Towards Efficient Cross-Blockchain Token Transfers

Philipp Frauenthaler*, Marten Sigwart*, Michael Borkowski*, Taneli Hukkinen‡, Stefan Schulte*

* Distributed Systems Group
TU Wien, Vienna, Austria
{p.frauenthaler, m.sigwart, m.borkowski,
s.schulte}@infosys.tuwien.ac.at

‡ Pantos GmbH
Vienna, Austria
contact@pantos.io

***Abstract*—Interoperability between blockchains remains an open problem, with current interoperability approaches providing very limited means of cross-blockchain interaction, mostly in the form of atomic swaps. More general means of blockchain interoperability such as cross-blockchain data exchange, including cross-blockchain token transfer would contribute to dissolving today's fragmentation of the research and development field of blockchains. To address this issue, within the TAST research project, a cross-blockchain token was developed. However, the developed solution suffers from high synchronization cost.**

In this paper, we discuss requirements for more efficient cross-blockchain token transfers, describe open research challenges, and give an outlook on two approaches aiming to overcome these challenges.

I. Introduction

Since the presentation of Bitcoin [9], the first implementation of a blockchain protocol in widespread use, the utility and feasibility of decentralized ledgers has been demonstrated for various use cases and fields [6]. As discussed in our previous work [1, 2, 3, 4], research activities related to blockchains cover, among others, the addition of new layers to Bitcoin itself [12], improvements to the Bitcoin codebase [8], and the development of entirely new blockchains [13]. The diversity and richness of this research field comes with an increasing number of technologies and implementations [7], causing structural problems within the blockchain community. The vast amount of blockchains and other projects in existence causes severe fragmentation of the research and development field. Interoperability is mostly not foreseen, with blockchains instead competing for users and developers [2].

Therefore, in the Token Atomic Swap Technology (TAST) research project[1], we aim to create a platform for cross-blockchain interoperability. The overarching goal is to connect the fragmented fields of research and development by investigating possible means of interconnecting various blockchain-related projects. For instance, this can be done by developing currencies and tokens usable on more than just one blockchain (cross-blockchain tokens), investigating the transfer of data across blockchains (cross-blockchain data storage), or by enabling more complex interactions such as calling smart contract functions from different blockchains (cross-blockchain smart contract invocations).

White Paper, TU Wien; May 2019, version 1.0.

[1]http://www.infosys.tuwien.ac.at/tast/

As a first step towards more general blockchain interoperability, we aim to create a cross-blockchain token. Earlier in the TAST project, we developed a protocol enabling a first kind of such tokens [4]. The protocol enables a token to exist on multiple blockchains at the same time. However, the designed protocol suffers from a couple of limitations, e.g., very high synchronization cost. Therefore, in the work at hand, we revisit the original vision of a cross-blockchain token and introduce two approaches currently being worked on within the TAST project that aim to overcome the current limitations.

To this end, Section II defines concrete requirements for realizing our original vision of a cross-blockchain token, and Section III discusses the limitations of the current protocol. In Sections IV and V, we introduce two approaches for overcoming these limitations. Finally, Section VI concludes the paper.

II. Requirements for Cross-Blockchain Token Transfers

The TAST project aims to enable a cross-blockchain token. Ideally, such a token enables users to freely choose on which blockchain they want to hold their assets, i.e., users are not tied to particular blockchains and are able to hold different denominations of the token on multiple blockchains at the same time. This would have the further advantage that the distribution of assets across the participating blockchains could give an indication about the significance of a particular blockchain.

If a new blockchain technology emerges and offers novel features, users should be able to transfer their assets to this new blockchain taking advantage of the novel capabilities. Finally, different blockchains employ different consensus mechanisms, block sizes, confirmation times, hashing algorithms, network models, and scripting capabilities. A cross-blockchain token should take into account this diversity by enabling cross-blockchain token transfers across a wide variety of different blockchains. That is, in contrast to developing new blockchains offering the capabilities for a cross-blockchain token, we aim to enable a cross-blockchain token for existing blockchains. Ideally, blockchains which already have many users are able to participate in the envisioned cross-blockchain token.

Let us assume there exists a token that can be transferred between multiple blockchains. That is, holders of the token are free to choose on which blockchain they keep their assets. Further, participants can transfer amounts of the token from one "source" blockchain to an arbitrary "destination" blockchain. Such transfers should only be successful, i.e., the specified amount of tokens is created on the destination chain, if the same amount of tokens has been burned (i.e., destroyed) on the source chain. If this was not the case, tokens could effectively be created out of nothing since there is no assurance that tokens that are being created on the destination chain have actually been burned on the source chain. The ability to create tokens out of nothing would subsequently deflate the value of the token. Therefore, before tokens are created on the destination chain, the destination chain needs some kind of proof that the same amount of tokens has been burned on the source chain.

