IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 23, 2019, accepted July 22, 2019, date of publication August 12, 2019, date of current version August 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2934707

Dextt: Deterministic Cross-Blockchain Token Transfers

MICHAEL BORKOWSKI[1], MARTEN SIGWART[1], PHILIPP FRAUENTHALER[1], TANELI HUKKINEN[2], AND STEFAN SCHULTE[1]
[1]Distributed Systems Group, TU Wien, 1040 Vienna, Austria
[2]Pantos GmbH, 1020 Vienna, Austria

Corresponding author: Michael Borkowski (m.borkowski@infosys.tuwien.ac.at)

This work was supported in part by the Pantos GmbH through the TAST Research Project, and in part by the TU Wien University Library through its Open Access Funding Programme.

ABSTRACT Current blockchain technologies provide very limited interoperability. Restrictions with regard to asset transfers and data exchange between different blockchains reduce the usability and comfort of users, and hinder novel developments within the blockchain space. As a first step towards blockchain interoperability, we propose the DeXTT cross-blockchain transfer protocol, which can be used to record a token transfer on any number of blockchains simultaneously in a decentralized manner. We provide a reference implementation using Solidity, and evaluate its performance. We show logarithmic scalability of DeXTT with respect to the number of participating nodes, and analyze cost requirements of the transferred tokens.

INDEX TERMS Blockchain interoperability, cross-blockchain proof problem, eventual consistency, claim-first transactions, deterministic witnesses.

I. INTRODUCTION

Blockchain technologies, the underlying mechanism of cryptocurrencies, have gained significant interest in both industry and research [1]. After the feasibility of decentralized ledgers has been demonstrated by Bitcoin [2], significant investment into research and development related to blockchains and cryptocurrencies has been ignited. Outcomes of these research and development activities add additional layers on top of existing blockchain implementations [3], [4], provide improvements of Bitcoin itself [5], and facilitate entirely new blockchains [6], which provide sophisticated concepts, such as smart contracts [7]. Furthermore, there is substantial research on potential use cases of blockchains in various economic, social, political, and engineering fields [8]. Nevertheless, the ways in which blockchains could potentially interact with each other remain mostly unexplored.

The constant increase in the number of independent, unconnected blockchain technologies causes significant fragmentation of the research and development field, and poses challenges for both users and developers of blockchain technologies [9]. On the one hand, users have to choose which currency and which blockchain to use. Choosing novel, innovative blockchains enables users to utilize new features and to take advantage of state-of-the-art technology. However, users also risk the loss of funds if the security of such a novel blockchain is subsequently breached, potentially leading to a total loss of funds [10]. Choosing mature, well-known blockchains reduces the risk of such losses, since these blockchains are more likely to have been analyzed in depth [11], but innovative features of novel blockchains may remain unavailable.

On the other hand, when designing decentralized blockchain-based applications, currently, developers must decide which blockchain to base their application on. This can form a substantial impedance to research and technical progress, since individual technologies form isolated solutions, and interoperability between blockchains is mostly not given.

We therefore aim to enable blockchain interoperability. As an overarching goal, we seek to provide means of interaction between blockchains, including cross-blockchain data transmission, cross-blockchain smart contract interaction, and cross-blockchain currency transfer. As a first step to enable such blockchain interoperability, we propose a protocol for cross-blockchain asset transfers, using tokens not locked within an individual blockchain. Instead, it can be used on any number of blockchains, and its transactions are autonomously synchronized across blockchains in a decen-

The associate editor coordinating the review of this article and approving it for publication was Bhaskar Prasad Rimal.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

tralized manner. The protocol presented in this paper does not depend on any single blockchain in particular, and tolerates total failure of any number of blockchains it is used on, as long as there is at least one blockchain remaining to operate on. Our solution prevents double spending, is resilient to the cross-blockchain proof problem (XPP) [12], and does not need external oracles or other means of cross-blockchain communication to function. We provide a reference implementation using Solidity, and evaluate its performance with regard to time and cost.

The contributions of this manuscript are as follows:

- We provide a formal definition of a blockchain's structure, its consensus mechanisms, and discuss the XPP, showing that strict consistency between blockchains is not feasible in practice.
- Due to this consistency limitation, we discuss how to provide eventual consistency for cross-blockchain token transfers by utilizing concepts such as claim-first transactions and deterministic witnesses.
- We formally define *Deterministic Cross-Blockchain Token Transfers* (DeXTT), a protocol implementing eventual consistency for cross-blockchain token transfers.
- We provide a reference implementation in Solidity, presenting and evaluating DeXTT.

The remainder of this paper is structured as follows. In Section II, we discuss underlying technologies, provide a brief discussion of blockchains, transaction types, claim-first transactions, and witness rewards, outline the XPP, and define notation used throughout this work. Section III presents the transfer protocol in detail, and Section IV provides an evaluation of our approach. Section V gives an overview of related work. Finally, Section VI summarizes and concludes the paper.

II. BACKGROUND

Our work aims at providing a protocol for cross-blockchain asset transfers, ensuring that such transfers are performed in a decentralized and trustworthy manner. Assets can be represented on blockchains in various ways. Apart from native currencies (e.g., Ether on the Ethereum blockchain, or Bitcoin on the Bitcoin blockchain), there are other types of assets, commonly called *tokens* [13]. In the recent past, various asset types with different properties have been discussed, such as fungibility, divisibility, and types of implementation like the *User-Issued Asset* (UIA) and *Unspent Transaction Output* (UTXO) models. We refer to our previous work for a thorough analysis [13].

In the work at hand, we discuss a token type that is able to exist on a given number of blockchains simultaneously, i.e., a *pan-blockchain token*. We refer to an implementation of this token as *PBT*. PBTs are not locked to a single blockchain and can be traded using the DeXTT protocol, which ensures synchronization of token balances across blockchains. We refer to the set of blockchains participating in this protocol as an *ecosystem of blockchains*. According to the DeXTT protocol, a wallet $\mathcal{W}_w$ is holding a given number of PBTs not only on a given blockchain, but on all blockchains in the ecosystem in an equal amount. We demonstrate this in Tables 1 and 2. In Table 1, we show traditional assets held on three different blockchains, $\mathcal{C}_a$, $\mathcal{C}_b$, and $\mathcal{C}_c$. In contrast, Table 2 shows that PBT balances are synchronized across blockchains. Thus, a transfer from $\mathcal{W}_w$ to another wallet $\mathcal{W}_v$ is required to be recorded on all participating blockchains, and there must be consensus among all participating blockchains about the balance of each wallet. We use such a model to avoid having to rely on one particular blockchain.

TABLE 1. Examble balances of wallets W_a, W_b, and W_c using traditional blockchain assets.

Blockchain $\mathcal{C}_a$	Blockchain $\mathcal{C}_b$	Blockchain $\mathcal{C}_c$
W_a: 4 ETH W_b: 2 EOS	W_b: 1 LTC	W_a: 3 BTC W_c: 1 USDT

TABLE 2. Examble balances of wallets W_a, W_b, and W_c using PBTs synchronized by DeXTT.

Blockchain $\mathcal{C}_a$	Blockchain $\mathcal{C}_b$	Blockchain $\mathcal{C}_c$
W_a: 2 PBT W_b: 3 PBT W_c: 1 PBT	W_a: 2 PBT W_b: 3 PBT W_c: 1 PBT	W_a: 2 PBT W_b: 3 PBT W_c: 1 PBT

In contrast to keeping assets on a single blockchain, synchronizing their balances across blockchains allows users to react to security breaches in blockchains. For instance, if an attacker manages to illegally modify balances on a blockchain, all other blockchains remain synchronized and the attacker has no way to propagate the illegal changes. In addition, as blockchains are publicly readable, any such breach would rapidly reduce the community's trust in the blockchain, and the compromised blockchain would simply be abandoned.

A. NOTATIONS AND CONVENTIONS

In the following, we use particular notations for concise description of certain objects: We denote blockchains as $\mathcal{C}$, identifying them with a subscript letter, e.g., $\mathcal{C}_a$. Additionally, we denote wallets as $\mathcal{W}$ with a subscript letter, e.g., $\mathcal{W}_s$, $\mathcal{W}_d$, or $\mathcal{W}_w$. A wallet consists of a pair of corresponding keys, out of which one is a public key, and one is a private key. When referring to a token transfer in general, $\mathcal{W}_s$ is used to denote the source (sending) wallet, $\mathcal{W}_d$ is used to denote the destination (receiving) wallet, and $\mathcal{W}_w$ denotes a witness as discussed in Section II-C.

