
DLT Workshop

Pisa – February 12th, 2019


# Transferable Anonymous Payments via TumbleBit in Permissioned Blockchains

**Claudio Ferretti** [1], **Alberto Leporati** [1]

**Luca Mariot** [1], **Luca Nizzardo** [2]

e-mail: alberto.leporati@unimib.it

[1] Dip. Informatica, Univ. Milano – Bicocca, Italia

[2] IMDEA Software Institute, Madrid, Spain

# Goal of this work

- **Goal:** to allow
  - Alice (A, the payer) to pay 1 BTC to Bob (B, the payee)
  - later, B passes his BTC to Charlie (C)
  - nobody is able to link the addresses of A, B and C by looking at the transactions stored in the blockchain (payments are anonymous)
- This could be done by using a trusted mixer, that always behave honestly:
  - A gives 1 BTC and B's address to the mixer
  - the mixer gives 1 BTC to B
  - later, B gives 1 BTC and C's address to the mixer
  - the mixer gives 1 BTC to C
- However, it's better to assume that the mixer may behave dishonestly

# Goal of this work

- **TumbleBit** [Heilman et al., 2017] is a protocol that:
  - runs on Bitcoin's blockchain
  - allows A to pay 1 BTC to B
  - even the mixer (called tumbler) cannot link transactions between payers and payees
  - does not allow the mixer to steal coins while processing transactions
  - is scalable, since it mainly works off-chain, and performs only two on-chain operations (setup of the payment channel, and cash out)
- However, if B wants to pass his BTC to C, he has to interact once again with the mixer
  - we modify TumbleBit's protocol so that B can directly (off-chain) pass his BTC to C

# Some tools

- Offer transaction $T_o$:
  - a payer 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 $C$
- Fulfill transaction $T_f$:
  - posted by B on the blockchain to redeem the bitcoins offered by A
  - contains B's public key, and points to the output of $T_o$
  - contains a predicate which satisfies condition $C$ in $T_o$
- *Time-locked* transactions: A can specify in $T_o$ a time window *tw*, before which the fulfill transaction $T_f$ has to be registered on the blockchain

# Some tools

- TumbleBit uses the Pay-To-Script-Hash (P2SH) format:
  - A stores in $T_o$ the hash of a redeem script, which contains the condition $C$ to be satisfied
  - 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

# Some tools

- Condition $C$ can also be of different types:
  - *hashing condition*: given a cryptographic hash function $H$, and a fingerprint $y$, provide a preimage $x$ of $y$. TumbleBit uses SHA-256
  - *signing condition*: given a digital signature scheme $\mathbf{S}$, the signature $S$ of $T_f$ must verify under a public key $PK$, that is, $\text{Ver}_{PK}(T_f) = S$. TumbleBit uses ECDSA with Secp256k1, to make the protocol compatible with the Bitcoin scripting language
- Hashing and signing conditions can also be composed by requiring that two preimages/signatures are provided:
  - *double-hashing condition*: given the hash function $H$ and a pair of values $(w, z)$, provide a preimage $x$ of $w$ and a preimage $y$ of $z$
  - *2-of-2 escrow condition*: two signatures $S_1$ and $S_2$ are required, one under a public key $PK_1$ and the other on the public key $PK_2$

# TumbleBit's overview

- TumbleBit has three phases:
  - *Escrow phase*: the payment channels between A and T and between T and B are set up. Moreover, both A and T put 1 BTC in escrow
  - *Payment phase*: an interaction between T and B is used to generate a puzzle, which is then solved during an interaction between A and T. These interactions are performed off-chain
  - *Cash-out phase*: T redeems 1 BTC from A, and B redeems 1 BTC from T
- Using time-locked transactions, the beginning and the end of each phase can be marked by a fixed amount of new blocks appended on the Blockchain
  - A, B, and T know when each phase begins and ends

# TumbleBit's phases

- The *escrow phase* is composed of three steps:
  1. B asks T to open a payment channel: T posts on the blockchain a 2-of-2 escrow transaction $T_{T,B}$ which escrows 1 BTC, with the following condition $C_{T,B}$:

> B can claim 1 BTC by providing the two signatures $S_T$ and $S_B$, which verify under the public keys $PK_T$ and $PK_B$ of T and B, respectively. This transaction is time-locked to a time window $tw_2$, after which T can claim back its BTC

# TumbleBit's phases

2. A opens a payment channel to T and escrows 1 BTC by registering another 2-of-2 escrow transaction $T_{A,T}$ on the blockchain, with the following condition $C_{A,T}$:

> 1 BTC can be claimed by T by presenting two signatures $S_A$ and $S_T$ which verify respectively under the public keys $PK_A$ and $PK_T$.
> Also this transaction is time-locked to a time window $tw_1 < tw_2$, after which A can claim back her BTC

# TumbleBit's phases

3. T and B engage in a puzzle-promise protocol, after which B receives from T a pair of values $(c, z)$, where $c$ is the encryption of the Tumbler's signature $S_T$:

$$c = Enc_{\varepsilon}(S_T)$$

where *Enc* is a symmetric encryption algorithm, and $\varepsilon$ is an encryption key randomly chosen by T.