Now assume it is possible to create such proofs guaranteeing that a certain amount of some token $\mathcal{T}$ has been burned on some source blockchain $\mathcal{C}_s$ and that these proofs can be used to create the same amount of $\mathcal{T}$ on some destination blockchain $\mathcal{C}_d$. Two further requirements emerge. First, faking a proof needs to be prevented at all cost. Participants should not be able to counterfeit a proof certifying that some amount of $\mathcal{T}$ has been destroyed on $\mathcal{C}_s$ without it actually having occurred. Second, if a proof is correct (i.e., it has not been faked), it should only be usable once to create the same amount of $\mathcal{T}$ on a different blockchain, i.e., on chain $\mathcal{C}_d$. Hence, a single proof cannot be used multiple times to re-create tokens. Essentially, disregard of both these requirements would enable participants to illegally create tokens out of nothing—again deflating the value of the tokens.

To sum up, we define the general requirements for a cross-blockchain token transfer as follows.

(1) It should not be possible to create tokens on the destination blockchain $\mathcal{C}_d$, without first burning the same amount of tokens on the source blockchain $\mathcal{C}_s$.
(2) When transferring some amount of tokens from $\mathcal{C}_s$ to $\mathcal{C}_d$, the amount should only be created on $\mathcal{C}_d$, if it can be proven that the same amount has been burned on $\mathcal{C}_s$.
(3) It should not be possible to fake the burning of tokens.
(4) Every token that was burned on one blockchain can only be (re-)created once on another blockchain.
(5) It should not be possible to destroy tokens on one chain without recreating them on another.

III. Progress within TAST

The TAST research project has resulted in numerous contributions throughout its progress to date. One of these contributions is a research prototype[2] demonstrating the use of the concepts discussed in previous publications using Solidity. This prototype already provides the means for realizing cross-blockchain token transfers. In the following, we briefly summarize the main concepts implemented by this prototype. Furthermore, we discuss limitations of the current approach.

As outlined in Section II, the creation of a certain amount of assets on some blockchain $\mathcal{C}_d$ requires the same amount to be destroyed on another blockchain $\mathcal{C}_s$, ensuring that assets cannot be created out of nothing. Blockchains cannot natively verify events taken place on other blockchains, e.g., a smart contract running on blockchain $\mathcal{C}_d$ cannot natively access data located on blockchain $\mathcal{C}_s$ [1]. Hence, to enable a blockchain $\mathcal{C}_d$ to check whether a user holds enough assets on some other blockchain $\mathcal{C}_s$, updates of balances need to be synchronized across all blockchains, i.e., each wallet balance is stored on each participating blockchain. To synchronize balances of all participating blockchains, the research prototype leverages so-called *deterministic witnesses* [2]. The idea is to give observing parties (witnesses) sufficient incentive to propagate information of a transfer (e.g., which user wallet is supposed to receive the transfer amount) to the other blockchains. Through the propagation of transfer information to each participating blockchain, balances are synchronized on all blockchains. Thus, each wallet has the same balance on each blockchain. A transfer of some amount of assets from a wallet $\mathcal{W}_A$ to a wallet $\mathcal{W}_B$ eventually leads to the reduction of $\mathcal{W}_A$ on all blockchains and to the increase of $\mathcal{W}_B$ on all blockchains. This synchronization happens through the aforementioned deterministic witnesses and ensures eventual consistency between the balances stored on different blockchains.

In order to incentivize witnesses to propagate information across blockchains, witnesses receive a reward for their services. Since the reward consists of some amount of the cross-blockchain token, the reward itself leads to a balance change, which again has to be propagated to all participating blockchains. Hence, despite the involvement of many witnesses, only one witness will receive a reward on all blockchains. To determine the witness receiving the reward, a contest takes place which leverages the determinism prevalent in blockchain technologies. In this contest, not the timing of a witness decides on the reward decision, but a hash value generated from the witness address together with the Proof of Intent (PoI), which is provided in the transaction that aims to (re-)create assets. This means that the contest winner is determined from the PoI data together with a pool of potential witnesses. Thus, the outcome (the answer to the question of who receives the witness reward) is deterministic and therefore identical across all blockchains for each individual transfer. However, for different transfers, different witnesses may be chosen to receive the reward.