In this work, we use the concept of *transactions* to denote actions executed on a blockchain which modify the blockchain state. We use the expression "$\mathcal{W}_w$ posts the transaction TRANS on $\mathcal{C}_c$" to describe the conceptual protocol, where TRANS is the transaction type used (one out of CLAIM, CONTEST, FINALIZE, VETO, and FINALIZE-VETO, as presented

in Section III). In a scenario where smart contracts are used, this translates to the key pair of $\mathcal{W}_w$ being used to sign a call to the smart contract (on blockchain $\mathcal{C}_c$), where the function `trans()` is invoked. For certain transactions, we define preconditions (e.g., sufficient balances), which can be implemented as checks within the smart contract function. The transactions posted by wallets can either originate from the action of a user, or be initiated by a program (e.g., a wallet application) acting autonomously.

To denote our transactions, we use the notation as shown in (1), where TRANS is the transaction type used, $\mathcal{W}_w$ is the wallet (i.e., the pair of keys) used to sign and post the transaction, a, b, and c denote data contained in the transaction (i.e., the arguments), and σ is the signature when using the private key of $\mathcal{W}_w$ to sign the data $[a, b, c]$. For brevity, we use only σ to denote a multivariate value, e.g., a three-variate ECDSA signature.

$$\mathcal{W}_w : \text{TRANS} \left[a,\ b,\ c \right]_\sigma \tag{1}$$

We denote a transfer of x PBTs from $\mathcal{W}_s$ to $\mathcal{W}_d$ as $\mathcal{W}_s \xrightarrow{x} \mathcal{W}_d$. Furthermore, we denote the PBT balance of $\mathcal{W}_w$ recorded on $\mathcal{C}_c$ as $\mathcal{C}_c : \mathcal{W}_w$.

B. THE CROSS-BLOCKCHAIN PROOF PROBLEM

When developing blockchain interoperability, we aim at enabling interaction between blockchains. As described in Section I, this includes cross-blockchain data transmission, smart contract interaction, and currency transfer. All of these tasks require consistency between blockchains, i.e., data on one blockchain must be consistent with data on another blockchain. This implies that the presence of data on one blockchain must be a strict and reliable indication of related data on another blockchain.

In our scenario, where we aim to synchronize token balances on various blockchains, we might envision a naïve approach where strict consistency is supported across blockchains, i.e., a token transfer recorded on one blockchain can be directly and indubitably detected on another blockchain. This would allow easy synchronization of token balances. However, in this section, we show formally that such strict consistency between blockchains would require constraints which are not feasible in practice using contemporary blockchains, deeming strict consistency between blockchains impossible.

1) DEFINITIONS

To the best of our knowledge, in existing literature, there is no formal definition of blockchains in general. This is mostly due to the variety of existing blockchain technologies and implementations. Some blockchains, e.g., Ethereum [6], are formally defined, but many others are only defined by their source code, and lack formal documentation or definition.

Since we aim to reason about aspects of blockchain interoperability, we require term definitions applying to as many blockchains as possible. We have therefore collected definitions, notation and wording from existing literature [6], [7], [14], and in alignment with this literature to the greatest extent possible, provide our own definitions of certain technical terms in the following.

Definition 1 (Blockchain): A *blockchain* $\mathcal{C}$ is a distributed, decentralized, periodically growing, publicly writeable, append-only data structure consisting of *blocks*, with a defined genesis block, transaction consensus, and arbitration consensus.

Definition 2 (Blocks, Genesis Block): Within a blockchain, a *block* B is a data structure linked to one parent block $P(B)$, containing arbitrary payload data. The graph of linked blocks must not contain cycles, and the *genesis block*, denoted as B_0, is the only block without a parent.

Definition 3 (Lineage): The *lineage* of a block B, denoted as $\text{lin}(B)$, is the line of descent of B from the genesis block B_0, i.e., $\text{lin}(B) = \text{lin}(P(B)) \cup B$, where $\text{lin}(B_0) = B_0$.

Since blocks must not form cycles, the lineage of a block is always a finite set.

Definition 4 (Transaction Consensus): The *transaction consensus* is a well-defined set of properties a block B needs to have in order to be deemed a *valid* next block of its lineage $\text{lin}(B)$.

We define the transaction consensus as the function $\text{tc} : \mathbb{B}^* \times \mathbb{B} \to \{0, 1\}$ (where $\mathbb{B}$ is the set of all possible blocks, and $\mathbb{B}^*$ is the set of all possible finite sets of blocks with arbitrary cardinality):

$$\text{tc}(\text{lin}(B), B) = \begin{cases} 1 & \text{if } B \text{ is a valid descendant of } \text{lin}(B) \\ 0 & \text{otherwise} \end{cases}$$

The function tc decides whether a block B is a valid successor block of the lineage $\text{lin}(B)$, i.e., it returns a (boolean) decision value. Note that this definition implies that a blockchain is self-contained, i.e., the validity of each block can be decided by only regarding its lineage (that is, without taking into account external, off-chain data).

The transaction consensus defines the structure of a blockchain. For Bitcoin, it consists of the block and transaction structure, and the definition of the Script opcodes and their effects. For Ethereum, the transaction consensus consists of the definition of data structures required for blocks, the storage and memory definition and the opcodes of the Ethereum Virtual Machine (EVM).

The entirety of valid blocks forms a tree with the genesis block at the root. There can be multiple blocks which are currently not referenced by any other block as parents, i.e., *leaf blocks*. Blockchains therefore require a method of determining which leaf block to use (e.g., to reference as parent for a newly created block).

We call this process arbitration and define a corresponding consensus:

Definition 5 (Arbitration Consensus): The *arbitration consensus*, given a set of leaf blocks, deterministically returns one preferred leaf block.

The arbitration consensus is hard-coded into each node participating in the blockchain. For instance, in the original implementation of Bitcoin, the arbitration seeks the longest chain, i.e., the block with the longest lineage wins.[1] For Ethereum, the leaf of the lineage with the highest total difficulty (an attribute of Ethereum blocks) is selected. The arbitration consensus only considers valid blocks.

Definition 6 (Main Chain, Orphans): The *main chain* consists of the leaf node currently selected by the arbitration consensus, together with its lineage. Valid blocks not part of the main chain are called *orphans*.

Definition 7 (Data Containment): Data D is *included in* blockchain $\mathcal{C}$, denoted as $D \in \mathcal{C}$, iff[2] D is part of a valid block within the main chain of $\mathcal{C}$.

Data within blocks which are not on the main chain is not regarded as the canonical state of the blockchain as a whole. For the purpose of this work, when examining whether certain data is *included in* a certain blockchain, we are technically interested whether the data is included in a block on the main chain. However, no node can be certain that the chain it currently regards as the main chain is not superseded by another chain, unknown to the node [1]. We can therefore only realistically answer questions regarding whether data is or is not part of any (valid) block, either on the main chain, or on the lineage of an orphan.

Note that due to the aforementioned variety of different existing blockchain technologies, it is not trivial to generalize definitions to include all implementations. However, our definitions cover the most commonly used blockchain technologies, including but not limited to Bitcoin [2], Ethereum [6] and its fork Ethereum Classic [15], as well as Litecoin [5], Dash [16] and Waves [17].

2) CROSS-BLOCKCHAIN PROOFS

We now return to the XPP and, for the sake of reasoning, assume that the creation of a cross-blockchain proof for ensuring strict consistency is indeed possible. For our purposes, this means that the presence of this proof implies strictly the presence of the data to be proven:

Assumption 1: For any $D \in \mathcal{C}_b$, data $D_{\text{proof}} \in \mathcal{C}_a$ can serve as reliable proof that $D \in \mathcal{C}_b$, such that $D_{\text{proof}} \in \mathcal{C}_a \implies D \in \mathcal{C}_b$.

Without loss of generality, we assume that D_{proof} is included in B_a on $\mathcal{C}_a$, and D is included in B_b on $\mathcal{C}_b$. We denote tc_a as the transaction consensus of $\mathcal{C}_a$, and tc_b as the transaction consensus of $\mathcal{C}_b$.

$D \in \mathcal{C}_b$ holds iff $\text{tc}_b(\text{lin}(B_b), B_b) = 1$, i.e., if B_b is valid according to the transaction consensus. Since tc accepts the lineage $\text{lin}(B_b)$, in general, the outcome can depend on any data within $\text{lin}(B_b)$.

Accordingly, $D_{\text{proof}} \in \mathcal{C}_a$ holds iff $\text{tc}_a(\text{lin}(B_a), B_a) = 1$. However, due to Assumption 1, $D_{\text{proof}} \in \mathcal{C}_a \implies D \in \mathcal{C}_b$ holds, so we arrive at:

$$D_{\text{proof}} \in \mathcal{C}_a \iff \text{tc}_a(\text{lin}(B_a), B_a) = 1 \implies \text{tc}_b(\text{lin}(B_b), B_b) = 1 \iff D \in \mathcal{C}_b$$

Thus, $\text{tc}_a(\text{lin}(B_a), B_a) = 1 \implies \text{tc}_b(\text{lin}(B_b), B_b) = 1$ follows, i.e., the transaction consensus of $\mathcal{C}_a$ verifying the validity of B_a must verify the validity of B_b, which depends on its lineage $\text{lin}(B_b)$ according to Definition 4. Note that this does not mean that tc_b necessarily requires all of the lineage data $\text{lin}(B_b)$. Nevertheless, in the general case, the entire lineage data of B_b might be required for verifying the validity of B_a.

