Trustable Blockchain Interoperability: Securing Asset Transfers on Permissioned Blockchains

SUBMISSION DATE / POSTED DATE

25-04-2022 / 02-05-2022

CITATION

Pedreira, Catarina; Belchior, Rafael; Matos, Miguel; Vasconcelos, André (2022): Trustable Blockchain Interoperability: Securing Asset Transfers on Permissioned Blockchains. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.19651248.v1

DOI

10.36227/techrxiv.19651248.v1

Trustable Blockchain Interoperability: Securing Asset Transfers on Permissioned Blockchains

Catarina Pedreira
Departamento de Engenharia Informática
Instituto Superior Técnico
Portugal
catarina.pedreira@tecnico.ulisboa.pt

Rafael Belchior
Departamento de Engenharia Informática
Instituto Superior Técnico & INESC-ID
Portugal
rafael.belchior@tecnico.ulisboa.pt

Miguel Matos
Departamento de Engenharia Informática
Instituto Superior Técnico & INESC-ID
Portugal
miguel.marques.matos@tecnico.ulisboa.pt

André Vasconcelos
Departamento de Engenharia Informática
Instituto Superior Técnico & INESC-ID
Portugal
andre.vasconcelos@tecnico.ulisboa.pt

***Abstract*—Blockchains currently exist in silos, competing when they could be cooperating. Interoperability is essential to allow for communication between them and motivate mass adoption. In permissioned blockchains, interoperability is harder given their opaqueness. The solutions proposed so far to address interoperability require a trusted private third party, which may be insecure and is not ideal. We propose T-ODAP, a secure multi-layered protocol that enables a trustless solution for permissioned blockchain interoperability, eliminating the need for trust in the protocol's participants. It provides a Decentralized View Storage, a connector that connects permissioned blockchains to the latter, and a trustless version of the ODAP protocol. T-ODAP models the participants as rational agents using game theory techniques and is implemented using *Polkadot* and *Hyperledger Cactus*. We tested the implemented solution, evaluated the system's robustness in face of attacks, and concluded that the system is *(k,t)-weak-robust*.**

***Index Terms*—Blockchain, Permissioned, Security, Trustless**

I. Introduction

Blockchains are becoming more and more relevant in today's world as they have been proven to have the potential to redefine the digital economy [11]. In fact, many use cases besides cryptocurrencies have emerged for the technology over time. A blockchain is a distributed, immutable ledger that stores transactions, containing application dependent data. A blockchain that places restrictions on who its participants are, only allows them to perform certain actions and is controlled by a node or group of nodes is considered to be permissioned or private, while a blockchain that allows anyone to join and contribute to the network is permissionless (or public) [2].

Blockchains currently exist in silos - many blockchain projects encompass different characteristics and specialize in very distinct areas. Instead of cooperating, they often compete. In this context, organizations are able to choose from a wide range of options. However, this is a delicate choice - it is hard to learn the technology and expensive to invest in it [7]. Blockchain *interoperability* is of utmost importance, since it allows risk reduction by enabling migration across different blockchains. This way, once a blockchain becomes obsolete, it is possible to replace it. Additionally, *interoperability* enables the creation of new use cases, exploiting synergies between different solutions and scaling of existing ones [7], potentially fostering the technology's adoption.

Both permissioned and permissionless blockchains require blockchain *interoperability*. For permissionless blockchains, there are currently several solutions that provide interoperability. This is challenging to achieve, but still feasible due to the open nature of these systems. When it comes to permissioned blockchains, the situation is more challenging. These blockchains are opaque and thus it is against their nature to share the internal state with the outside world. This raises a challenge - in order to know the internal state of a permissioned blockchain, we have to take the word of at least one node belonging to the latter and the state we obtain might be incorrect if we are dealing with malicious nodes. Some *interoperability* solutions have also been arising for permissioned chains, yet most are centralized which may not be adequate due to the need to trust an extra entity.