Despite enabling cross-blockchain token transfers, this approach poses significant challenges. First, the synchronization of any balance change across all blockchains leads to excessive synchronization cost. The more blockchains are supported by the protocol, i.e., the more blockchains participate in cross-blockchain token transfers, the higher the synchronization cost becomes. Second, the devised approach provides no means of adding a new blockchain later on. Since every blockchain stores the current balance of each wallet, these balances must also be synchronized with a new blockchain. This leads inevitably to the open question how all existing balances can be transferred to a new blockchain without relying on a trusted third party. Third, in order to verify digital signatures, all blockchains must support the same implementations of

[2]https://github.com/pantos-io/dextt-prototype

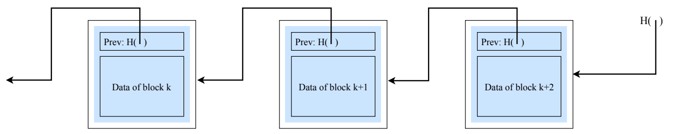

Figure 1: A blockchain is a linked list that utilizes hash pointers instead of regular pointers [10].

the required cryptographic primitives. Fourth, the proposed approach does not allow to determine the significance of individual blockchains (e.g., how much assets are stored on each blockchain), since each blockchain stores the same wallet balances. Finally, it is not possible for users to hold different denominations of an asset on different blockchains at the same time.

In the next step, we aim to restrict the required interaction of a cross-blockchain token transfer to the two blockchains directly involved in the transfer avoiding the need for synchronization across all participating blockchain, i.e., balances are only changed on the two blockchains directly involved in the transfer. Assuming a certain amount of assets are transferred from blockchain $\mathcal{C}_s$ to blockchain $\mathcal{C}_d$, then the balance of the sender should be decreased on $\mathcal{C}_s$ and the balance of the receiver should be increased on $\mathcal{C}_d$, without changing balances on any other participating blockchain. In order to restrict the required synchronization to the two blockchains directly involved in the transfer, we aim to make use of the concepts developed within TAST as well as new techniques. In Sections IV and V, we introduce two possible approaches suitable for reducing the synchronization cost.

IV. Approach 1: SPV-Based Cross-Blockchain Token Transfer

The basic idea of approach 1 is to enable users to construct a cryptographic proof certifying that a particular amount of assets has been burned on some blockchain $\mathcal{C}_s$. By submitting this proof to another blockchain $\mathcal{C}_d$, users can prove to $\mathcal{C}_d$ that the specified amount of assets has been destroyed on $\mathcal{C}_s$. After a successful verification of this proof, the claimed tokens are (re-)created on $\mathcal{C}_d$. In the following, we elaborate on this approach in more detail. Furthermore, we outline some challenges that have to be solved in order to make the proposed approach feasible in practice.

A. Preliminaries

Basically, a blockchain is a linked list that is composed of a sequence of blocks [10]. As shown in Fig. 1, each block contains a pointer to the preceding block. Instead of regular pointers, blockchains utilize hash pointers. Hash pointers not only hold a reference to the location of some data, but are composed of the hash value of the referenced data. In case the referenced data is altered, the hash pointer does not match the data anymore since an alteration also changes the hash of the data. Thus, hash pointers provide integrity guarantees. Therefore, by connecting blocks with hash pointers, a blockchain represents a tamper-evident data structure [10].

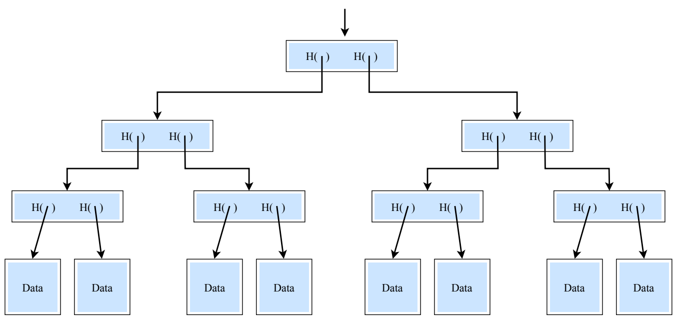

Figure 2: A Merkle tree is a tree that is built with hash pointers instead of regular pointers [10].

In blockchain networks such as Bitcoin and Ethereum, blocks do not directly store transactions, they only contain some kind of metadata such as block number, difficulty and the hash pointer to the previous block [10, 5]. Thus, in the following we refer to these lightweight blocks as block headers. We use the term block to denote both the block header containing the metadata and the set of transactions belonging to the block, i.e., a block is made up of a block header and a set of transactions.

