouchers, and thus resist DoS and sybil attacks (Section 4.1).

7 Conclusion

In this work we developed an eCash inspired technique that can be used to enhance anonymity in Bitcoin transactions that happen on the blockchain or via micropayment channel networks (off-blockchain). Both our schemes provide fair-exchange security, forgery and double-spending security and moreover are resistant to DoS and Sybil attacks. Regarding anonymity, our on-blockchain scheme is anonymous against malicious users or a malicious intermediary $\mathcal{I}$. Our off-blockchain scheme is still anonymous against malicious users but is only anonymous against an honest-but-curious $\mathcal{I}$. Achieving anonymity against a malicious intermediary for off-blockchain schemes is left as an interesting open problem.

Acknowledgments.

We thank Dimitris Papadopoulos, Ann Ming Samborski and the anonymous reviewers for comments on this draft. This work was funded by the National Science Foundation under grants 1012910 and 1350733.

References

1. Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better - how to make bitcoin a better currency. In *Financial Cryptography and Data Security*. Springer, 2012.
2. Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE Security and Privacy (SP)*, pages 459–474, 2014.
3. George Bissias, A Pinar Ozisik, Brian N Levine, and Marc Liberatore. Sybil-resistant mixing for bitcoin. In *Workshop on Privacy in the Electronic Society*, pages 149–158. ACM, 2014.
4. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, volume 2567, pages 31–46, 2003.
5. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, 2001.
6. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Security and Privacy (SP)*, 2015.
7. Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, JoshuaA. Kroll, and EdwardW. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *Financial Cryptography and Data Security*, 2014.
8. David Chaum. Blind signature system. In *CRYPTO*, 1983.
9. Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18. Springer, 2015.
10. G Maxwell. Coinjoin: Bitcoin privacy for the real world, 2013.
11. Gregory Maxwell. Coinswap: transaction graph disjoint trustless trading, 2013.
12. S Meiklejohn, M Pomarole, G Jordan, K Levchenko, GM Voelker, S Savage, and D McCoy. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pages 127–139, 2013.
13. Sarah Meiklejohn and Claudio Orlandi. Privacy-enhancing overlays in bitcoin. In *Financial Cryptography and Data Security*, volume 8976, pages 127–141. 2015.
14. Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE Security and Privacy (SP)*, pages 397–411, 2013.
15. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. (2012):28, 2008.
16. Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. Technical report, Technical Report (draft). https://lightning. network, 2015.
17. Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.
18. Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *ESORICS*, pages 345–364. Springer, 2014.
19. Amitabh Saxena, Janardan Misra, and Aritra Dhar. Increasing anonymity in bitcoin. In *Financial Cryptography and Data Security*, pages 122–139. Springer, 2014.
20. Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.
21. Peter Todd. Bip 65: Op checklocktimeverify. *Bitcoin improvement proposal*, 2014.
22. Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies.
23. Luke Valenta and Brendan Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In *Financial Cryptography and Data Security*, pages 112–126. Springer, 2015.
24. Jan Henrik Ziegeldorf, Fred Grossmann, Martin Henze, Nicolas Inden, and Klaus Wehrle. Coinparty: Secure multi-party mixing of bitcoins. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 75–86. ACM, 2015.