In addition, we observe that the outcome of tc_a with regard to B_a must be equivalent to the outcome of tc_b with regard to B_b. More formally, there must exist a mapping $m : \mathbb{B}^* \times \mathbb{B} \to \mathbb{B}$, where for each $m(\text{lin}(\beta), \beta) = \alpha$, it holds that $\text{tc}_a(\text{lin}(\alpha), \alpha) = 1 \implies \text{tc}_b(\text{lin}(\beta), \beta) = 1$. In other words, for every block β (and its lineage) on $\mathcal{C}_b$ where tc_b evaluates to 1, a block α must exist or be createable on $\mathcal{C}_a$ where tc_a returning 1 for α is a sufficient condition of tc_b returning 1 for β. In simpler terms, tc_a must be able to *mimic* tc_b.

Summarizing the above, there are two main challenges to cross-blockchain proofs: (i) proving $D \in \mathcal{C}_b$ on $\mathcal{C}_a$ requires the inclusion of the necessary subset (potentially all) of the data of $\text{lin}(B_b)$ on $\mathcal{C}_a$, and (ii) additionally, tc_a must be powerful enough to mimic tc_b.

From the above reasoning, we return to the original intuition that the presence of certain data (e.g., a specific transaction) on a given blockchain is *rooted* in this blockchain, and that it cannot be verified without verifying the entire blockchain:

Lemma 1 (Lemma of Rooted Blockchains): For any $D \in \mathcal{C}_b$, the existence of $D_{\text{proof}} \in \mathcal{C}_a \iff D \in \mathcal{C}_b$ implies (i) access to the lineage of the block containing D on $\mathcal{C}_a$, and (ii) a sufficiently powerful transaction consensus tc_a to mimic tc_b.

3) IMPLICATIONS

Lemma 1 presents two requirements for verification of the presence of data on $\mathcal{C}_b$ within $\mathcal{C}_a$, namely (i) the presence of a subset of the block lineage of $\mathcal{C}_b$ on $\mathcal{C}_a$, and (ii) the verifiability of the transaction consensus of $\mathcal{C}_b$ by the transaction consensus of $\mathcal{C}_a$.

Considering the first requirement, the practical implication is that blocks stored on $\mathcal{C}_b$ must, in one way or another, be stored on $\mathcal{C}_a$. While for a given instance, the subset of required lineage blocks might be small, in general, this can affect many or all blocks of $\mathcal{C}_b$. Compression algorithms can be used to minimize the amount of data required for this storage. However, in practice, data stored on blockchains is already stored in a relatively minimized form, since storage space is relatively expensive in blockchains [18], [19]. This means that both $\mathcal{C}_a$ and $\mathcal{C}_b$ can be assumed to be coded in a relatively storage-saving form (i.e., with high entropy).

[1] The original paper [2] defines "Nodes always consider the longest chain to be the correct one", but implementations use block difficulty. Elaborating on this differentiation is outside of the scope of our work.

[2] "iff" is equivalent to "if and only if"

Furthermore, even if compression is feasible, information has a lower limit on required storage space [20].

Since storage is expensive, storing a (potentially large) subset of a blockchain's block history on another blockchain is infeasible. We therefore argue that this aspect alone makes cross-blockchain proofs impossible under practical considerations.

Furthermore, we consider the second requirement. Even if the block history necessary for verification is provided, according to the second requirement, the transaction consensus tc_a must be able to validate blocks on $\mathcal{C}_b$. In practice, blockchains use a transaction consensus consisting of simple operations stored in transactions (e.g., Script, the scripting system used by Bitcoin), or, in more complex cases, smart contracts (e.g., EVM).

In practical terms, this implies that the instruction set used by $\mathcal{C}_a$ must be able to simulate the instruction set used by $\mathcal{C}_b$. In cases both chains use Turing-complete virtual machines, such as the EVM, this is the case. However, in other cases, such as Script, verifying more complex blockchains (such as Ethereum) imposes a limitation with regard to computational complexity.

C. CROSS-BLOCKCHAIN BALANCE CONSISTENCY

We have seen that due to the XPP, strict consistency between blockchains is not possible in practice. This means that we cannot use a traditional, strictly consistent protocol for transferring token balances. Therefore, in our proposal, we relax this requirement to eventual consistency, i.e., we accept temporary disagreement with regard to balances, as we show in the following. In practice, blockchains themselves only provide eventual consistency, since there is no guarantee when data submitted to the network will be included in a block [1]. Therefore, using eventual consistency for synchronizing data between blockchains is a feasible approach.

For the purpose of this paper, we follow the assumption that each user of DeXTT is generally interested in all operational (non-failed) blockchains in an ecosystem, and specifically, in the consistency of their balance across all blockchains. A blockchain is assumed to be *failed* if for some reason public trust into it is lost, e.g., due to a published exploit.

Therefore, all interested parties (i.e., wallet holders) are monitoring all blockchains in the ecosystem, and if a party participates in the protocol on one blockchain, it also participates on all other blockchains. We motivate this by defining that any inconsistency in wallet balances between non-failed blockchains effectively renders the wallet useless.

We propose to achieve eventual consistency using *claim-first transactions* [12]. While traditionally, blockchain transfers disallow claiming tokens before they have been marked as spent, we explicitly decouple the required temporal order of SPEND $\rightarrow$ CLAIM and allow its reversal, i.e., claiming tokens before spending them. In our case, for a certain period of time, tokens are allowed to exist in the balance of both the sender and the receiver (on different blockchains), namely until the information is propagated to all blockchains. In the presented protocol, we provide a mechanism to enforce eventual spending of the tokens in the sender balance, as described in Section III.

In order to ensure such eventual consistency, we rely on parties observing a transfer to propagate this information across blockchains. These parties are denominated as *observers*. A monetary incentive is provided for any observer in order to ensure propagation. We use part of the transferred PBTs for these *witness rewards*. The main challenge of this approach is the decision which observer receives the reward. Using a first-come-first-serve basis is not feasible, since it is possible that on one blockchain, one observer is the first to propagate the transfer and claim the reward, while on another blockchain, another observer takes this place. This would lead to two different observers receiving a reward on two different blockchains, and therefore, to potentially inconsistent balances.

In this work, we address this problem by using *deterministic witnesses* [21]. In short, instead of using a first-come-first-serve reward distribution, we define a *witness contest*. Its duration is fixed to a validity period, *contestants* (i.e., observers aiming to become reward candidates) can register for the contest, and the decision of who wins the contest is made deterministically and predictably by each blockchain at the end of the contest. In Section III, we propose an approach for deciding the winning witness in a way that is fair (i.e., all contestants have the same chance of winning), while at the same time, it is purely deterministic, and—given the assumptions discussed above—assures all blockchains reach the same decision about assigning witness rewards.

Our approach therefore solves the problem of assigning witness rewards, which is required as an incentive for observers of a cross-blockchain transfer to propagate this transfer information, ensuring eventual consistency across the ecosystem of blockchains.

D. CRYPTOGRAPHIC SIGNATURES AND HASHES

In our approach, we make extensive use of cryptographic signatures and hashes, which are essential for blockchains themselves. For instance, the ECDSA algorithm [22] is used by Ethereum for creating and verifying signatures, and is also implemented natively and available to the EVM [23]. We use Solidity, the smart contract language of Ethereum, for the reference implementation of DeXTT. However, we note that DeXTT is not limited to Solidity or the EVM, and other blockchains offering signatures and hash algorithms can very well be used. The only crucial property required by our approach is a distribution of hash values which is approximately uniform. KECCAK256, the hash algorithm used by Ethereum, satisfies this requirement [24], as does the SHA-256 algorithm used by Bitcoin [25].

III. DECENTRALIZED CROSS-BLOCKCHAIN TRANSFERS

In the following, we present the DeXTT protocol, together with an example transaction. In our example, we consider

TABLE 3. Initial state of the involved blockchains at $t = 0$.

