Transferable Anonymous Payments via TumbleBit in Permissioned Blockchains

Claudio Ferretti[1], Alberto Leporati[1], Luca Mariot[1], and Luca Nizzardo[2]

[1] DISCo, University of Milano-Bicocca, Milano, Italy
{claudio.ferretti, alberto.leporati, luca.mariot}@unimib.it
[2] IMDEA Software Institute, Madrid, Spain
luca.nizzardo@imdea.org

Abstract

We propose a modification to the TumbleBit protocol, the off-chain payment hub enabling anonymous transactions among users in the Bitcoin network. Specifically, we modify the first step of the protocol by making the tumbler node sending a P2SH transaction on the blockchain claiming that any user can redeem 1 Bitcoin by providing a SHA-2 preimage of a value chosen by the Tumbler. We remark that our modification enables Bob to *transfer* its Bitcoin to a third user, and argue that this modified TumbleBit protocol could find applications in permissioned blockchains, for example in the context of anonymous payments between different banks or fintech companies.

1 Introduction

The advent of Bitcoin back in 2008 [21] gave people the perception of being able to exchange value over the web in a trustless and anonymous way. Nevertheless, it is now widely known that Bitcoin anonymity properties are weaker than initially expected, as underlined in several recent works [16, 26]. These anonymity weaknesses opened the way to a research effort devoted to provide anonymity-enhancing services for Bitcoin, like [30, 11, 20]. In a few words, the aim of these works is to mask the addresses of payers and payees among a set of different payers/payees, so that each pair $(payer, payee)$ is difficult to sort out.

Once dealing with such a problem, one can simply employ a trusted mixer of transactions, in a way that as soon as the mixer behaves honestly, no data is leaked. But how can we be sure that such a mixer is not spoofing clients data for some reason? In a way, an optimal solution would guarantee anonymity even with respect to the mixer itself. Moreover, apart from preserving privacy, we must also avoid that a greedy mixer steals coins while processing the mixing of transactions. In general, we want to provide a service where a mixer cannot link transactions between payer and payee and, on top of that, he is prevented from stealing coins.

An interesting solution to this problem was proposed by Heilman et. al in [14], with the introduction of *TumbleBit*. TumbleBit is a unidirectional unlinkable payment hub that uses an untrusted intermediary, called *Tumbler*, to enhance anonymity. In TumbleBit, payments are backed in Bitcoin, and the tumbler cannot break users' anonymity, nor steal users' Bitcoins, nor create money and send it to itself. Moreover, TumbleBit improves on scalability, since payments are enforced by off-chain interactions between payer, payee and tumbler. Specifically, on-chain operations are only involved in two points of the protocol, namely during the setup of the channel payments and the cash-out phase.

In their original form, blockchains have been conceived as distributed data structures which may be used to enforce public verifiability among a set of mutually distrusting parties, without the need to resort to a trusted third party. Hence, blockchains such as that of Bitcoin are *permissionless* by design, meaning that any party can join the network and can read and write

CEUR-WS.org/Vol-2334/DLTpaper5.pdf

transactions over the blockchain. Recently *permissioned* blockchains have also been proposed, where read and write access are granted by the network administrators only to certain authorized actors. Examples of permissioned blockchains frameworks include *Hyperledger Fabric* [2], *Corda* [6], and *MultiChain* [13]. A natural question is whether permissioned blockchains are actually useful in any real-world use case. As a matter of fact, the ability to select authorized users who can interact with the blockchain implicitly assumes that they are not mutually distrustful – thus rising the legitimate doubt that any application domain where a permissioned blockchain is proposed can be addressed by traditional centralized databases approaches. Wüst and Gervais [31] investigated this issue in depth, describing several scenarios where it makes sense to use a permissioned blockchain, including *interbank payments*, *supply chain management* and *decentralized autonomous organizations*.

Depending on the underlying use case, anonymity can be an important feature also in permissioned blockchains. One example is the aforementioned scenario of interbank payments, where different banks needs to transfer financial value between them or their customers. As noted in [31], this problem is usually addressed through the involvement of a central bank that acts as a trusted third party. Using a permissioned blockchain in this case could simplify the whole payment process. In fact, the central bank would only behave as a trusted authority giving read and write access on the blockchain to the banks participating in the system, and it would not be involved in the verifiability of the transactions. It seems reasonable, however, that banks taking part in this permissioned blockchain would like to keep their monetary flows anonymous. Of course, the actors involved in this application scenario could also be any kind of organizations other than banks that wish to exchange money among them while holding in high regard the privacy of their transactions, such as *fintech companies* [10]. Consequently, the use of tumbler nodes to anonymize transactions among users is also motivated in the context of permissioned blockchains.