The transactions of a block are stored in another data structure, the so-called Merkle Tree [10, 5, 11]. A Merkle tree is a tree structure in which each node uses hash pointers to reference its child nodes. The leaf nodes of the tree contain the data blocks (e.g., in case of Bitcoin the transactions). An example of a Merkle tree is depicted in Fig. 2. Analogous to the blockchain, in case any node is changed the hash pointer stored in the node's parent is not going to match up. As long as the hash pointer referencing the root node (the so-called Merkle root hash) cannot be altered, any alteration is evident. Another useful feature of a Merkle tree is the ability to construct a concise proof of memberships showing that a particular data block is included in the tree. To prove to someone the inclusion of a certain data block, it is sufficient to show the nodes of the path from that data block up to the root node of the Merkle tree. Everyone that is in possession of the hash pointer to the root node of the Merkle tree can verify a provided proof of membership by simply recalculating the hashes of all nodes along the path from the data block up to the root node. If the calculated root hash matches the root hash that is in possession of the verifier, the membership has been successfully proven [10].

Blockchains like Bitcoin or Ethereum combine the two data structures (blockchain and Merkle tree) by storing the hash pointer referencing the root node of a Merkle tree in the block header, together with other metadata such as block number and difficulty. The concrete structure of a block header (e.g., which fields are stored in a block header) depends on the system that is used. For instance, a block header of the Bitcoin blockchain contains only one Merkle root hash, whereas in Ethereum, a block header contains the root hashes of multiple Merkle trees [10, 5]. In Fig. 3, a simplified structure of the Bitcoin blockchain illustrating the combination of the chain of blocks

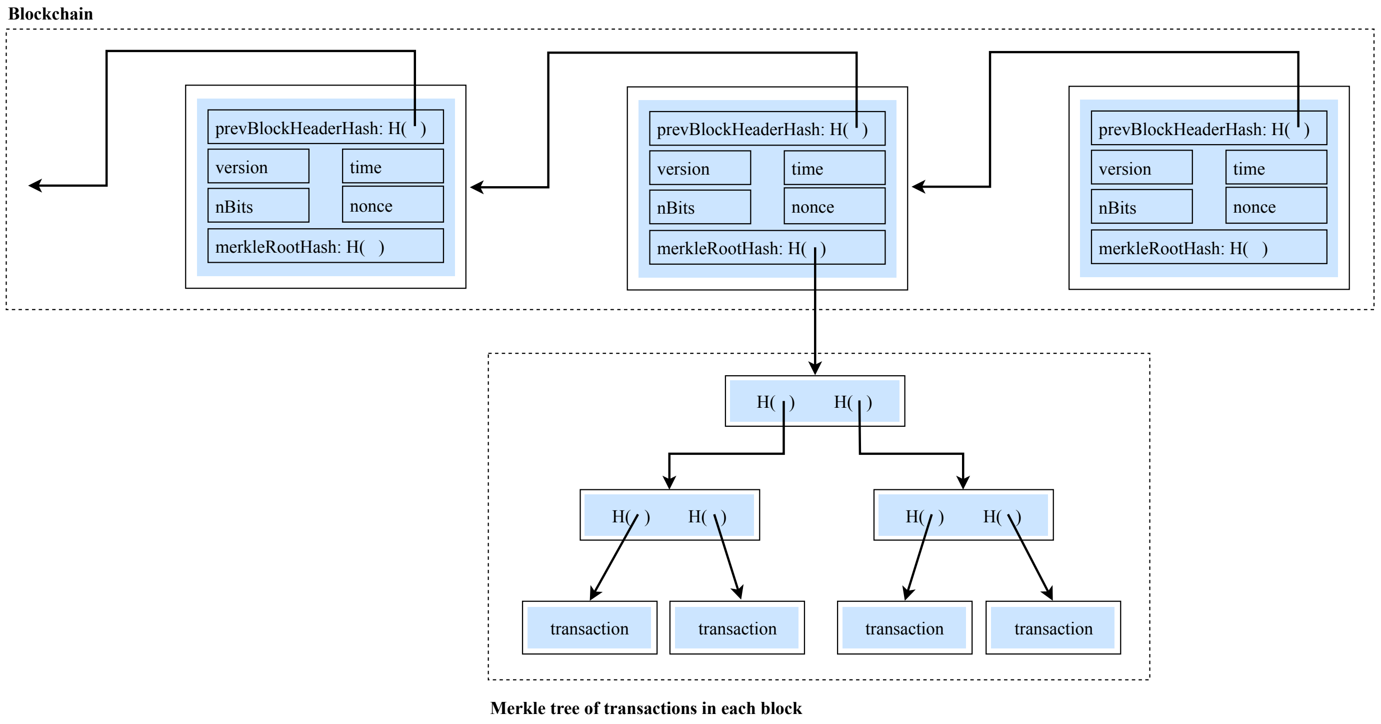

Figure 3: A simplified structure of the Bitcoin blockchain [10, 11].