Blockchain $\mathcal{C}_a$		Blockchain $\mathcal{C}_b$		Blockchain $\mathcal{C}_c$	
$\mathcal{W}_s$ balance:	80	$\mathcal{W}_s$ balance:	80	$\mathcal{W}_s$ balance:	80
$\mathcal{W}_d$ balance:	0	$\mathcal{W}_d$ balance:	0	$\mathcal{W}_d$ balance:	0
$\mathcal{W}_w$ balance:	0	$\mathcal{W}_w$ balance:	0	$\mathcal{W}_w$ balance:	0

three blockchains participating in cross-blockchain transfers, $\mathcal{C}_a$, $\mathcal{C}_b$, and $\mathcal{C}_c$. Note, however, that our approach is applicable to an arbitrary number of blockchains. Furthermore, we consider the wallets $\mathcal{W}_s$, $\mathcal{W}_d$, $\mathcal{W}_u$, $\mathcal{W}_v$, and $\mathcal{W}_w$. We assume that initially, $\mathcal{W}_s$ has 80 PBTs, and all other wallets have a balance of zero (see Table 3). We furthermore use a fixed reward of 1 PBT for the witness propagating this transaction across the blockchain ecosystem. Note that pro rata fees (e.g., 1% of the transferred PBTs, or an amount selected by the sender) are also possible and the exact fee model is an economic choice. We will discuss this in more detail in Section IV-B.

As discussed in Section II-C, claim-first transactions require all blockchains within the ecosystem to maintain and synchronize token balances. Therefore, the initial situation is as depicted in Table 3. Balances for $\mathcal{W}_u$ and $\mathcal{W}_v$ are not shown, as they will remain zero throughout the example.

A. TRANSFER INITIATION

In the following, we assume that $\mathcal{W}_s$ intends to transfer 20 PBTs to $\mathcal{W}_d$, i.e., reduce the PBT balance of $\mathcal{W}_s$ by 20, increase the PBT balance of $\mathcal{W}_d$ by 19 (20 reduced by 1, the witness reward), and increase the PBT balance of a (yet to be decided) witness wallet by 1. As stated in Section II-C, we only require eventual consistency for this transfer, i.e., a temporary overlap is allowed where $\mathcal{W}_d$ has already received 19 PBTs, but the balance of $\mathcal{W}_s$ is still unchanged.

Therefore, $\mathcal{W}_s$ signs this intent, confirming that indeed, 20 PBTs—minus 1 PBT of witness reward—are to be transferred to $\mathcal{W}_d$. Furthermore, we define a validity period for the transfer, which denotes the time during which the witness selection for the transfer has to take place. In our example scenario, this time span lasts for 1 minute. However, this time can be set significantly shorter or longer, depending on the use case. We provide an analysis of the impact of this parameter in Section IV-A.

We denote the entirety of the sender's intent using the notation shown in (2), where $[t_0, t_1]$ is the validity period, and α denotes the signature of the entire content of the brackets by $\mathcal{W}_s$. The resulting signature itself is denoted as α. We use the ECDSA algorithm, natively supported by the EVM, for all signatures. However, as pointed out in Section II-D, other algorithms could also be used, assuming that their verification is supported on all involved blockchains.

$$\left[\mathcal{W}_s \xrightarrow{x} \mathcal{W}_d,\ t_0,\ t_1 \right]_\alpha \tag{2}$$

The data contained in (2) is transferred to the receiving wallet $\mathcal{W}_d$. This transfer can happen on any blockchain within the ecosystem, or using an off-chain channel. Since all of the data contained in (2) will be published throughout the DeXTT transaction, this channel does not need to be secure, and we do not specifically define any communication means. The receiving wallet then counter-signs the data from (2) using its respective private key, yielding the entire *Proof of Intent* (*PoI*), as shown in (3).

$$\left[\mathcal{W}_s \xrightarrow{x} \mathcal{W}_d,\ t_0,\ t_1,\ \alpha \right]_\beta \tag{3}$$

The PoI contains all information necessary to prove to any blockchain (i.e., to its smart contracts and miners) that the transfer is authorized by the sender and accepted by the receiver. The receiver can now post this PoI using a transaction we call CLAIM. This transaction allows the receiver to publish the PoI in order to later claim the transferred PBTs. The receiver can post this on any blockchain within the ecosystem, and does not need to post it on more than one blockchain. The CLAIM transaction is defined and noted as shown in (4).

$$\mathcal{W}_d : \text{CLAIM} \left[\mathcal{W}_s \xrightarrow{x} \mathcal{W}_d,\ t_0,\ t_1,\ \alpha \right]_\beta \tag{4}$$

The preconditions for the CLAIM transaction are (i) that the PoI is valid (i.e., that the signatures α and β are correct), (ii) that the balance of the source wallet $\mathcal{W}_s$ is sufficient, (iii) that the PoI is not expired, i.e., that t_1 has not yet passed, and (iv) that no PoI is known to the blockchain on which it is posted with an overlapping validity period and the same source wallet $\mathcal{W}_s$. In other words, a wallet must not sign an outgoing PoI while another outgoing PoI is still pending. This is done in order to prevent a double-spending attack, where two PoIs are signed which are conflicting, i.e., which, if both were executed, would reduce the sender's balance below zero.

The purpose of the CLAIM transaction is the publishing of the PoI, which can then be propagated across the blockchain ecosystem as described later.

In our example, we assume that the receiver $\mathcal{W}_d$ posts the CLAIM transaction (containing the PoI) on $\mathcal{C}_a$ as shown in (5), where 1 and 61 mark the validity period in seconds (i.e., one minute total validity), `0xAA` is assumed to be the signature α, and `0xBB` is assumed to be the signature β. For brevity, one-byte signatures are used for demonstration in this example. Naturally, in reality, the signature hashes are longer (e.g., 32 bytes for KECCAK256).

$$\mathcal{W}_d : \text{CLAIM} \left[\mathcal{W}_s \xrightarrow{20} \mathcal{W}_d,\ 1,\ 61,\ \texttt{0xAA} \right]_{\texttt{0xBB}} \tag{5}$$

The CLAIM transaction on $\mathcal{C}_a$ changes the blockchain state as shown in Table 4. We see that the PoI has been stored within $\mathcal{C}_a$, which is referred to by its signature α. The balances remain unchanged on $\mathcal{C}_a$ because the validity period is not yet concluded, i.e., t_1 is not yet reached. Naturally, since no information has been posted yet to $\mathcal{C}_b$ and $\mathcal{C}_c$, these blockchains also remain unchanged.

TABLE 4. State after PoI publication at $t = 1$.

Blockchain $\mathcal{C}_a$		Blockchain $\mathcal{C}_b$		Blockchain $\mathcal{C}_c$	
$\mathcal{W}_s$ balance:	80	$\mathcal{W}_s$ balance:	80	$\mathcal{W}_s$ balance:	80
$\mathcal{W}_d$ balance:	0	$\mathcal{W}_d$ balance:	0	$\mathcal{W}_d$ balance:	0
$\mathcal{W}_w$ balance:	0	$\mathcal{W}_w$ balance:	0	$\mathcal{W}_w$ balance:	0
PoI `0xAA`:					
$\mathcal{W}_s \xrightarrow{20} \mathcal{W}_d$					
$t_1 = 61$					

TABLE 5. State during witness Contest at $t = 2$.

Blockchain $\mathcal{C}_a$		Blockchain $\mathcal{C}_b$		Blockchain $\mathcal{C}_c$	
$\mathcal{W}_s$ balance:	80	$\mathcal{W}_s$ balance:	80	$\mathcal{W}_s$ balance:	80
$\mathcal{W}_d$ balance:	0	$\mathcal{W}_d$ balance:	0	$\mathcal{W}_d$ balance:	0
$\mathcal{W}_w$ balance:	0	$\mathcal{W}_w$ balance:	0	$\mathcal{W}_w$ balance:	0
PoI `0xAA`:					
$\mathcal{W}_s \xrightarrow{20} \mathcal{W}_d$					
$t_1 = 61$					
Contestants:					
$\mathcal{W}_u$ (`0xC2`)					
$\mathcal{W}_v$ (`0xC3`)					
$\mathcal{W}_w$ (`0xC1`)					

B. WITNESS CONTEST

At this point, the information about the intended transfer (the PoI) is only recorded on $\mathcal{C}_a$. However, this information must be propagated to all other blockchains to ensure consistency of balances across blockchains. We use the following mechanism, which we refer to as the *witness contest*, to ensure this consistency.

Any party observing the CLAIM transaction on $\mathcal{C}_a$ can become a contestant, i.e., a candidate for receiving a reward. In order to become a contestant, the party must propagate the PoI across all blockchains in the ecosystem. We define the transaction used for this as CONTEST. This transaction is defined for any arbitrary wallet $\mathcal{W}_o$ as shown in (6), where the new signature ω is the result of the contestant $\mathcal{W}_o$ signing the PoI. This signature will later play a role in determining the winner of the witness contest, as described in Section III-C.

$$\mathcal{W}_o : \text{CONTEST} \left[\mathcal{W}_s \xrightarrow{x} \mathcal{W}_d,\ t_0,\ t_1,\ \alpha,\ \beta \right]_{\omega} \tag{6}$$

The CONTEST transaction can be posted multiple times by various contestants during the validity period. Again, the PoI must be valid and must not violate any PoI's validity period.