In this paper we propose a modification of the TumbleBit protocol which enables the anonymous transfer of a token to a third party. In particular, the tumbler node is involved in the initial transfer of the token between Alice and Bob, as per the original protocol. However, Bob can subsequently forward the received Bitcoin to a third player, Charlie, without the necessity of interacting again with the Tumbler, while the anonymity of the whole transaction is preserved. In order to accomplish this feature, we modify the initial part of the TumbleBit protocol by using a *Pay-To-Script-Hash* (P2SH) transaction, whose escrow condition consists in providing a pair of SHA-2 preimages respectively chosen by the Tumbler and Bob. Contrarily, in the original protocol a multisignature escrow transaction is posted on the blockchain, which requires both the Tumbler's and Bob's signatures to redeem the escrowed Bitcoin. We remark that this modification allows Bob to anonymously transfer the token received from Alice to Charlie without needing further interaction with the Tumbler, due to the fact that our P2SH transaction does not bind the receiver to a specific address, as in the case of the 2-of-2 escrow transaction used in the original protocol. However, we also note that using our modified protocol in permissionless blockchains is not secure, since a malicious miner could steal Bitcoins by *mauling* the modified P2SH transaction [3]. Generally speaking, mauling is an attack where an adversary who knows a transaction T of a user can construct a second transaction which is semantically equivalent but syntactically different. In our case, mauling can occur in permissionless blockchains since a dishonest miner can take Charlie's transaction which contains the pair of preimages necessary to redeem the Bitcoin, and include them in a new transaction where the recipient address is that of the miner. Hence, our protocol can be adopted to enable transferable anonymous payments only in a permissioned setting where the validators nodes are trusted, such as in the interbank payment scenario mentioned above.

The rest of this paper is structured as follows. Section 2 covers all necessary background definitions and notions about Bitcoin transactions, which will be used in the description of our protocol modification. In Section 3 we give a general survey of the anonymization approaches in the existing literature. Section 4 focuses on the original TumbleBit protocol, while Section 5 describes our modification to the first escrow phase and discusses its main properties, namely the possibility of a further anonymous transfer without involving the Tumbler. Finally, in Section 6 we sum up the main contribution of this paper, remarking its limitations in a permissionless setting, and discussing its possible application in permissioned blockchains.

2 Background on Bitcoin Transactions

We now recall the basic elements and operations involved in Bitcoin transactions, focusing in particular on those that we will use in the description of the TumbleBit protocol. For further information on the subject, we refer the reader to [22].

In the Bitcoin network, each user $\mathcal{U}$ is identified by a *public address* $PK_{\mathcal{U}}$, which corresponds to an ECDSA public key. The amount of Bitcoins a user possesses is determined by a set of *transactions*, all of which are registered on the Blockchain, and which specify the sum of Bitcoins that have been currently transferred to that user.

In particular, each transaction is composed of multiple *inputs* and *outputs*, which are respectively payers' and payees' public addresses. Each output in a transaction T_1 can be transferred only to a single input of another transaction T_2. In particular, *double-spending* the quantity of Bitcoins contained in the output of T_1 would require having two transactions T_2 and T_3, each pointing to the output of T_1 in one of their inputs. However, this event is prevented by the security properties of the Bitcoin protocol, which under certain assumptions (i.e., that an attacker cannot control more than 50% of the miners in the network) ensures that double-spending transactions are not added to the Blockchain.

Bitcoin transactions can be of different types, and they can be specified in the *Script* language. In particular, the transactions involved in the TumbleBit protocol are the following:

- *Offer Transactions*, T_o: in these transactions, a payer $\mathcal{A}$ commits to pay a certain quantity of Bitcoins to any other party in the Bitcoin network who is able to sign another transaction satisfying a certain condition C. Transaction T_o is also signed by $\mathcal{A}$.

- *Fulfill Transactions*, T_f: this is the transaction that a payee $\mathcal{B}$ must produce and post on the Blockchain in order to redeem the Bitcoins offered to him by $\mathcal{A}$ in an offer transaction T_o. In particular, T_f contains the public key of $\mathcal{B}$, points to the output of T_o, and contains a predicate which satisfies condition C in T_o. Transactions T_f is also signed by $\mathcal{B}$.

The typical workflow of an offer-fulfill transaction pair is as follows: first, the offer transaction T_o is registered on the Blockchain. Second, when the fulfill transaction T_f is also validated on the Blockchain, the Bitcoins specified in T_o are transferred from $\mathcal{A}$ (i.e. the entity who signed T_o) to $\mathcal{B}$ (the entity who signed T_f).