and Merkle trees is shown.

A major advantage of this combination is the ability to verify the inclusion of a transaction in a blockchain without the need to download the full blockchain. In order to verify that a particular transaction is included in a certain block, it is sufficient to have access to the block headers, e.g., in case of Bitcoin and Ethereum to a copy of the block headers of the longest Proof of Work (PoW) chain. Storing only these headers requires less space than storing the complete blocks. If someone wants to prove that a particular transaction is included in some block, they have to provide the Merkle proof of membership containing the transaction to a verifier. The verifier simply checks the proof by recalculating the hashes of the nodes along the path constituting the proof. If the calculated root hash matches with the Merkle root hash stored in the corresponding block header, the verifier knowns that the provided transaction is part of the block without having access to the transactions. However, Merkle proofs of membership do not prove that a certain block is included in the blockchain. Thus, in addition to the verification of the Merkle proof of membership, it is necessary to check whether the block is included in the longest proof-of-work chain. This kind of verification is called *Simplified Payment Verification (SPV)* and was first introduced by Satoshi Nakamoto [5, 9, 10].

B. The Protocol

In approach 1, we aim to leverage SPVs to enable users to prove to some blockchain $\mathcal{C}_d$ that a certain amount of assets has been destroyed on another chain $\mathcal{C}_s$, i.e., to show that a particular DESTROY transaction is included in some block of blockchain $\mathcal{C}_s$. Hence, when attempting to create tokens on $\mathcal{C}_d$ via a CREATE transaction, a user has to provide a Merkle proof of membership to $\mathcal{C}_d$ and $\mathcal{C}_d$ must have access to the block headers of $\mathcal{C}_s$ (i.e., $\mathcal{C}_d$ has to be continuously updated with new block headers of $\mathcal{C}_s$). An important aspect of this approach is that only the balances of the two blockchains directly involved in the transfer are changed, thus eliminating the excessive synchronization cost entailed by the current prototype. However, utilizing SPVs poses some challenges that have to be solved in order to make the approach viable in practice. Next, we elaborate on these challenges.

C. Challenges

To realize secure cross-blockchain token transfers using SPV proofs, several challenges have to be solved. Below we describe these challenges and briefly explain possible approaches to solve them.

1) Efficiency of Proofs: While SPV proofs can cryptographically verify whether or not a certain transaction has been included within a certain blockchain, they require the validation of all block headers from the genesis block up to the block where the transaction is included. For each block header, the prior block as well as the PoW needs to verified. Hence, when submitting a transaction to the destination blockchain containing an SPV about a transaction included in the source blockchain, the destination blockchain either needs to know about all block headers of the source blockchain or they have to be provided by the user submitting the SPV. However, storage as well as computation are expensive operations in blockchain systems, since state and state transitions are replicated across all nodes of a blockchain [11]. Therefore storing and verifying all block headers in order to verify an SPV proof could easily become too expensive. To tackle this issue, a solution is needed that can reliably verify SPV proofs while keeping storage and computation requirements to a minimum. In the TAST project, we are currently developing solutions where block headers only need to be verified once and where not all information of block headers need to be stored. As long as block headers of the source blockchain are regularly propagated to the destination blockchain, the cost for verifying SPV proofs remains reasonable.

2) Propagation of Block Headers: As stated above, in the TAST project, we are currently developing solutions for keeping cost of an SPV proof verification low, under the assumption that the recent block headers of the source blockchain are regularly propagated to the destination blockchain. However, therein lies another challenge. How do we ensure that block headers are regularly propagated to the destination blockchain? A potential solution to this is to incentivize participants to regularly submit new block headers of the source chain to the destination chain, that is, participants can gain a reward if they propagate new block headers.