Interoperability for permissioned blockchains enables relevant use cases, such as cross-border asset transfers between banks. These are, generally, still a very inconvenient form of payment given the high transaction fees, the lack of transparency and the high latency. In this scenario, with permissioned blockchain interoperability, each bank could be associated to a permissioned blockchain (given that a bank's data cannot be public) and be able to transfer assets from one blockchain to the other in a much faster, cheaper and secure way. Moreover, in this context, the interoperability mechanism should be trustless for a more secure solution - the less we have to place trust on centralized intermediaries, the better, given that we are dealing with sensitive information.

However, there are still not enterprise-grade solutions (this is, that have some steps implemented towards standardization) that enable interoperability across public and private blockchains.

ODAP (Open Digital Asset Protocol) [5], [9] is a cross-communication protocol that operates between two gateways to transfer assets between blockchains. This asset transfer is unidirectional and comparable to atomic swaps, where an asset is *locked* on one blockchain and its representation is created on another [6]. This solution is not trustless given that the gateways need to trust each other.

Therefore, a more decentralized, trustless and secure solution for permissioned blockchains' *interoperability* is needed. We address this with T-ODAP, a multi-layered secure solution for cross-chain asset transfers with a focus on permissioned blockchains. In the first layer, T-ODAP encompasses a trustless system that performs the publication of permissioned blockchain's internal state proofs in a DVS, implemented in Polkadot [3]. The second layer comprises a connector built in Hyperledger Cactus [10], that is compatible with several permissioned blockchains and can connect the latter to the DVS.The technical requirements for blockchains to be compatible with T-ODAP are: 1) have simple read and store functionaltiy, 2) have smart contract functionality that can use time, such that state (namely assets) can be locked. Finally, the third layer entails the use of the DVS and state proofs to build a more trustless and secure version of the ODAP protocol. In order to model the behavior of the protocol's participants, we used game theory techniques.

We evaluate T-ODAP's robustness in face of attacks and conclude that the system is *(k,t)-weak-robust*, similarly to mechanisms such as HTLC-based payment schemes or sidechain protocols [14]. We test the correct functioning of the first two layers of T-ODAP through Hyperledger Cactus, which enables blockchain and smart contract testing. We present the metrics we would have evaluated if we had had the opportunity, as well as expressing our predictions for the results to expect, in relation to ODAP as our baseline.

A. Work Objectives

The main goal of our work is to provide a secure and robust system that allows for trustless permissioned blockchain interoperability through the use of the DVS. The DVS is implemented in the form of a Polkadot smart contract and the connector is implemented in Hyperledger Cactus. The implementation of the theoretical model (i.e. the adaptation of the ODAP protocol) is intended for future work.

The following research questions are tackled by our solution:

1) How to guarantee the internal state proofs' correctness and integrity if permissioned blockchains are opaque?
2) How can we effectively model the dynamics of the protocol in regards to its rational participants, using game theory?
3) How to make T-ODAP strongly robust in terms of resilience to attacks?

B. Contributions

This work makes the following contributions:

1) A Decentralized View Storage built as a Polkadot smart contract, which allows for trustless state sharing between opaque blockchains, that can be leveraged for multiple use cases;
2) A public Polkadot Connector implemented in Hyperledger Cactus, capable of connecting several permissioned blockchains to Polkadot. Besides implementing the connector, we contributed to the open-source community of Hyperledger;
3) A theoretical model of a trustless adaptation of the ODAP protocol, T-ODAP, that leverages the DVS (and the connector in order to interact with the latter);
4) A game theory based analysis that demonstrates that T-ODAP is (k,t)-weak-robust.

The implementation of the Polkadot connector and the DVS smart contract can be found in https://github.com/CatarinaPedreira/cactus/tree/polkadot-connector-draft.

II. Related Work

In this section, we present the related work on blockchain technology, blockchain interoperability, and game theory to provide a better understanding of our protocol.