In the *Script* language, both types of transactions can be specified by $\mathcal{A}$ and $\mathcal{B}$ using the *Pay-To-Script-Hash* (P2SH) format, originally proposed in the Bitcoin Improvement Proposal BIP16 [1]. More specifically, in a P2SH transaction $\mathcal{A}$ stores in her T_o offer transaction the hash of a *redeem script*, which contains the condition C to be satisfied in order to validate the fulfill transaction. On the other hand, $\mathcal{B}$ generates the fulfill transaction T_f by including the redeem script of the corresponding T_o and a set of input values that are fed to the script. If the hashed version of the redeem script in T_f equals the one contained in the offer transaction T_o,

then the redeem script is run over the set of input values, and condition C is met if and only if the script returns true as output value.

The condition C that must be satisfied by the fulfill transaction can also be of different types. In particular, the two basic types of conditions employed in the TumbleBit protocol and in our modification are the following:

- *Hashing condition*: given a cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^m$ and a value $y \in \{0,1\}^d$, the condition C in T_o states that T_f must report a preimage $x \in H^{-1}(y)$ of y, that is, a value $x \in \{0,1\}^*$ such that $H(x) = y$. In what follows, we will assume that the underlying hash function is SHA-2 [23], with length of the message digest fixed to $d = 256$ bits.

- *Signing condition*: given a digital signature scheme $\mathcal{S} = \langle Sgn, Ver, \mathcal{K}_E, \mathcal{K}_D \rangle$, condition C in T_o specifies that the signature S of the fulfill transaction T_f must verify under a public key $PK \in \mathcal{K}_D$, that is, $Ver_{PK}(T_f) = S$. As in the case of TumbleBit, we will assume in the following that the digital signature scheme is *ECDSA* over the *Secp256k1* elliptic curve [25], in order to make it compatible with the Bitcoin scripting language.

The hashing and signing conditions can also be composed by requiring that *two* preimages/signatures are provided to fulfill the offer transaction, as described below:

- *Double-Hashing condition*: given a cryptographic hash function H as in the case of the hashing condition and a pair of values $(w, z) \in \{0,1\}^d$, condition C is met if and only if the fulfill transaction T_f contains a pair of values $(x, y) \in \{0,1\}^*$ such that $H(x) = w$ and $H(y) = z$.

- *2-of-2 Escrow condition*: This condition is built on top of the signing condition described above. In particular, a user puts in *escrow* a certain amount of Bitcoins, and two signatures S_1 and S_2 are required to be verified in order to redeem the Bitcoins, one under a public key PK_1 and the other on a second public key PK_2.

Remark also that Bitcoin transactions can be *time-locked*: for example, in the offer transaction T_o Alice can specify a *time window*, tw, before which the condition C must be satisfied. If the time window tw expires without a corresponding fulfill transaction T_f being registered on the Blockchain, the Bitcoins put in escrow for transaction T_o can be reclaimed by Alice. In the Bitcoin network, time windows can be specified in number of blocks added to the Blockchain, since each new block is appended approximately every 10 minutes.

3 Related Work

We now give a broad overview of the literature concerning the anonymization of transactions over the Bitcoin network. Starting from TumbleBit, which is the main reference for our modified protocol and which will be described in detail in the next section, we then cover the other main approaches to anonymization, namely micropayment channels, mixers, anonymous cryptocurrencies, and fair exchange protocols.

TumbleBit. The TumbleBit protocol, originally proposed in [14], assumes that there are a payer Alice ($\mathcal{A}$), a payee Bob ($\mathcal{B}$) and a Tumbler ($\mathcal{T}$). Alice wants to pay 1 Bitcoin to Bob but she does not want to send her payment directly on the Blockchain; instead, she uses a Tumbler to get anonymity. The way the whole procedure works is basically the following: the Tumbler

issues a transaction of 1 Bitcoin to Bob, and the transaction is conditioned to the solution of a puzzle p. Bob and the Tumbler interact to create a puzzle p, which is modified by Bob into a puzzle p' and then sent to Alice. Alice receives p' and issues a transaction of 1 Bitcoin to the Tumbler which is conditioned to the solution of the puzzle p'. The Tumbler receives 1 Bitcoin by giving to Alice the solution s' of p'. Alice then sends s' to Bob, who derives from it a solution s of p. Now Bob also gets 1 Bitcoin and the payment is finalized.