In our example, we assume that $\mathcal{W}_u$ is the first to post a CONTEST transaction on $\mathcal{C}_b$ as shown in (7), where again, 1 and 61 denote the validity period, `0xAA` and `0xBB` are the PoI signatures, and `0xC2` is the signature resulting from $\mathcal{W}_u$ signing the PoI. The signature values in this example are chosen arbitrarily in order to demonstrate the subsequent witness contest.

$$\mathcal{W}_u : \text{CONTEST} \left[\mathcal{W}_s \xrightarrow{20} \mathcal{W}_d,\ 1,\ 61,\ \texttt{0xAA},\ \texttt{0xBB} \right]_{\texttt{0xC2}} \tag{7}$$

Next, we assume that the other observers $\mathcal{W}_v$ and $\mathcal{W}_w$ become contestants by posting similar CONTEST transactions. We assume that the resulting signature ω for $\mathcal{W}_v$ is `0xC3`, and that the signature for $\mathcal{W}_w$ is `0xC1`.

$$\mathcal{W}_v : \text{CONTEST} \left[\mathcal{W}_s \xrightarrow{20} \mathcal{W}_d,\ 1,\ 61,\ \texttt{0xAA},\ \texttt{0xBB} \right]_{\texttt{0xC3}} \tag{8}$$

$$\mathcal{W}_w : \text{CONTEST} \left[\mathcal{W}_s \xrightarrow{20} \mathcal{W}_d,\ 1,\ 61,\ \texttt{0xAA},\ \texttt{0xBB} \right]_{\texttt{0xC1}} \tag{9}$$

Transactions (7–9) are eventually posted to $\mathcal{C}_a$, $\mathcal{C}_b$, and $\mathcal{C}_c$. This is because every contestant participating in the contest is interested in participating in all blockchains in the ecosystem to maintain their own consistency.

The state resulting from the three contestants posting to $\mathcal{C}_a$, $\mathcal{C}_b$, and $\mathcal{C}_c$ is shown in Table 5. The blockchain maintains a list of contestants together with their ω signature values.

C. DETERMINISTIC WITNESS SELECTION

After the expiration of t_1, the witness contest ends, a winning witness must be selected, and is awarded with the witness reward. This is performed by the FINALIZE transaction, which must be triggered after t_1.

Conceptually, this transaction is purely time-based. It can be triggered by the receiver, by any other party, or using a decentralized solution like the *Ethereum Alarm Clock* [26]. The latter approach has the advantage of being independent of any party's activity. However, for simplicity, in our current approach and the discussion below, we assume that the destination wallet $\mathcal{W}_d$ posts the FINALIZE transaction on each blockchain. The FINALIZE transaction is defined in (10).

$$\text{FINALIZE} \left[\alpha \right] \tag{10}$$

The FINALIZE transaction only requires the parameter α, identifying the PoI, because the blockchain already contains all necessary information about the PoI. The precondition of t_1 being expired ($t > t_1$) is necessary for the FINALIZE transaction to avoid premature finalization.

The effect of the FINALIZE transaction is that the contest for the PoI referred to by its signature α is concluded. This means that the winning witness is awarded the witness reward, which, according to the introduction of Section III, is 1 PBT in our current approach. Furthermore, the conclusion of the contest performs the actual transfer of PBTs, i.e., x PBTs are deducted from the balance of $\mathcal{W}_s$, and $\mathcal{W}_d$ receives $(x - 1)$ PBTs (x reduced by the witness reward). This action is executed on all blockchains, since FINALIZE is posted on all blockchains.

We define the winning witness to be the contestant with the lowest signature ω (i.e., with its value closest to zero). This signature cannot be influenced by the contestants, since it is

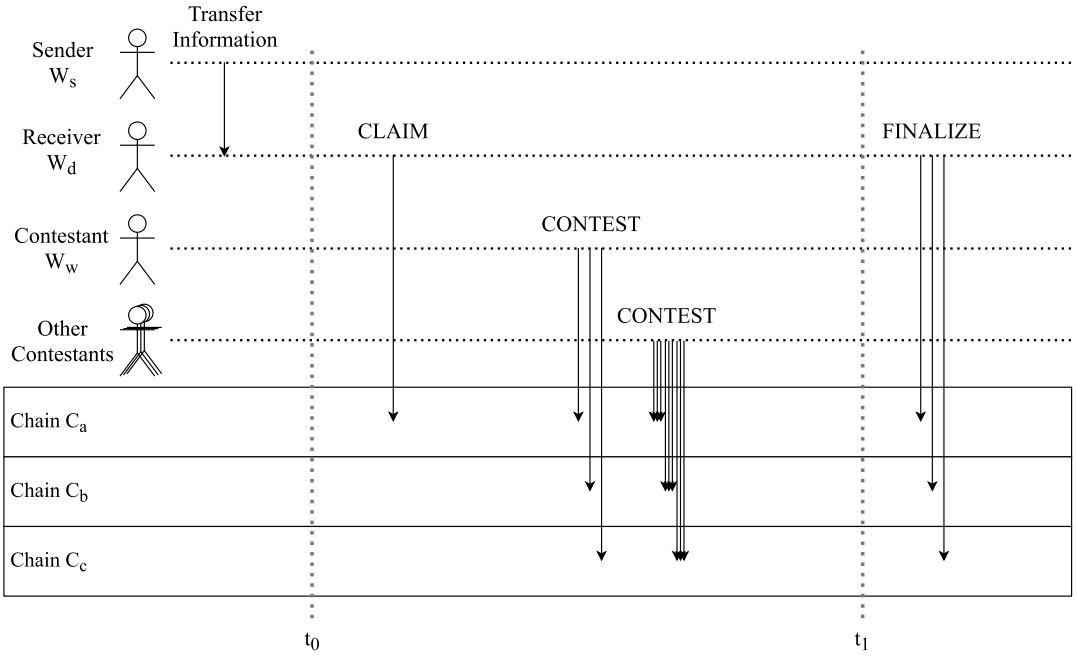

FIGURE 1. Sequence of Transactions within a DeXTT Transfer.

TABLE 6. Final state after witness contest at $t > 61$.

Blockchain $\mathcal{C}_a$		Blockchain $\mathcal{C}_b$		Blockchain $\mathcal{C}_c$	
$\mathcal{W}_s$ balance:	60	$\mathcal{W}_s$ balance:	60	$\mathcal{W}_s$ balance:	60
$\mathcal{W}_d$ balance:	19	$\mathcal{W}_d$ balance:	19	$\mathcal{W}_d$ balance:	19
$\mathcal{W}_w$ balance:	1	$\mathcal{W}_w$ balance:	1	$\mathcal{W}_w$ balance:	1

only formed from the PoI data and the contestants' private key. Accordingly, the contestants have no way of increasing their chances of winning a particular contest, except for creating a large number of wallets (private keys).

Such "mining for wallets" is not a violation of our protocol and no threat to its fairness, since doing so is computationally expensive, and therefore creates cost on its own. There exists a break-even point of the witness reward and the cost created by the creation of a large number of wallets [21]. Effectively, this challenge is comparable to mining in Proof of Work (PoW) in that resources, i.e., computing power, can be traded for rewards.

In our example above, the witness with the lowest ω is $\mathcal{W}_w$, with $\omega = \texttt{0xC1}$. Therefore, this witness is awarded with the witness reward. The final blockchain state is shown in Table 6. The balances of the competing contestants $\mathcal{W}_u$ and $\mathcal{W}_v$ remain zero. The expired PoIs are no longer shown for brevity.

Figure 1 shows an overview of the transactions posted by various wallets on various blockchains. The contestant which ultimately becomes the winning witness ($\mathcal{W}_w$) is shown separately from all other contestants, since this wallet is later assigned the witness reward. We see that first, the sender $\mathcal{W}_s$ provides the receiver $\mathcal{W}_d$ with the transfer information shown in (2). This may happen before or after t_0. Then, not sooner than t_0, the receiver posts a CLAIM transaction to one of the blockchains (in this case, $\mathcal{C}_a$). This is observed by all contestants, which then post CONTEST transactions to all blockchains. Note that the CONTEST transactions do not have to follow any particular order, and can be posted concurrently by all contestants (as depicted in Figure 1). After t_1 expires, the receiver (here: $\mathcal{W}_d$) posts the FINALIZE transaction, which finalizes the transfer and deterministically assigns the witness reward to the contest winner (here: $\mathcal{W}_w$).

D. PREVENTION OF DOUBLE SPENDING

A malicious sender might sign two different PoIs conflicting with each other. For instance, a sender owning 10 PBTs might create two PoIs, transferring 8 PBTs each, to two different wallets. Executing these transfers would reduce the sender's balance by 16 PBTs in total, resulting in -6 PBTs.