A. Blockchain Interoperability

Interoperability can be defined as "the ability of two or more software components to cooperate despite differences in language, interface, and execution platform" [12]. In the blockchain context, interoperability is a relatively new theme - interest from academia and industry did not start growing until about three years ago [7]. This type of interoperability emerged due to the desire to create new synergies between blockchains, thus creating new use cases. Due to the core differences between permissioned and permissionless blockchains, the interoperability problem is distinct for each of the types. Even within the same type, it can take many different forms due to the huge variety of existing blockchain infrastructures.

Several blockchain interoperability mechanisms exist. These can be divided in three different categories - Public Connectors, Blockchain of Blockchains and Hybrid Solutions, each of which is then divided into subcategories. The latter are described in [7].

Particularly important for our work, is a Blockchain of Blockchains named Polkadot [3], a system that (among other features) allows for interoperability between several blockchains, and which leverages its own internal, costumizable blockchains with a parallel nature - parachains (which are also interoperable between themselves). Hyperledger Cactus is also fundamental for our work - included in the trusted relay type (a hybrid solution). Cactus achieves interoperability between several blockchains (many of them permissioned) through the use of trusted nodes - Cactus nodes.

XCLAIM [13] is a framework that works to achieve blockchain interoperability in a trustless way (i.e. without the need of a centralized trusted third party), leveraging game theory techniques. It is mainly focused on permissionless blockchains, while in our work we focus on permissioned blockchains.

Additionally, a recent protocol [4] focuses on enabling an external party to observe and verify a permissioned blockchain's internal state, providing that there is at least one honest member present in the blockchain's committee. The state verification is achieved through the use of a secure public ledger that acts as a bulletin board - public bulletin - in which snapshots of the permissioned blockchain's state (named ”view” by the authors) are published at fixed intervals.

As mentioned above, ODAP is a cross-communication protocol that operates between two gateways to perform asset transfers between blockchains represented by those gateways. The latter is rather flexible, allowing blockchains of both types (both permissioned or permissionless) to transfer assets to each other. In ODAP (Open Digital Asset Protocol), the gateways are trusted, i.e. it is assumed that they will not drop an asset before a given transfer or that they will not transfer it to the wrong gateway.

III. T-ODAP: Decentralizing Asset Transfers

In this section, we introduce our approach, T-ODAP - Trustless Open Digital Asset Protocol, for the problem introduced in Section I.

The solution should provide a set of functional requirements. In T-ODAP, Cross-chain Transfers are performed by both the end-user (the Client), since this is the entity that triggers the T-ODAP protocol in the first place by having an asset to transfer, and also by the Source Gateway, which conducts the process of transferring the asset from one ledger to the other. Store Proofs corresponds to the storage of evidence of a blockchain's internal state (view). The Source and Recipient Ledger publish views of their own states in the DVS. Both Source and Recipient Gateways Verify Facts against the views published in the DVS. Depending on the results of this verification, their behavior is different. The Source Gateway triggers the action of a Rollback Asset Transfer in the Source Ledger in case it is verified that the Recipient Gateway did not follow the protocol. The same goes for the Recipient Gateway and Ledger in case the Source Gateway misbehaves. The ledgers do this by either unlocking the asset if they are the Source Ledger, or by not creating the corresponding asset if they are the Recipient Ledger. Finally, the Polkadot Connector is able to Deploy a Smart Contract (crucial given that the DVS is implemented as a Smart Contract) and is also able to Connect several Blockchains to Polkadot, enabling blockchains to connect to the DVS.

A. Assumptions

Similarly to related work [13], we assume that adversaries (in this case, the gateways) are computationally bounded and rational agents, motivated by actions that increase their utility (this is, maximizing profit while keeping their reputation) and avoiding actions that decrease their utility. Keeping reputation is important to gateways because they are identified and benefit from an open-market, that is driven by reputation and the law of demand.

As such, the latter can attempt to perform any attack that potentially maximizes their utility, such as not completing an asset transfer. In our context, we assume that a malicious node is any node which deviates from the established protocol T-ODAP.