Payment Channels. Other approaches to the same anonymity problem can be found in micropayment channels: the most famous solution is the one known as the *Lighting Network*, proposed by Poon et al. in [24] and recently implemented by Blockstream; another example is *Duplex Micropayment* channels, proposed in [8] by Decker et al. This systems consists in an initial on-blockchain pairwise escrow which is then updated offline via paths of intermediaries. It is similar in spirit to what TumbleBit does, but there is a main difference, which relies in the fact that intermediaries in micropayment channels can link payments, while the tumbler in TumbleBit cannot.

In [15] Heilman et al. proposed a way to ensure anonymity in micropayment channel networks; nevertheless, that work is not compatible with Bitcoin. Moreover, an interaction between payer, payee and tumbler is required every time for each off-chain payment, allowing a malicious tumbler to correlate payer and payee by correlating the timing of interactions.

Anonymous Cryptocurrencies. In order to overcome Bitcoin anonymity issues, different privacy preserving cryptocurrencies have been recently introduced: two interesting examples are *Monero*, which makes use of stealth addresses and group signatures, and *Zcash*, which employs shielded transactions to preserve users' anonymity via ZK-Snarks. Moreover, on top of Zcash we have an off-chain unlinkable payment channel called *Bolt*, proposed by Green and Miers [12].

Prior Work on Bitcoin Tumblers. Other examples of Bitcoin compatible tumblers are *Mixcoin* [5] and the subsequent *Blindcoin* [29], where a trusted third party is employed in order to mix Bitcoin addresses. In both of them, coin theft can be detected but not avoided, and in Mixcoin a malicious mixer has the power to break users' anonymity as well.

Apart from those, other known mixers are *CoinSwap* [18] and *Coinparty* [32]: the former allows two users to exchange bitcoins using an intermediary, which can link payer and payee but is prevented from stealing coins through fair exchange. The latter provides a secure solution if and only if at least 2/3 of the users are honest.

Other solutions for single transactions are provided by *CoinShuffle* [27] and *CoinShuffle++* [28], both built on top of *CoinJoin* [17]. Beside performing a mix in a single transaction, leading to scalability problems, these systems has been shown to be vulnerable to denial of service attacks, meaning that it is possible for a malicious user to join the protocol and then abort the protocol for the whole set of involved users.

Fair Exchange Protocols over the Blockchain. A similar problem is the so called *Zero Knowledge Contingent Payment* over the Blockchain. Here a payer commits to pay a fixed amount (let's say 1 Bitcoin) to a payee, if and only if the payee provides the evaluation of any function f over which the two parties agreed at the beginning of the protocol. This kind of protocols can be seen as a fair exchange which uses the Bitcoin Blockchain as a trusted third party. Intuitively, the payee evaluates the function f and encrypts the result using a key k. The key is then hashed as $h = H(k)$. The payee receives a ciphertext c, an element h and a zero-knowledge proof that ensures the output of the function is encoded in c using k, and that h is the hash of k via H. At this point the payer issues a transaction offering 1 Bitcoin under the condition of presenting an hash preimage of h. Once the payee publishes a preimage

k, he directly redeems the transaction while the payer is now able to decrypt the ciphertext containing the output of f over inputs of her choice. A first attempt to implement this solution was due to Maxwell [19], and was then broken and expanded by Campanelli et al. [7]. A similar result was obtained by Basanik et al. [4]. Recently, Dziembowski et al. [9] proposed a new interesting solution which does not use expensive cryptographic tools such as zero-knowledge proofs.

4 TumbleBit Protocol

Using the Bitcoin primitives described in Section 2, we now give a more detailed description of the TumbleBit protocol proposed in [14]. The protocol involves three actors: a payer *Alice* $\mathcal{A}$, a payee *Bob* $\mathcal{B}$, and a *Tumbler* $\mathcal{T}$. Alice wants to send Bob 1 Bitcoin, without anyone being able to trace back this transaction to the public keys of $\mathcal{A}$ and $\mathcal{B}$. Normally, Alice could simply issue a payment from her public address to Bob's public address, but this event would be registered on the Bitcoin Blockchain, which is public, and thus anyone could easily link payer and payee, by using for instance the techniques described in [16, 26]. Hence, both Alice and Bob need to interact with the Tumbler in order to anonymize the transaction.

As a preliminary remark, in what follows we assume that each payment consists of 1 Bitcoin. This is a necessary condition in order to ensure anonymity: as a matter of fact, if the transactions processed by the Tumbler were of different amounts, anyone could link together the public keys of payers and payees by simply looking on the Blockchain the incoming and outcoming transactions posted by the Tumbler.