3) Blockchain Forks: As described above, SPV proofs prove that a certain transaction is included in some block which is included in a valid branch of a blockchain. However, multiple valid branches of the same blockchain may exist in parallel (i.e., blockchain forks). Hence, it is not enough to verify an SPV proof in order to be certain that a cross-blockchain token transfer is valid, it also needs to be ensured that the corresponding DESTROY transaction was included in the branch of the source blockchain that represents the longest PoW chain (i.e., the branch of the blockchain that has been accepted by the majority of the chain validators as the true state of the blockchain). For this reason, PoW blockchains such as Bitcoin and Ethereum use the concept of confirmed blocks. Once a block is followed by a certain number of valid succeeding blocks, the block can be seen as confirmed and the contained transactions as finalized. Hence, in order to account for the possibility of blockchain forks, within approach 1, we plan to incorporate a concept similar to that of confirmed blocks to ensure the validity of cross-blockchain token transfers.

In summary, approach 1 aims to leverage SPV proofs to verify the inclusion of a DESTROY transaction within the source blockchain in order to accept a CREATE transaction on the destination blockchain. To make SPVs work in practice, solutions are being developed that minimize the storage and computational requirements for verifying such proofs.

V. Approach 2: Incentivized Majority Voting

Instead of relying on cryptographic proofs as in approach 1, approach 2 relies on observing parties (witnesses) to determine whether or not tokens can be created on the destination blockchain. We define witnesses as participants of the blockchain ecosystem which do not partake directly in the token transfer, but rather observe transfers and verify that the correct amount of tokens has actually been destroyed on the source blockchain before new tokens are created on the destination blockchain. Imagine, Alice wants to transfer 4 tokens from her account on blockchain $\mathcal{C}_s$ to her account on blockchain $\mathcal{C}_d$. She creates a DESTROY transaction on blockchain $\mathcal{C}_s$, and a CREATE transaction on blockchain $\mathcal{C}_d$. However, before the CREATE transaction can be finalized, it has to be verified that the DESTROY transaction has actually taken place. For that, witness Walter can send a WITNESS transaction to blockchain $\mathcal{C}_d$ saying he has witnessed the DESTROY transaction on blockchain $\mathcal{C}_s$. With the confirmation from Walter, the 4 tokens can be created on blockchain $\mathcal{C}_d$.

Several problems come to mind: First, what motivates Walter to send the WITNESS transaction? After all, it will cost him transaction fees. Second, Alice and Walter might be accomplices. Walter could say he witnessed the DESTROY transaction without it actually taking place. This would effectively create tokens for Alice out of nothing. We see, witnesses should be incentivized to not only participate in the verification process but to also post the correct validation results. For that purpose, we suggest a majority vote contest incentivizing witnesses to participate and to post correct validation results.

A. The Protocol

In the following, we describe the steps involved in the proposed protocol. For this, we assume that two blockchains $\mathcal{C}_s$ and $\mathcal{C}_d$ are involved. Furthermore, we assume Alice wants to move 4 tokens from blockchain $\mathcal{C}_s$ to blockchain $\mathcal{C}_d$.

1) In a first step, Alice creates and signs a CREATE transaction containing data such as source and destination blockchains and the amount of tokens to be transferred.
2) Alice submits this CREATE transaction to blockchain $\mathcal{C}_d$.
3) $\mathcal{C}_d$ starts a voting contest for witnesses to vote on the validity of the transfer. A witness has two voting options.
   a) The first option is to vote valid, indicating the transfer is valid, i.e., the correct amount of tokens has been destroyed on chain $\mathcal{C}_s$.
   b) The second option is to vote invalid indicating at least one condition is violated, e.g., Alice has not enough tokens on blockchain $\mathcal{C}_s$ or Alice has not submitted a DESTROY transaction to blockchain $\mathcal{C}_s$.

   During the contest, witnesses cannot see the votes of others, i.e., they do not know for what others have voted. Further, due to network latencies, participants cannot reliably know how many votes have already been submitted. After the contest expires, the votes are uncovered, revealing the option the majority has voted for. In order to participate, witnesses are required to provide a stake. Witnesses that are among the minority lose their stake, whereas witnesses that are among the majority keep their stake. If the majority votes valid, the reward consists of a transfer fee (e.g., a fraction of the transferred value) and the collected stakes of the witnesses that are among the minority. If the majority votes invalid, the reward consists of a penalty for the sender and the collected stakes of the witnesses that are among the minority.