In order to prevent such behavior, we introduce the VETO transaction. The VETO transaction can be called by any party noticing two conflicting PoIs (i.e., two PoIs with the same source, different destinations, and overlapping validity periods). Since such PoIs are forbidden by definition, the VETO transaction is used to penalize the sender, and to protect the receiver from losing PBTs due to inconsistent balances.

Since the VETO transaction requires incentive, we propose to use the same technique as presented above, i.e., a contest. Any observer of a PoI conflict can report this conflict using the VETO transaction, and after the expiration of the veto validity period, the observer with the lowest ω signature is assigned a reward.

We therefore define the VETO transaction as shown in (11), where α refers to the original PoI, which is known to the blockchain because it has already been posted on a given blockchain, and the remaining data $\mathcal{W}_s \xrightarrow{x'} \mathcal{W}_{d'}$ and t'_0, t'_1, α' describe the new, conflicting PoI.

$$\mathcal{W}_w : \text{VETO} \left[\alpha,\ \mathcal{W}_s \xrightarrow{x'} \mathcal{W}_{d'},\ t'_0,\ t'_1,\ \alpha' \right]_\omega \tag{11}$$

The VETO transaction, similar to CONTEST, is posted on all participating blockchains. Note that multiple observers can be expected to concurrently post VETO transactions. Therefore, it is possible that on one blockchain, a given PoI (e.g., where $\alpha = \texttt{0x10}$) is posted first, and a second PoI (e.g., where $\alpha' = \texttt{0x20}$) is presented as ''conflicting'' by a VETO transaction, while on another blockchain, the PoI where $\alpha = \texttt{0x20}$ is posted first, and the PoI with $\alpha' = \texttt{0x10}$ is posted in the VETO transaction as ''conflicting''. In the following, we define a behavior for the VETO transaction that still maintains consistency, regardless of the order of PoIs.

The preconditions for VETO are that α refers to a PoI already known to the blockchain, that the conflicting PoI is valid, and that the two PoIs are actually conflicting.

The effects of VETO are as follows: (i) The sender of the conflicting PoIs loses all PBTs, i.e., the balance is set to zero to penalize such protocol-violating behavior. (ii) Any PoI which has a non-expired validity period (i.e., every PoI where $t < t_1$) is canceled. This means that no FINALIZE transaction will be permitted for this PoI, the transfer itself will therefore not be executed, and no witness reward will be assigned. Finally, (iii) a new contest is started, called the *veto contest*. The veto contest is similar to a regular witness contest in that its purpose is the propagation of information (in this case, the information of conflicting PoIs).

We propose to use the same reward for the veto contest as for the regular witness contest (in our case, 1 PBT). Since all PBTs held by the sender are destroyed, and only 1 PBT is assigned to the winner of the veto contest, all remaining PBTs are lost. Furthermore, we propose the validity period expiration of the veto contest, t_{VETO}, to be defined as shown in (12).

$$t_{\text{VETO}} = \max(t_1, t'_1) + \max(t_1 - t_0, t'_1 - t'_0) \tag{12}$$

The definition shown in (12) states that the veto contest is valid until a point in time which is found by taking the later expiration time of the conflicting PoIs ($\max(t_1, t'_1)$) and adding the longer validity period ($\max(t_1 - t_0, t'_1 - t'_0)$). This is done to ensure that sufficient time is available for the veto contest. We note that this is an implementation detail and other approaches (e.g., a fixed period) are also possible.

The veto contest is concluded by a FINALIZE-VETO transaction, defined as shown in (13).

$$\text{FINALIZE-VETO} \left[\alpha,\ \alpha' \right] \tag{13}$$

The effect of the FINALIZE-VETO transaction is similar to that of the FINALIZE transaction, except that no actual transfer is executed. The witness reward is again assigned to the veto contestant—that is, a wallet posting a VETO transaction—with the lowest ω signature in the VETO transaction. Similar to FINALIZE, the FINALIZE-VETO transaction can be called by anyone, in particular, the winning veto contestant has monetary incentive in doing so.

IV. EVALUATION

The approach presented in Section III introduces transactions which change the state of different blockchains within a blockchain ecosystem, according to given rules. This can be implemented using smart contracts, e.g., using the Solidity language [27]—more specifically, the EVM—on the Ethereum blockchain. We use Solidity to create a reference implementation of the proposed protocol for evaluation purposes. The prototype is available as Open Source software at Github.[3] However, other ways of implementing such transactions exist, as we discuss in Section V.

In order to evaluate our approach, we investigate its functionality, performance, and cost impact in an ecosystem of blockchains with agents performing repeated token transfers. We achieve these goals by using our reference implementation consisting of Solidity smart contracts, deploying these smart contracts on a number of private Ethereum-based blockchains, and using testing client software to perform transfers with a given rate.

We ensure a reproducible and uniform ecosystem of blockchains by using three `geth` nodes in Proof of Authority (PoA) mode, creating three private blockchains. We choose PoA to achieve an energy-efficient testing and evaluation platform while being able to perform repeated experiments. Note that the consensus algorithm, i.e., PoW, PoA, or Proof of Stake (PoS), defines the behavior of blockchain nodes between each other and maintains data consistency in the network of a given blockchain [28]. However, the smart contract layer is independent of the consensus algorithm. Therefore, our evaluation on PoA is directly applicable to blockchains with any consensus algorithm, including PoW.

The `geth` nodes used in our experiments can be configured, for instance, with regard to block time and Gas limit. For our evaluation, we have observed the behavior of the live Ethereum blockchain (January 2019) and have configured our nodes to follow this behavior. Therefore, our nodes are configured to use a block time of 13 s on average, and a Gas limit of 8 million Ethereum Gas, mimicking the live Ethereum chain. We use private chains instead of the Ethereum main chain to enable a high number and low cost of repeatable experiments in an automated fashion without depending on external components, such as Ethereum nodes.

We use ten clients constantly and simultaneously initiating transfers within the blockchain ecosystem. This number is chosen as a balance between feasible and reproducible experiments and expected real-world conditions. While it is small compared to evaluations of other classes of distributed

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

systems, we note that the lack of scalability of blockchain technologies is a crucial issue in general, and is seen as one of the main challenges for existing blockchain technologies [29]. We refer to existing literature for a study on how scalability of blockchains can be improved [30].

In our experimental ecosystem, each client constantly transfers random amounts of PBTs to random wallets. If a client owns too few PBTs for a transaction, no transaction is performed until PBTs are available again. After a successful transfer, the client waits for a random time between 15 s and 30 s. Afterwards, the process is repeated throughout the entire experiment duration.

We perform two experiment series, as described in the following sections. The first series is used to evaluate DeXTT scalability and the impact of the transfer validity period, and consists of a series of 30-minute experiments, where each individual experiment uses an increased validity period. The second series consists of 20 experiments, again with a duration of 30 min each, used to measure the average cost of a DeXTT transfer.

A. SCALABILITY AND TIMING

The DeXTT protocol requires one CLAIM transaction per transfer, and for each transfer, one FINALIZE transaction per blockchain. In addition, each contestant posts one CONTEST transaction to each blockchain. We assume that candidates which no longer have a chance to win the witness contest (because a candidate with a lower signature ω for the given transaction is already known) do not post CONTEST transactions to avoid cost. As stated in Section II-D, the uniformly distributed KECCAK256 algorithm is used for signatures. Thus, on the average case, each CONTEST transaction halves the space of remaining possible winning signatures ω (because the expected value of the uniform distribution is the arithmetic mean of the domain). Therefore, with each CONTEST transaction, the likelihood of another candidate existing with a lower ω is halved. Following from this, on average, $\log_2\ n$ candidates will post a CONTEST transaction, where n is the number of total observers.

Transfer time in the DeXTT protocol is directly impacted by the validity period $[t_0, t_1]$ chosen by the sender. We therefore first evaluate the impact of the validity period. Using too short validity periods leads to corrupted transfers, i.e., transfers which cause permanently inconsistent balances, since observers cannot post CONTEST transactions in time. In such scenarios, eventual consistency between blockchains is not guaranteed. As stated above, we use a block time of 13 s, therefore, we start our experiments with 10 s, and increase the period by 5 s with each experiment. We then run our blockchain ecosystem for 30 min using each validity period and record the number of corrupted transactions. Note that we have to reset the inconsistent balances for wallets participating in a corrupted transaction in order to be able to run the experiments for 30 min.

Figure 2 shows the results of these experiments. Beyond 52 s, no corrupt transactions are observed. It becomes clear

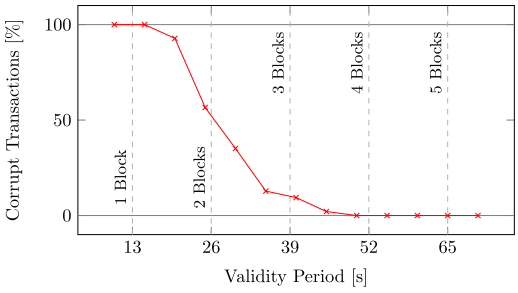

FIGURE 2. Impact of validity period on transaction success.