At a glance, the TumbleBit protocol can be divided in three phases, namely:

1. *Escrow phase*: in this phase, the payment channels between $\mathcal{A}$ and $\mathcal{T}$ and between $\mathcal{T}$ and $\mathcal{B}$ are set up. Moreover, both $\mathcal{A}$ and $\mathcal{T}$ put 1 Bitcoin in escrow.

2. *Payment phase*: in this phase, $\mathcal{A}$ and $\mathcal{B}$ interact with $\mathcal{T}$ to solve a cryptographic puzzle. In particular, the interaction between $\mathcal{T}$ and $\mathcal{B}$ is used to generate the puzzle, while the interaction between $\mathcal{A}$ and $\mathcal{T}$ is needed in order to solve the puzzle.

3. *Cash-out phase*: in the third phase, when the puzzle has been solved through the interaction of the three parties, all payment channels are closed. The Tumbler redeems 1 Bitcoin from Alice, while Bob redeems 1 Bitcoin from the Tumbler.

In particular, observe that only the escrow and cash-out phases require transactions to be registered on the Bitcoin Blockchain. Contrarily, all the steps in the intermediate Payment phase are performed off-chain. As described in [14], this feature provides a good scalability for the TumbleBit protocol, that can be used for fast off-chain payments.

Additionally, using a trick similar to the one enabling time-locked transactions mentioned in Section 2, the beginning and the end of each phase can be marked by a fixed amount of new blocks appended on the Blockchain, which have constant frequency. Hence, each of the three players knows exactly when each phase begins and when ends.

In what follows, we discuss each phase of the protocol, describing the interactions between Alice, Bob and the Tumbler node.

4.1 Escrow Phase

The protocol is initiated when the payer Alice $\mathcal{A}$ wants to send 1 Bitcoin to the payee Bob $\mathcal{B}$. The escrow phase is composed of three steps:

(1) In the first step, the payee $\mathcal{B}$ asks the Tumbler $\mathcal{T}$ to open a payment channel. To do so, the tumbler T posts on the Blockchain a 2-of-2 escrow transaction $T_{\mathcal{T},\mathcal{B}}$ which escrows 1 Bitcoin, where the condition is the following:

Escrow condition $C_{\mathcal{T},\mathcal{B}}$: Bob $\mathcal{B}$ can claim 1 Bitcoin by providing the two signatures $S_{\mathcal{T}}$ and $S_{\mathcal{B}}$, which verify under the public keys $PK_{\mathcal{T}}$ and $PK_{\mathcal{B}}$ respectively of $\mathcal{T}$ and $\mathcal{B}$.

Notice in particular that the transaction $T_{\mathcal{H}_{R,S}}$ is time-locked to a time window tw_2, after which the tumbler $\mathcal{T}$ can claim back its Bitcoin.

(2) On the other hand, in the second step Alice $\mathcal{A}$ opens a payment channel to $\mathcal{T}$ and escrows 1 Bitcoin by registering another 2-of-2 escrow transaction $T_{\mathcal{A},\mathcal{T}}$ on the Blockchain, where the condition to be fulfilled is as follows:

Escrow condition $C_{\mathcal{A},\mathcal{T}}$: 1 Bitcoin can be claimed by $\mathcal{T}$ by presenting two signatures $S_{\mathcal{A}}$ and $S_{\mathcal{T}}$ which verify respectively under the public key $PK_{\mathcal{A}}$ of $\mathcal{A}$ and under the public key $PK_{\mathcal{T}}$ of $\mathcal{T}$.

Similarly to the transaction $T_{\mathcal{T},\mathcal{B}}$ described above, transaction $T_{\mathcal{A},\mathcal{T}}$ is time-locked to a time window $tw_1 < tw_2$, after which Alice can claim back her Bitcoin if the escrow condition has not been fulfilled.

(3) In the third step of the escrow phase, the Tumbler and Bob engage in a *puzzle-promise protocol*, after which $\mathcal{B}$ receives from $\mathcal{T}$ a pair of values (c, z), where c is the encryption of the Tumbler's signature $S_{\mathcal{T}}$:

$$c = \mathsf{Enc}_{\varepsilon}(S_{\mathcal{T}}) \ . \tag{1}$$

In particular, Enc is a symmetric encryption algorithm (for example, AES), whose encryption key ε is randomly chosen by $\mathcal{T}$. On the other hand, z is the RSA encryption of the symmetric key ε under $\mathcal{T}$'s public key $PK_{\mathcal{T}} = (e, N)$:

$$z = \mathsf{RSA}(\varepsilon, e, N) = \varepsilon^e \mod N \ . \tag{2}$$

The puzzle-promise protocol is used to ensure that the Tumbler cannot act dishonestly by sending Bob a value c which does not correspond to the encryption of its signature $S_{\mathcal{T}}$. For the details of this protocol, which is based on the *cut-and-choose* technique, we refer the reader to [14].