Table 1: Voter preference depending on the size of the transfer fee with respect to the penalty and the actual validity of the transfer

	Transfer is valid	Transfer is invalid
Transfer fee > penalty	*Vote valid*	*Vote valid*
Transfer fee < penalty	*Vote invalid*	*Vote invalid*

4) Based on the result of the contest, the transfer is either accepted (i.e., the majority votes valid) or rejected (i.e., the majority votes invalid). In case the transfer is valid, the tokens are created and added to Alice' balance on blockchain $\mathcal{C}_d$. In case the transfer is invalid, Alice gets penalized (e.g., by deducting a certain amount of Alice' balance on blockchain $\mathcal{C}_d$).

Witnesses are only eligible for a reward if they are part of the majority. Additionally, if they are part of the minority, they will lose their stake. Hence, a witness is always motivated to be part of the majority. Therefore, under the assumption that the majority of the witnesses participating in the vote is honest, an individual witness is incentivized to vote honestly, too. On the other hand, under the assumption that the majority of the witnesses participating in the vote is dishonest, an individual witness is motivated to vote dishonestly, too.

Under the assumption that witnesses are trying to maximize their profits, whether to vote honestly or not, depends on the financial reward of the vote:

- If the majority votes *valid*, and the transfer is *valid*, the reward consists of a transfer fee and the stake of the minority.
- If the majority votes *valid*, and the transfer is *invalid*, the reward consists of the transfer fee and the stake of the minority (however, tokens get created out of thin air).
- Similarly, if the majority votes *invalid*, the reward consists of the penalty and the stake of the minority, regardless of whether the transfer is actually valid or not.

Since the potential reward for the majority coming from the minority's stake is uncertain (it could be zero in case everyone votes the same), and assuming witnesses aim to maximize rewards, voting preference becomes a simple consideration as seen in Table 1:

- If transfer fee is bigger than the penalty and the transfer is valid, then vote *valid*.
- If transfer fee is bigger than the penalty and the transfer is invalid, then vote *valid*.
- If transfer fee is smaller than the penalty and the transfer is valid, then vote *invalid*.
- If transfer fee is smaller than the penalty and the transfer is invalid, then vote *invalid*.

Hence, to prevent voters from having an optimal voting strategy, the reward should be equal regardless of whether witnesses vote valid or invalid (i.e., the transfer fee should be equal to the penalty). If the transfer fee is the same as the penalty, the majority has no financial preference of voting valid or invalid, since both yield the same financial rewards. Therefore, the question is: Under what circumstances can we assume that the majority will behave honestly? Our hypothesis is that, as long as the pay-off for voting honestly or dishonestly is equal, voting honestly is the preferred option for the majority since dishonest activity could potentially lead to tokens being created out of nothing which could eventually lead to a loss of value of the token. Next, we explore some of the specific challenges of approach 2.

B. Challenges

For approach 2 to realize cross-blockchain token transfers, several challenges need to be tackled. Among them are the prevention of double spending and 51% attacks as well as a potential nothing-at-stake problem.

1) Double Spending: Double spending can occur, for instance, if Alice submits a valid DESTROY transaction on blockchain $\mathcal{C}_s$, and then submits CREATE transactions on two blockchains, blockchain $\mathcal{C}_d$ and blockchain $\mathcal{C}_e$, at the same time. Witnesses on both chains would see the CREATE transactions and examine blockchain $\mathcal{C}_s$ for a valid DESTROY transaction. Since a valid DESTROY transaction exists on chain $\mathcal{C}_s$, witnesses are likely to vote *valid* on both chains $\mathcal{C}_d$ and $\mathcal{C}_e$, effectively double spending the tokens that were destroyed on chain $\mathcal{C}_s$. As such it needs to be ensured that one DESTROY transaction can only be used once to recreate the destroyed tokens on one other blockchain (see Section II, Requirement 4). Possible solutions are the concepts of "veto" transactions and validity periods as used in the prototypes of the TAST project.

2) 51% Attacks: A further challenge lies in possible 51% attacks. In the described approach, transfers get accepted depending on a majority vote. As such, the majority has complete power over the approval of transfers. Naturally, the system fails if the majority decides to vote dishonestly. Hence, it needs to be prevented that a majority of participants can organize and coordinate their votes to vote dishonestly. To solve this problem, incentives have to be carefully aligned in such a way that behaving honestly is always more rewarding than behaving dishonestly. The risk for participating in a 51% attack has to be high. A further challenge is to get as many users as possible to participate in the voting contest in the first place. If only a few users participate, successful coordination among participants is more likely, i.e., the probability of a 51% attack succeeding increases.