TABLE 7. Cost analysis.

	Cost (kGas)		Cost (USD)	
Transaction	Mean	σ	Mean	σ
CLAIM	57.7	11.1	0.0668	0.0128
CONTEST	81.5	64.2	0.0943	0.0743
FINALIZE	45.5	0.1	0.0527	< 0.0001
VETO	131.3	91.9	0.1520	0.1063
FINALIZE-VETO	48.6	1.7	0.0563	0.0020

that using the reference implementation and waiting for 4 blocks (52 s) is sufficient for ensuring consistency. Between 1 and 3 blocks (13 s and 39 s, respectively), the amount of corrupted transactions declines with a varying rate.

From this experiment, we conclude that using a validity period with the length of at least 4 blocks (52 s) is sufficient to maintain consistency using our reference implementation.

B. COST ANALYSIS OF DEXTT TRANSFERS

To estimate the cost incurred by DeXTT transfers, we run the same experiment 20 times. Based on our previous experiment, we choose 65 s (5 blocks, i.e., well above the determined limit of 52 s) as the duration of the validity period in each transaction. We record the average cost of each transaction. Table 7 shows an overview of the cost of the individual transactions involved in a DeXTT transfer. For each transaction, we show the mean cost, and its standard deviation, both in thousands of Ethereum Gas (kGas), and in USD. For this, we assume a Gas price of 10 Gwei (1 Ether $= 10^9$ Gwei $= 10^{18}$ wei) and a price of Ether of 115.71 USD. These values were obtained from the Ethereum live chain in January 2019. Note that our implementation is optimized in that CLAIM and CONTEST both use the same smart contract function. Nevertheless, we distinguish the semantic difference (posting of new transfer for CLAIM, and participating in a contest for CONTEST) in the results.

In the following, we assume m blockchains and n total observers. For our calculation, we assume that all observers monitor all blockchains, and post CONTEST transactions if it benefits them. A regular DeXTT transfer (i.e., one which does not contain a conflicting PoI, and therefore requires no veto) consists of one CLAIM transaction (on the target chain), $\log_2\ n$ CONTEST transactions (as discussed in Section IV-A)

on each blockchain, i.e., $m \log_2 n$ CONTEST transactions, and m FINALIZE transactions. The CLAIM transaction is posted by the receiver, and each CONTEST transactions is posted by an observer (thus becoming a contestant). While the FINALIZE transaction can be posted by any party, posting it is beneficial to the receiver (because it finalizes the transfer to the receiver), and therefore it can be expected that the receiver will bear its cost to finalize the transfer.

The expected cost in kGas for a DeXTT transfer are as follows: The receiver bears the cost for one CLAIM transaction (57.7 kGas) and m FINALIZE transactions (45.5 kGas each). Each of the $\log_2 n$ expected observers posting transactions bears the cost for m CONTEST transactions (81.5 kGas each). The sender does not bear any cost.

Assuming a blockchain ecosystem of 10 blockchains, the total transaction cost for the receiver is 0.59 USD. Each of the $\log_2 n$ observers posting transactions bears cost of 0.94 USD. These numbers represent our current reference implementation and can be regarded as an upper bound for DeXTT transfer cost. Any additional optimization to the smart contract code has the potential to further reduce the Gas cost of the individual transactions, and therefore, of the overall DeXTT transfer.

Additionally, these numbers allow us to reason about the economic impact of a currency using DeXTT transactions. Observers pay transaction cost of 0.94 USD, and potentially receive a witness reward, currently defined as 1 PBT. The chance of an observer winning is $\frac{1}{n}$, however, according to the discussion in Section IV-A on average, only $\log_2 n$ out of all n observers are expected to post CONTEST transactions. Hence, the likelihood for an observer posting a transaction to win the contest is $\frac{\log_2 n}{n}$.

Therefore, the investment for each observer is 0.94 USD, the contest reward is 1 PBT, and the winning likelihood is $\frac{\log_2 n}{n}$. From this, it follows that in order for the observer to have incentive to post CONTEST transactions in an ecosystem of $m = 10$ blockchains, the inequation shown in (14) must hold, where p is the price of 1 PBT in USD.

$$\frac{\log_2 n}{n} p > 0.94 \text{ [USD]} \tag{14}$$

In other words, the price of 1 PBT in USD divided by the number of observers must be higher than 0.94. Assuming $n = 10$ observers, the PBT price must be above 2.83 USD. Assuming $n = 100$, the PBT price must be above 14.15 USD. For $n = 1000$, the PBT price must be above 94.32 USD. This implies that as long as these price limits are observed, the DeXTT protocol is economically feasible. Ensuring this property is not in the scope of this paper, as the price of any asset is determined by supply and demand, and therefore the perceived value of PBT influences its price.

Note that these numbers assume $m = 10$ blockchains, and a fixed reward of 1 PBT. A pro rata reward, e.g., 1% of the transferred PBTs, would reduce the required PBT price but would also increase the complexity of calculating the witness incentive. Furthermore, a dynamic reward adaption based on the number of observers, similar to the variable mining rewards in Bitcoin, or a value selected by the sender, similar to the Gas price in Ethereum, can also be used to reduce the required PBT price, and therefore incentivize observers.

V. RELATED WORK

As discussed in Section I, blockchain interoperability can be used to address the fragmentation of the blockchain research field. Yet, to the best of our knowledge, contemporary approaches only cover limited parts of the functionality needed in order to achieve blockchain interoperability.

Initially, the only way to achieve any interoperability between different blockchains was to trade assets (tokens or native currencies) on centralized exchanges, which provide marketplace functionalities. Later on, decentralized exchanges such as Bisq [31] or 0x [32] emerged. Most recently, the Republic protocol [33] has been proposed, which includes a decentralized dark pool exchange, i.e., details about an exchange are kept secret.

All of these approaches, however, are concerned with the *exchange* of assets, generally using atomic swaps [34] for trustless asset exchange. In such an atomic swap, for instance, one party might transfer Bitcoin to another party, while the other party transfers Ether to the first, and each asset remains on its blockchain. In contrast, DeXTT can be used to transfer a single type of asset, and to ensure that the resulting balances are synchronized across blockchains. In our approach, no swap partner is required.

A different approach to tackle blockchain interoperability is presented in PolkaDot [35]. PolkaDot aims to provide a set of tools and techniques for developing applications which share security across blockchains and use an ''inter-chain communication protocol'' (ICMP). PolkaDot uses separate blockchains attached to a trusted blockchain—these separate blockchains are called *parachains*—and defines a set of various actors (validators, collators, and fishermen) to implement its interoperability features. In contrast, we build on top of existing blockchains, do not assume trust in any particular blockchain, and only require one type of actors, i.e., observers, which may become a winning witness during the contest. This means that while applications must be built on top of PolkaDot and use the PolkaDot ecosystem to use ICMP, DeXTT is a protocol which can be used as an extension of existing solutions, and the blockchains used can be chosen by the user.

To the best of our knowledge, the approach closest to the work at hand is Metronome [36], which claims to enable cross-blockchain transfers of Metronome tokens (MET). While the authors define a Proof of Exit, which can be used to claim tokens on the destination blockchain, no further technical details about this process are discussed. It remains unclear how Metronome tackles challenges like the XPP. In addition, Metronome only allows cross-blockchain transfers of MET tokens, while our approach allows arbitrary tokens to be transferred as long as the DeXTT protocol is used.

The DeXTT protocol presented in this paper is based on our own former work. The XPP and a precursor the DeXTT protocol have initially been discussed in [12]. Furthermore, in [21], we have conceptually described the deterministic witness selection approach. The work at hand significantly extends this former work by providing a concrete implementation of these approach within the DeXTT protocol, and using a single balance across all blockchains for each wallet, instead of individual per-blockchain balances.

While in our DeXTT prototype, we use smart contracts to implement the transactions defined in Section III, other methods exist. For instance, when considering the implementation on blockchains without smart contract support, one might add backwards-compatible layers on top of such blockchains, providing the required capabilities for the transactions presented in this work.

A similar approach is used by OmniLayer [4] or CounterParty [3], [37], which add such layers for enhanced features. For this work, however, we use our reference Solidity implementation of DeXTT for evaluation and cost analysis, postponing the integration of approaches such as OmniLayer or CounterParty to future work. Nevertheless, our current evaluation is sufficient to demonstrate the overall functionality of the DeXTT protocol using Solidity smart contracts and the conceptual applicability.

VI. CONCLUSION

In this paper, we have presented DeXTT, a protocol for transferring cross-blockchain tokens tradeable on multiple blockchains. This reduces dependency on a single blockchain and risk, e.g., of selecting a blockchain which later suffers from a security breach. DeXTT can be used for the exchange of any assets on any number of blockchains. This can be used for a cross-blockchain cryptocurrency, but also for other assets, possibly representing rights to real assets.