4.2 Payment Phase

After the setup of the payment channels is done, Alice and Bob can proceed to the payment phase, which is composed of the following steps:

(1) Bob samples a random element $\alpha \in \mathbb{Z}_N^*$, and uses it to *re-randomize* the puzzle z received from $\mathcal{T}$, by computing

$$z' = \alpha \cdot z \tag{3}$$

This step is also called *blinding*. Next, Bob sends z' to Alice.

(2) After receiving z' from Bob, Alice and the Tumbler engage in a *puzzle-solving protocol*, whose detailed description can be found [14]. Essentially, the goal of $\mathcal{A}$ in this step is to obtain from $\mathcal{T}$ the solution of the puzzle z' to send back to $\mathcal{B}$. On the other hand, $\mathcal{T}$ wants $\mathcal{A}$ to sign the fulfill transaction associated to $\mathcal{T}_{\mathcal{A},\mathcal{T}}$, in order to redeem her Bitcoin escrowed in the first phase. Similarly to the puzzle-promise protocol mentioned in the third step of Section 4.1, the

puzzle-solving protocol ensures that the solution obtained by Alice from the Tumbler is indeed the solution ε' of the blinded puzzle sent by Bob, that is,

$$\varepsilon' = \mathsf{RSA}^{-1}(z', d, N) = z'^{d} \mod N \ , \tag{4}$$

where d is the private key of $\mathcal{T}$. Further, the puzzle-solving protocol guarantees that the Tumbler receives Alice's signature $S_{\mathcal{A}}$ by providing the correct solution of the puzzle z'.

(3) Finally, once the puzzle-solving protocol is completed, Alice sends back the solution of the blinded puzzle ε' to Bob, who can unblind it by computing

$$\varepsilon = \frac{\varepsilon'}{\alpha} \ . \tag{5}$$

Additionally, in order to be sure that Alice sent a correct solution, Bob checks the obtained value ε by verifying that $\varepsilon^e = z \mod N$.

4.3 Cash-out Phase

The cash-out phase, which closes all payment channels opened in the escrow phase, unfolds through the following steps:

(1) $\mathcal{B}$ decrypts the value c given to him by $\mathcal{T}$, thus obtaining its signature $S_{\mathcal{T}}$:

$$S_{\mathcal{T}} = \mathsf{Enc}_{\varepsilon}^{-1}(c) \ . \tag{6}$$

Consequently, Bob is able to satisfy the escrow condition of the 2-of-2 escrow transaction $T_{\mathcal{T},\mathcal{B}}$, since he now possesses both his signature and the Tumbler's. Hence, Bob posts on the Blockchain a fulfill transaction $T_{f(\mathcal{T},\mathcal{B})}$, and retrieves the Bitcoin escrowed by the Tumbler.

(2) On the other hand, after the puzzle-solving protocol in the second step of the payment phase the Tumbler successfully received Alice's signature $S_{\mathcal{A}}$. Hence, $\mathcal{T}$ posts a fulfill transaction $T_{f(\mathcal{A},\mathcal{T})}$ on the blockchain, and redeem Alice's Bitcoin.

5 Modification of the Protocol

Suppose now the following scenario: after Bob successfully received the solution of the blinded puzzle z' from Alice, he wishes to anonymously forward to Charlie $\mathcal{C}$ the Bitcoin escrowed by the Tumbler, but without interacting further with it. This functionality cannot be achieved in the original TumbleBit protocol: as described in step (1) of the escrow phase, the Tumbler posts a 2-of-2 escrow transaction on the Blockchain, which requires both the Tumbler's and Bob's signatures to be fulfilled. The problem stems from the fact that this transaction binds the recipient to provide a signature that verifies under Bob's public key. Hence, Bob and Charlie should engage with the Tumbler in another round of the TumbleBit protocol.

To address the above problem, we now describe our modification to the TumbleBit protocol. Specifically, the modified part concerns the first step of the escrow phase as follows:

(1) **New escrow step:** As in Section 4.1, this step is initiated when the payee $\mathcal{B}$ asks the Tumbler $\mathcal{T}$ to open a payment channel. However, this time $\mathcal{B}$ samples a random string $r \xleftarrow{\$} \{0,1\}^*$ with uniform probability, computes the hash value $R = \mathsf{SHA256}(\mathsf{r})$, and finally sends R to $\mathcal{T}$. The tumbler T, on the other side, samples a value $s \xleftarrow{\$} \{0,1\}^*$, computes $S = \mathsf{SHA256}(\mathsf{s})$ and posts on the Blockchain an offer transaction $T_{\mathcal{H}_{R,S}}$ which escrows 1 Bitcoin, where the condition is the following:

Escrow condition $C_{\mathcal{H}_{R,S}}$: 1 Bitcoin can be claimed by any recipient who provides SHA256 preimages of both R and S, that is, two values $x, y \in \{0,1\}^*$ such that $\mathsf{SHA256}(x) = R$ and $\mathsf{SHA256}(y) = S$.