3) Nothing-at-Stake Problem: If a transfer is deemed invalid by the majority vote, the payoff for the majority consists of the stake of the minority as well as a penalty from the user who initiated the transfer. The protocol intends to deduct the penalty from the funds of the user initiating the transfer on the destination blockchain since this is where the vote takes place. However, what happens if that user does not have any existing funds on the destination blockchain? The user would have nothing to lose, since no funds could be deducted as a penalty. To combat such scenarios, one possible solution would be the employment of a kind of bail system. Users intending to transfer funds to a blockchain where they currently hold no assets would have to find someone with sufficient funds on that blockchain that provides the penalty for the user in case the transfer is deemed invalid. Of course, it is unlikely that users provide bail for users if there is no kind of reward for them as well. The reward could consist of a percentage of the transferred amount of tokens if the transfer is successful. In case the transfer is declined, it is unlikely that the user who initiated the transfer will find users which will bail them out again in the future. A further challenge in that regard would be adding completely new blockchains to the ecosystem. A bail system as described above does not work, since no user would own any funds on the new blockchain.

Summarizing, approach 2 declines or accepts token transfers between blockchains based on the outcome of a validity vote by witnesses. The main challenge consists of building an incentive structure which (a) motivates as many witnesses as possible to participate in the vote, and that (b) distributes rewards and penalties in such a way that witnesses are always inclined to vote honestly rather than dishonestly.

VI. Conclusion

In this paper, we revisited the original vision of the TAST project and defined requirements for efficient cross-blockchain token transfers. We summarized our previous work and explained the limitations of the developed prototypes. We then introduced two approaches that are currently being explored for overcoming the described limitations. One approach uses cryptographic proofs while the other one relies on reward-incentivized voting contests to determine the validity of transfers. While both approaches come with their own set of challenges, both ultimately aim to restrict the interaction of a cross-blockchain token transfer to the two blockchains directly involved.

Disclaimer

Acknowledgment

The work presented in this paper has received funding from Pantos GmbH within the TAST research project.

References

[1] M. Borkowski et al. *Caught in Chains: Claim-First Transactions for Cross-Blockchain Asset Transfers*. 2018. URL: http://dsg.tuwien.ac.at/staff/mborkowski/pub/tast/tast-white-paper-2.pdf. White Paper, Technische Universität Wien. Version 1.1. Accessed 2019-05-20.

[2] M. Borkowski et al. *Deterministic Witnesses for Claim-First Transactions*. 2018. URL: http://dsg.tuwien.ac.at/staff/mborkowski/pub/tast/tast-white-paper-3.pdf. White Paper, Technische Universität Wien. Version 1.0. Accessed 2019-05-20.

[3] M. Borkowski et al. *Towards Atomic Cross-Chain Token Transfers: State of the Art and Open Questions within TAST*. 2018. URL: http://dsg.tuwien.ac.at/staff/mborkowski/pub/tast/tast-white-paper-1.pdf. White Paper, Technische Universität Wien. Version 1.2. Accessed 2019-05-20.

[4] M. Borkowski et al. *Cross Blockchain Technologies: Review, State of the Art, and Outlook*. 2019. URL: http://dsg.tuwien.ac.at/staff/mborkowski/pub/tast/tast-white-paper-4.pdf. White Paper, Technische Universität Wien. Version 1.0. Accessed 2019-05-20.

[5] V. Buterin. *Merkling in Ethereum*. URL: https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/. Accessed 2019-05-20.

[6] T. M. Fernández-Caramés et al. "A Review on the Use of Blockchain for the Internet of Things". In: *IEEE Access* 6 (2018), pp. 32979–33001.

[7] I. Konstantinidis et al. "Blockchain for Business Applications: A Systematic Literature Review". In: *Business Information Systems*. Ed. by W. Abramowicz et al. Springer, 2018, pp. 384–399.

[8] *Litecoin*. URL: https://litecoin.org/. Website. Accessed 2019-02-14.

[9] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. White Paper.

[10] A. Narayanan et al. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press, 2016.

[11] F. Tschorsch et al. "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies". In: *IEEE Communications Surveys & Tutorials* 18 (2016), pp. 2084–2123.

[12] J. Willett et al. *Omni Protocol Specification*. 2017. URL: https://github.com/OmniLayer/spec. Version 0.5 (Git: 2017-01-23). Accessed 2019-02-14.

[13] G. Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. 2018. URL: https://ethereum.github.io/yellowpaper/paper.pdf. White Paper. Version 69351d5, 2018-12-10. Accessed 2019-02-14.