In terms of the network, we assume that honest nodes are well-connected and there is a maximum delay in which they receive transaction broadcasts from users. When it comes to the DVS, we assume that each permissioned network comprises at least one auditor node (a member of that network) which validates conflicting views, deciding which ones are valid and which are not, in case of a dispute. Thus, we assume all views published in the DVS are valid (i.e. correspond to the correct internal state).

B. System Overview

Our approach is composed of several layers that stack on each other - the DVS in the bottom layer, then the Polkadot Connector and finally T-ODAP.

In the bottom layer of our protocol, we have the DVS. The DVS is based on the Public Bulletin for permissioned ledgers presented in Section II and is implemented as a smart contract in Polkadot. The state proofs (views) are publicly available for external clients to observe and verify facts against. In practice, they correspond to a digest of a permissioned blockchain's internal state. The necessity for a Public Bulletin or a DVS stems from the fact that permissioned blockchains are closed systems. In order to allow for a truly secure and trustless interoperability between permissioned blockchains (or between a permissioned and a permissionless chain), there needs to exist a system which securely shares the state of the latter for external observers. Similarly to the Public Bulletin, the DVS (Decentralized View Storage) is an immutable public bulletin where state proofs of a permissioned blockchain are regularly (i.e. every *k* blocks) published by its corresponding committee members.

It also considers a malicious but cautious committee, along with at least one honest member in the latter, which reports a conflict if it witnesses malicious behavior. However, there are some key differences. As stated by the authors of [4], it is possible that several valid external views exist for the same state. Differently from the aforementioned work, our algorithm encompasses a voting mechanism. A quorum of members vote on the view and the collective decision determines if that view is either valid or inconclusive, case in which a view conflict is reported which must be solved externally, by an auditor node. This auditor node exists in each permissioned network, and corresponds to a node which function is to decide if a given view is valid or not.

In our algorithm, even if all but one member voted positively on a view, one negative vote is sufficient to raise a conflict on that view (and vice-versa). Since there is at least one honest member in the committee, an invalid view will never be published (even if the honest member is not a voting member in that round, it can still report the view). As mentioned before, when a conflict is raised on a view, the view is deemed as either valid or invalid by the auditor node. The members who voted contrarily are recognized as being malicious and held accountable for their action. This alone can discourage a malicious node from trying to block the system indefinitely by constantly voting negatively, since reputation is rather important in a permissioned network (given that every entity is known). In future work, there can be additional punishment for malicious actors in the DVS. The DVS's publishing frequency should be adjusted depending on the blockchain leveraging it, i.e. for a blockchain in which blocks are frequently added to the chain, the number should be higher and vice-versa.

The Polkadot connector emerges on top of the DVS layer, as a bridge for permissioned blockchains to be able to access Polkadot. This is possible since the connector is part of Hyperledger Cactus (see Section II). Thus, a permissioned blockchain supported by Hyperledger Cactus can use the latter to access Polkadot through this connector. The connector also implements mechanisms to deploy smart contracts to the Polkadot network and to interact with them, by being able to call read and write function from those contracts. Since the DVS is implemented in the form of a smart contract and deployed in Polkadot, the Polkadot Connector is able to interact with the latter.

Our work's final layer arises as a trustless version of the existing protocol ODAP, leveraging the use of a DVS for permissioned blockchains' internal state sharing. This way, T-ODAP does not require that gateways trust each other since they can verify each other's state in the DVS, prior to any asset transfer occurring. T-ODAP is compatible with both permissionless and permissioned blockchains, however in this work we focused in the latter. This is because the DVS is necessary for proving the internal state of permissioned blockchains, but not needed for permissionless ones given that these are publicly verifiable.