After this step, the protocol proceeds as in the original version, by changing the relevant steps where the elements of the offer transaction $T_{\mathcal{H}_{R,S}}$ are involved. In particular, in the third step of the escrow phase the output of the puzzle-promise protocol is the pair (c, z) where c is the hash value S computed by the Tumbler, while z is an RSA encryption of its preimage s:

$$c = S = \mathsf{SHA256}(s) \tag{7}$$

$$z = \mathsf{RSA}(s, e, N) = s^e \mod N \ , \tag{8}$$

where (e, N) is again the Tumbler's public key. The payment phase unfolds exactly as in the original version of the protocol: Bob blinds the RSA puzzle z by multiplying it with a random value $\alpha \in \mathbb{Z}_n^*$ and sends it to Alice, and the puzzle-solving protocol between Alice and the Tumbler proceeds in the same way. Once Bob gets back the solution from Alice, he unblinds it by dividing it by α. At this point, Bob has obtained the preimage s of S under SHA256, and he can fulfill the offer transaction posted by the Tumbler on the Blockchain, since he possesses both his preimage r and the Tumbler's preimage s.

However, notice in particular that, in the redeem script of transaction $T_{\mathcal{H}_{R,S}}$, the Tumbler does not need to bind the recipient providing the preimages to a specific address. Hence, *any* user who is able to give the SHA256 preimages of R and S can claim 1 Bitcoin from $\mathcal{T}$, not necessarily $\mathcal{B}$. If we assume that Bob wants to forward 1 Bitcoin to Charlie $\mathcal{C}$, he can simply send $\mathcal{C}$ the pair (r, s), and Charlie can successively redeem $\mathcal{T}$'s Bitcoin by posting the relevant fulfill transaction on the Blockchain.

Remark that the anonymity properties of the original TumbleBit protocol are preserved, that is, the Tumbler is not able to link the sequence of payments between Alice, Bob and Charlie. In fact, the first payment between Alice and Bob is protected by the anonymity properties of the original TumbleBit protocol, since the only modification that we introduced (i.e. the use of a hashing condition instead of 2-of-2 escrow) does not impact the RSA blinding and the puzzle-promise and puzzle-solving protocols, which are the crucial elements of TumbleBit anonymity. On the other hand, the anonymity of the payment between Bob and Charlie follows from the fact that, since the pair (r, s) is transmitted off-chain, there is no trace on any interaction between the two users on the blockchain.

6 Conclusions

In this paper, we introduced a modification of the TumbleBit protocol to enable an additional anonymous transfer of a Bitcoin to a third user, without involving the Tumbler. In particular, this property is achieved by employing a P2SH transaction in the first escrow step: instead of using the original 2-of-2 multisignature transaction, the Tumbler posts on the Blockchain a P2SH transaction whose escrow condition requires providing the preimages of two SHA256 values, respectively computed by the Tumbler itself and by Bob. Since this condition does not bind the recipient of the transaction to a specific address, Bob can forward the pair of preimages to a third user Charlie, who can in turn redeem the Bitcoin by registering the corresponding fulfill transaction on the blockchain. Since the preimages are forwarded off-chain, the anonymity properties of the original TumbleBit protocol are also preserved on this additional payment, that is, the Tumbler is not able to link Bob and Charlie together.

Nevertheless, we remark that this modified protocol cannot be securely used without relying on strong hypotheses, which are usually not satisfied in permissionless blockchains, such as in the Bitcoin case. Indeed, if the receiver of a transaction is not specified, any malicious miner can take the pair of preimages (r, s) from Charlie's transaction and include it in a new transaction where the receiver's address is its own one. This problem is known in the literature as *mauling* [3]. So, in order to make this protocol work securely, one should assume that miners are not willing to do any mauling. Although this is a rather restrictive assumption for a permissionless setting, it is much more reasonable in permissioned blockchains, since in this case the validators nodes are all known and can be trusted to a sufficiently high degree.