$z$ is the RSA encryption of the symmetric key $\varepsilon$ under T's public key $PK_T = (e, N)$:

$$z = \text{RSA}(\varepsilon, e, N) = \varepsilon^{e} \bmod N$$

This puzzle-promise protocol is used to ensure that T cannot act dishonestly by sending B a value $c$ that does not correspond to the encryption of its signature $S_T$

# TumbleBit's phases

- The *payment phase* is also composed of three steps:
  1. B samples a random element $\alpha \in \mathbb{Z}_N^*$, and uses it to re-randomize the puzzle $z$ received from $T$, by computing $z' = \alpha \cdot z$.
     This step is also called *blinding*. Next, B sends $z'$ to A
  2. After receiving $z'$ from B, A and T engage in a *puzzle-solving protocol*.
     The goal of A is to obtain from T the solution of the puzzle $z'$ to send back to B, whereas T wants A to sign the fulfill transaction associated to $T_{A,T}$, in order to redeem her Bitcoin escrowed in the first phase

# TumbleBit's phases

The puzzle-solving protocol ensures that:

- the value obtained by A from T is indeed the solution $\varepsilon'$ of the blinded puzzle sent by B, that is:

$$\varepsilon' = z'^{d} \bmod N$$

where $d$ is the private key of $T$

- T receives A's signature $S_A$ by providing the correct solution of the puzzle $z'$

3. A sends the solution $\varepsilon'$ of the blinded puzzle to B, who can unblind it by computing

$$\varepsilon = \varepsilon' / \alpha$$

To be sure that A sent the correct solution, B checks the obtained value $\varepsilon$ by verifying that $\varepsilon^e = z \bmod N$

# TumbleBit's phases

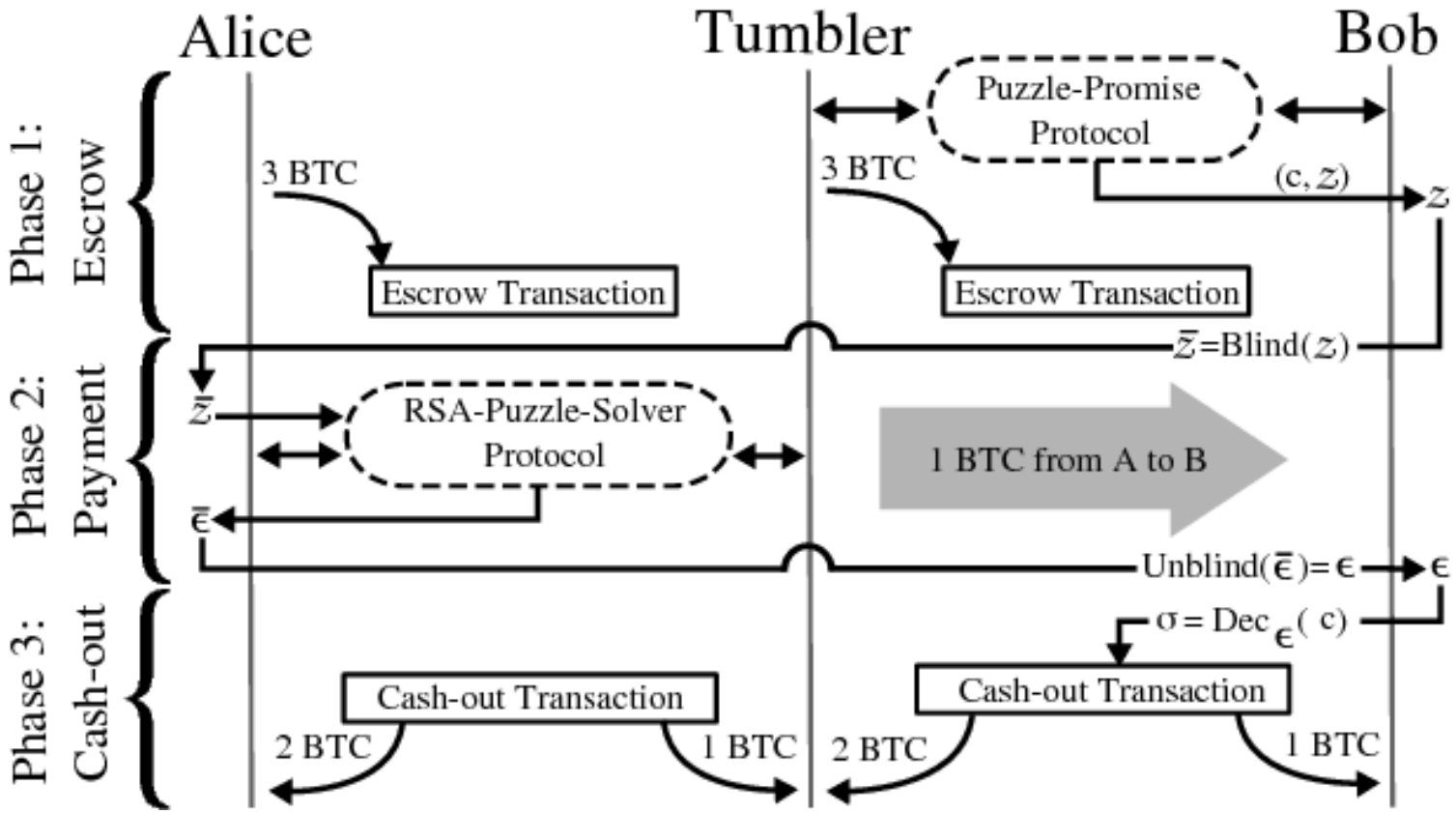
- The *cash-out phase* is composed of two steps:
  1. B decrypts the value $c$ given to him by T, thus obtaining its signature $S_T$:

     $$S_T = Enc_{\varepsilon}^{-1}(c)$$

     B is thus able to satisfy the escrow condition of the 2-of-2 escrow transaction $T_{T,B}$.

     Hence, B posts on the blockchain a fulfill transaction $T_{f(T,B)}$, and retrieves the Bitcoin escrowed by T
  2. On the other hand, T received A's signature $S_A$. Hence, T posts a fulfill transaction $T_{f(A,T)}$ on the blockchain, and redeems A's Bitcoin

# Blockchains: research



# Our modification

- After receiving $\varepsilon'$ from A, B would like to anonymously forward to C the BTC escrowed by T, but without interacting further with it
- This cannot be done with the original protocol, since in step (1) of the escrow phase, T posts a 2-of-2 escrow transaction on the blockchain, which requires both T's and B's signatures to be fulfilled
  - B and C should engage with T in another round of the TumbleBit protocol
- We modify the initial part of the TumbleBit protocol by using a P2SH transaction, whose escrow condition consists of providing a pair of SHA-2 preimages, respectively chosen by the Tumbler and Bob
  - the P2SH transaction does not bind the receiver to a specific address

# More in detail

- The first step of the escrow phase is modified as follows:
  - when B asks T to open a payment channel, B samples a random string $r \in \{0,1\}^*$ with uniform probability, computes $R$ = SHA-256($r$), and sends $R$ to T
  - on the other side, T samples a value $s \in \{0,1\}^*$, computes $S$ = SHA-256($s$) and posts on the blockchain an offer transaction $T_{H(R,S)}$ which escrows 1 BTC, where the condition is $C_{H(R,S)}$:

> 1 BTC can be claimed by any recipient who provides the SHA-256 preimages $x$, $y$ of $R$ and $S$, respectively

# More in detail

- After this step, the protocol proceeds as in the original version, by changing the relevant steps where the elements of the offer transaction $T_{H(R,S)}$ are involved:
  - 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 T, while $z$ is an RSA encryption of its preimage $s$:

$$c = S = \text{SHA-256}(s)$$

$$z = \text{RSA}(s, e, N) = s^e \bmod N$$

where $(e, N)$ is again T's public key

# More in detail

- The payment phase unfolds exactly as in the original version of the protocol:
  - B blinds the RSA puzzle $z$, and sends the blinded version $z'$ to A
  - the puzzle-solving protocol between A and T proceeds in the same way
  - once B gets the solution from A, he unblinds it by dividing it by $\alpha$. So B has obtained the preimage $s$ of $S$, and he can fulfill the offer transaction posted by T on the Blockchain, since he possesses both his preimage $r$ and T's preimage $s$
- Notice that any user who is able to give the SHA-256 preimages of $R$ and $S$ can claim 1 BTC from T, not necessarily B
  - to forward 1 BTC to C, B can simply send to C the pair ($r$, $s$)

# Security properties

- The anonymity properties of the original TumbleBit protocol are preserved: T is not able to link the sequence of payments between A, B and C
  - the payment A $\rightarrow$ B is made as in the original TumbleBit protocol
  - the payment B $\rightarrow$ C is made off-chain
- However, this does not work in permissionless blockchains, because the tumbler can steal Bitcoins by *mauling* the modified P2SH transaction:
  - it can take the pair ($r$, $s$) from C's transaction, and include it in a new transaction where he is the recipient

# Discussion and open problems

- Is it useful in permissioned blockchain?
  - the validators have incentives to not steal the tokens in this way, otherwise they are banned from the consortium (coin theft can be detected but not avoided)
  - interbank payments, where central banks are the tumblers?
- Can we make something better, using Turing-complete languages?
  - zcash-like anonymity (via ZK-Snarks)?
  - an off-chain unlinkable payment channel called Bolt has already been proposed [Green & Miers, 2017]

# Thanks for your attention!


Alberto Leporati

alberto.leporati@unimib.it