The following actors exist in T-ODAP:

- *Source Ledger* B_S - The ledger that desires to transfer an asset to the recipient ledger, by locking *x* units from asset type *a* to be created in the latter;
- *Source Gateway* G_S - The gateway that transfers the locked *x* units from asset type *a* to the recipient gateway;
- *Decentralized View Storage (DVS)* - The immutable bulletin where a permissioned blockchain's internal state proofs are published regularly;
- *Recipient Gateway* G_R - The gateway that responds to G_S and is the target of the transfer;
- *Recipient Ledger* B_R - The ledger that receives the asset transfer, by creating the corresponding tokens in its ledger and making them available.

We previously saw that rational agents are motivated by actions that increase their utility and unmotivated by the actions that decrease it. In order to provide a secure protocol and motivate the players to choose desired actions (actions according with the specification of the protocol) instead of the contrary, T-ODAP punishes a gateway each time it chooses an undesired action. This punishment consists of decreasing that gateway's public reputation, making it less likely that it is chosen in the next T-ODAP instance. This mechanism works because each gateway is publicly identified (e.g., needs to be registered within a virtual asset service provider). The latter has a negative value associated to it, which will decrease the player's overall utility.

C. Protocol

We now discuss the design and architecture of the T-ODAP protocol. Figure 1, built with the Archimate language [1], illustrates the latter. In this figure, we can observe the several components forming T-ODAP's architecture, which are divided in four different groups for a better understanding.

The first group (on top) comprises the source ledger B_s, as well as an asset of type A and the source gateway G_s, which executes the asset transfer. The source gateway is a specialized type of Cactus Node, and it can be defined as ”a computer system in a blockchain network for the purpose of assisting in the movement of virtual assets into (out of) the blockchain network” [8]. The end-user is connected to G_s. The latter is the component which triggers the whole protocol, by issuing a CC-Tx asset transfer request. This request is associated with the transfer of *x* units of an asset of a given type *A* from B_s, which (if the protocol is successful) will be created as *y* units of an asset of given type *B* in the recipient ledger B_r. The second group is similar to the first one, however this one comprises, instead, a recipient ledger B_r, the corresponding recipient gateway G_r which interacts with G_s and which is the target of the transfer, and the resulting created *y* units of asset of type *B* in B_r. This group is not directly connected to an end-user. Both gateways interact with each other, being that G_s is the one that initiates the connection. Then, we can observe the third and fourth groups. The third group encompasses Hyperledger Cactus and its several connectors to blockchains/interoperability mechanisms (not all are represented), as well as its several Cactus Nodes. The fourth group comprises the Polkadot network and the DVS smart contract, deployed in it. In order to (indirectly) access the DVS, the gateways have to leverage the Polkadot Connector. Before the protocol instance begins, the connector connects to Polkadot and deploys the contract code containing the logic of the DVS. Then, G_s and G_r can use the connector to retrieve and read views from it, analyzing the state of B_r and B_s, respectively.

B_s and B_r are also connected to the DVS since, in order to guarantee the integrity of the views, the latter must be published by members of the blockchain itself and not by the gateways. If the gateways were able to publish views, since the latter can be malicious, we would not be able to be certain that the published views were always correct.