As a transfer protocol, DeXTT ensures eventual consistency of balances across blockchains, and prohibits double spending. We have presented the protocol, implemented it in Solidity, and provided an experimental evaluation, highlighting its performance with regard to time and cost.

Our evaluation shows that the reference implementation of DeXTT requires at least 4 blocks for maintaining eventual consistency. Furthermore, we show that a DeXTT transfer using our implementation costs 103.2 kGas for the receiver, and 81.5 kGas for any contributing observer. We also provide an analysis of the economic impact of the witness rewards based on the parameters of the multi-blockchain ecosystem used.

In future work, we will address the main limitation of our current evaluation by implementing DeXTT using additional technologies such as OmniLayer and therefore evaluate the performance of DeXTT in a blockchain ecosystem consisting of mixed blockchain types. Furthermore, we aim to implement DeXTT on other native smart contract platforms such as EOS.IO [38]. In addition, we aim to evaluate more refined approaches for the veto contest, which can be used to relax the currently strict requirements towards signed PoIs.

ACKNOWLEDGMENT

The authors would like to thank Christoph Ritzer for numerous fruitful discussions regarding the lemma of rooted blockchains.

REFERENCES

[1] A. Zohar, "Bitcoin: Under the hood," *Commun. ACM*, vol. 58, no. 9, pp. 104–113, 2015.
[2] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: May 31, 2019.[Online]. Available: https://bitcoin.org/bitcoin.pdf
[3] *Counterparty*. Accessed: May 31, 2019. [Online]. Available: https://counterparty.io/docs/
[4] J. Willett, M. Hidskes, D. Johnston, R. Gross, and M. Schneider. *Omni Protocol Specification*. Accessed: May 31, 2019. [Online]. Available: https://github.com/OmniLayer/spec
[5] *Litecoin*. Accessed: Feb. 15, 2019. [Online]. Available: https://litecoin.org/
[6] G. Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Accessed: May 31, 2019. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf
[7] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[8] T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the Use of blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32979–33001, 2018.
[9] H. D. Bandara, X. Xu, and I. Weber, "Patterns for Blockchain Migration," 2019, *arXiv:1906.00239*. [Online]. Available: https://arxiv.org/abs/1906.00239s
[10] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, Mar. 2017.
[11] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, to be published. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17318332
[12] M. Borkowski, C. Ritzer, D. McDonald, and S. Schulte. *Caught in Chains: Claim-First Transactions for Cross-Blockchain Asset Transfers*. Accessed: May 31, 2019. [Online]. Available: https://dsg.tuwien.ac.at/projects/tast/pub/tast-white-paper-2.pdf
[13] M. Borkowski, D. McDonald, C. Ritzer, and S. Schulte. *Towards Atomic Cross-Chain Token Transfers: State of the Art and Open Questions Within TAST*. Accessed: May 31, 2019. [Online]. Available: https://dsg.tuwien.ac.at/projects/tast/pub/tast-white-paper-1.pdf
[14] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. USENIX Annu. Tech. Conf.*, 2016, pp. 181–194.
[15] M. Beck. *Into the Ether With Ethereum Classic*. Accessed: May 31, 2019. [Online]. Available: https://ethereumclassic.github.io/assets/etc-thesis.pdf
[16] E. Duffield and D. Diaz. *Dash: A Privacy-Centric Crypto-Currency*. Accessed: May 31, 2019. [Online]. Available: https://github.com/dashpay/dash/wiki/Whitepaper
[17] *WAVES Whitepaper*. Accessed: Apr. 13, 2018. [Online]. Available: https://wesdewayne.files.wordpress.com/2017/05/waves-whitepaper.pdf
[18] M. Conoscenti, A. Vetró, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl.*, Dec. 2016, pp. 1–6.
[19] T. Hukkinen, "Reducing blockchain transaction costs in a distributed energy market application," M.S. Thesis, Dept. Comput. Sci., Aalto Univ., Helsinki, Finland, 2018.
[20] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.
[21] M. Borkowski, C. Ritzer, and S. Schulte. *Deterministic Witnesses for Claim-First Transactions*. Accessed: May 31, 2019. [Online]. Available: https://dsg.tuwien.ac.at/projects/tast/pub/tast-white-paper-3.pdf
[22] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.
[23] Y. Hirai, "Defining the ethereum virtual machine for interactive theorem provers," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2017, pp. 520–535.

[24] A. Gholipour and S. Mirzakuchaki, "A pseudorandom number generator with keccak hash function," *Int. J. Comput. Electr. Eng.*, vol. 3, no. 6, pp. 896–899, Dec. 2011.
[25] H. Gilbert and H. Handschuh, "Security analysis of SHA-256 and sisters," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, 2003, pp. 175–193.
[26] P. R. Berg and M. Milton. *Chronos: An Open Protocol for Streaming Money*. Accessed: May 31, 2019. [Online]. Available: http://chronosprotocol.org/chronos-white-paper.pdf
[27] C. Dannen, *Introducing Ethereum Solidity*. New York, NY, USA: Springer, 2017.
[28] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jun. 2017, pp. 557–564.
[29] J. Herrera-Joancomartí and C. Pérez-Solá, "Privacy in bitcoin transactions: New challenges from blockchain scalability solutions," in *Modeling Decisions for Artificial Intelligence*. New York, NY, USA: Springer, 2016, pp. 26–44.
[30] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Proc. Int. Workshop Open Problems Netw. Secur.*, 2015, pp. 112–125.
[31] C. Beams. *The Peer-to-Peer Bitcoin Exchange*. Accessed: May 31, 2019. [Online]. Available: https://github.com/bisq-network/bisq-docs/blob/master/exchange/whitepaper.adoc
[32] W. Warren and A. Bandeali. *0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain*. Accessed: May 31, 2019. [Online]. Available: https://0xproject.com/pdfs/0x_white_paper.pdf
[33] T. Zhang and L. Wang. *Republic Protocol: A Decentralized Dark Pool Exchange Providing Atomic Swaps for Ethereum-Based Assets and Bitcoin*. Accessed: May 31, 2019. [Online]. Available: https://releases.republicprotocol.com/whitepaper/1.0.0/whitepaper_1.0.0.pdf
[34] M. Herlihy, "Atomic cross-chain swaps," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2018, pp. 245–254.
[35] G. Wood. *PolkaDot: Vision for a Heterogeneous Multi-Chain Framework*. Accessed: May 31, 2019. [Online]. Available: https://polkadot.network/PolkaDotPaper.pdf
[36] *Metronome: Owner's Manual*. Accessed: May 31, 2019. [Online]. Available: https://www.metronome.io/pdf/owners_manual.pdf
[37] *Counterparty Protocol Specification*. Accessed: May 31, 2019. [Online]. Available: https://github.com/Counterparty XCP/Documentation/blob/master/Developers/protocol_specification.md
[38] *EOS.IO Technical White Paper v2*. Accessed: Apr. 8, 2019. [Online]. Available: https://github.com/EOSIO/Documen tation/blob/master/TechnicalWhitePaper.md

MICHAEL BORKOWSKI received the master's degree in software engineering and Internet computing from TU Wien, in 2015, where he is currently pursuing the Ph.D. degree with the Distributed Systems Group. He is currently a Research Scientist with the German Aerospace Center (DLR), Braunschweig, Germany. The core area of his research is using predictive technologies in the field of distributed systems, with a focus on machine learning. He is also contributing to the TAST Research Project.

MARTEN SIGWART received the master's degree in computer science from the Technische Universität Berlin, in 2018. He is currently pursuing the Ph.D. degree with the Distributed Systems Group, TU Wien, where he is also a Project Assistant. During his master's degree, he participated in the Erasmus Program at TU Wien. He is contributing to the TAST Research Project.

PHILIPP FRAUENTHALER received the master's degree in software engineering and Internet computing from TU Wien, in 2018, where he is currently pursuing the Ph.D. degree with the Distributed Systems Group. He is also a Project Assistant with the Distributed Systems Group, TU Wien. Before joining the Distributed Systems Group in February 2019, he worked for five years as a Software Engineer, developing enterprise software for insurances. He is also contributing to the TAST Research Project.

TANELI HUKKINEN is currently a Software Engineer with Bitpanda GmbH. He is also an Engineer in the blockchain field and is contributing to the TAST Research Project within Pantos as a Scientific Advisor and an Expert on blockchains.

STEFAN SCHULTE is currently an Associate Professor with the Faculty of Informatics, TU Wien. His research interests include the areas of cloud computing and the Internet of Things, and the application of blockchain technologies for data provenance and process enactment. Findings from his research have been published in more than 100 refereed scholarly publications.