Thus, we believe that this modified TumbleBit protocol can find interesting applications to enable anonymous payments in permissioned blockchains. As mentioned in the Introduction, one possible scenario is that of interbank payment systems, where different banks (or fintech companies) wants to anonymously exchange financial value. The use of anonymous payments in this setting is motivated by the fact that each bank or fintech company, usually, wishes to keep its transactions private, notwithstanding the fact that in a permissioned blockchain a certain degree of trust is usually assumed. Moreover, an additional advantage with respect to the availability of the system is that our modified protocol does not require any interaction with the Tumbler in the second payment between Bob and Charlie: hence, if the Tumbler node is not always online to provide its anonymization service, the anonymous transfer of the token between Bob and Charlie can still be performed.

As a closing remark, observe that the original TumbleBit protocol has been designed to be compatible with the Bitcoin network, and in particular with its *Script* language for specifying transactions. This raises the question of how to implement our modified protocol in a permissioned setting, since the scripting languages used in frameworks such as *Hyperledger Fabric* [2] are rather different than *Script*. We propose two possible approaches to address this problem. First, a straightforward solution would be to adopt a permissioned blockchain framework such as *MultiChain* [13], whose scripting languages are built on top of Bitcoin *Script*, and are thus compatible with it. On the other hand, an interesting alternative to consider would be to adapt our modified protocol to other scripting languages that are able to replicate the properties of Bitcoin transactions. *Hyperledger* could constitute an interesting candidate for developing this idea, since its scripting language is *Turing-complete*, and can thus be used in principle to simulate transactions written in *Script*. We plan to investigate this issue as a future direction of research.

References

[1] G. Andresen. Bip-0016: Pay to script hash, 2014. https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki.

[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick. Hyperledger fabric: A distributed operating system for permissioned blockchains. *CoRR*, abs/1801.10228, 2018.

[3] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. On the malleability of bitcoin transactions. In *Financial Cryptography and Data Security - FC 2015 International Workshops, BITCOIN, WAHC, and Wearable, San Juan, Puerto Rico, January 30, 2015, Revised Selected Papers*, pages 1–18, 2015.

[4] W. Banasik, S. Dziembowski, and D. Malinowski. Efficient zero-knowledge contingent payments in cryptocurrencies without scripts. In *European Symposium on Research in Computer Security*, pages 261–280. Springer, 2016.

[5] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.

[6] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn. Corda: An introduction. *R3 CEV, August*, 2016.

[7] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 229–243. ACM, 2017.

[8] C. Decker and R. Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[9] S. Dziembowski, L. Eckey, and S. Faust. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 967–984, New York, NY, USA, 2018. ACM.

[10] I. Eyal. Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities. *IEEE Computer*, 50(9):38–49, 2017.

[11] Grams. Helixlight: Helix made simple. https://grams7enufi7jmdl.onion.to/helix/light., 2016.

[12] M. Green and I. Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489. ACM, 2017.

[13] G. Greenspan. Multichain private blockchainwhite paper. *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 2015.

[14] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*, 2017.

[15] E. Heilman, F. Baldimtsi, and S. Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 43–60. Springer, 2016.

[16] S. M. M. P. G. Jordan, K. L. D. McCoy, and G. M. V. S. Savage. A fistful of bitcoins: Characterizing payments among men with no names. 2013.

[17] Maxwell. Coinjoin: Bitcoin privacy for the real world. https://bitcointalk.org/index.php?topic=279249.0, 2013.

[18] Maxwell. Coinswap: transaction graph disjoint trustless trading. https://bitcointalk.org/index.php?topic=321228.0, 2013.

[19] Maxwell. The first successful zero-knowledge contingent payment. https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/, 2016.

[20] M. Möser and R. Böhme. Join me on a market for anonymity.

[21] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system.

[22] A. Narayanan, J. Bonneau, E. W. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction*. Princeton University Press, 2016.

[23] N. I. of Standards and Technology. Specifications for the secure hash standard, 2002. https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf.

[24] J. Poon and T. Dryja. The bitcoin lightning network.

[25] C. Research. Sec 2: Recommended elliptic curve domain parameters, 2010. http://www.secg.org/sec2-v2.pdf.

[26] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[27] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing

for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014.

[28] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle++, a fast peer-to-peer coin mixing protocol. https://bitcointalk.org/index.php?topic=1497271, 2016.

[29] L. Valenta and B. Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 112–126. Springer, 2015.

[30] Wikipedia. Bitcoin fog. https://en.wikipedia.org/wiki/Bitcoin_Fog, 2016.

[31] K. Wüst and A. Gervais. Do you need a blockchain? *IACR Cryptology ePrint Archive*, 2017:375, 2017.

[32] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. 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.