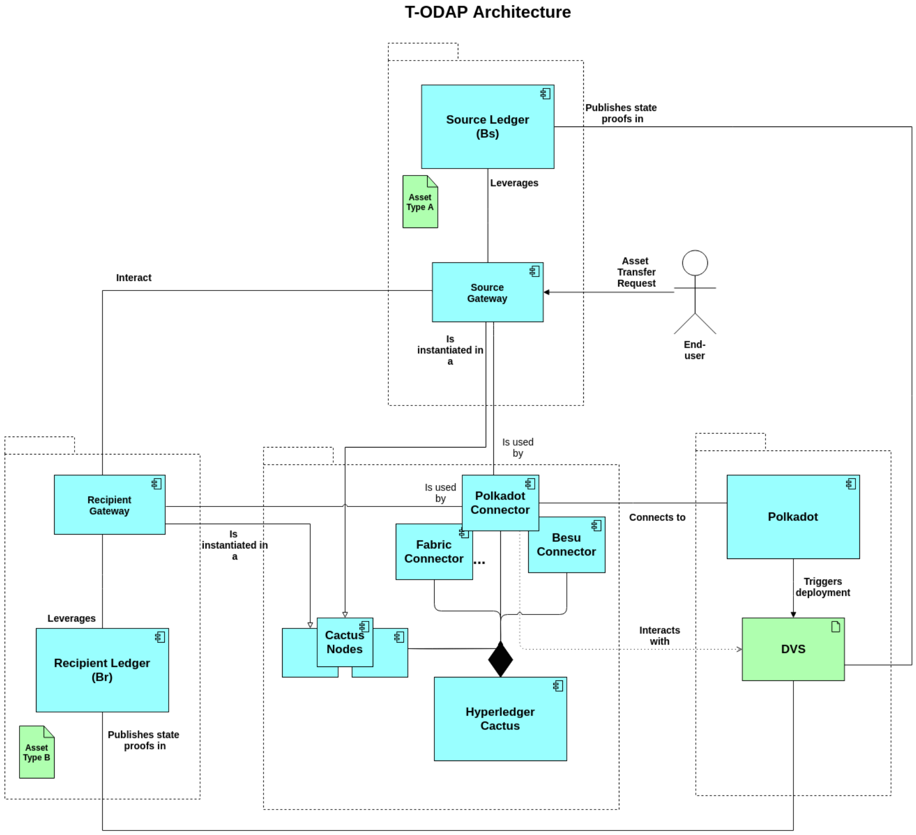

Fig. 1. Archimate T-ODAP Protocol Architecture

D. Protocol Flow

We will now discuss the protocol flow of T-ODAP.

Figure 2 illustrates an example of T-ODAP's protocol flow, having Fabric as the source ledger and Quorum as its recipient counterpart. Here, we can observe the main differences in relation to ODAP:

- DVS is a participant;
- Phase 3 - View Publication Flow - is introduced.

We can also observe that an if condition is introduced at the bottom, which depends on the outcome of the last step of Phase 3.

First, an end-user (i.e. an application) issues a CC-Tx asset transfer request through Cactus, which triggers the beginning of the protocol. Then, Phase 1 (Transfer Initiation Flow) and Phase 2 (Lock-Evidence Verification Flow) take place. The first phase leverages the initiation processes, necessary for connection between the gateways and the second phase takes care of locking the asset, along with verifying that the recipient ledger is indeed interested in receiving the asset transfer. Then, Phase 3 begins. Here, we begin with Hyperledger Fabric (B_s) publishing a view at a given time t (note that views are frequently published, with the value *k* depending on the source blockchain). This step is particularly important since it shares the internal state of Fabric at that moment in time, and since this view contains information about the state of the asset to be transferred. The protocol proceeds with G_r retrieving Fabric's most recent published view, in order to be able to analyze its contents and confirm that the asset is indeed in a blocked state. Note that, in this stage, G_r only retrieves the view after a given time *t* has passed. This amount of time depends on the blockchain B_s. This is due to the fact that even if the asset is locked, the view containing this information might only be published after some time, or the network can have some delay. To guarantee that the retrieved view contains the correct and most recent information about the lock, we wait *t* units of time. The outcome of this verification triggers one of two options within the protocol:

- If the asset is indeed locked, Phase 4 (Commitment Establishment Flow) takes place. This phase comprises a preparation commit, a final lock (by B_s) and a final commit of the transfer, containing all the information necessary for B_r to create the asset.

  After G_r claims that the asset was created in Quorum, Phase 5 (Asset Creation Verification Flow) starts. Here, G_s will retrieve Quorum's most recent published view (again, waiting *t* units of time before doing so) and verify if the information provided by G_r is correct. In case it is, the transfer process finishes with success, having the asset in its final state - digital twin asset. Otherwise, G_r attempted to execute an attack by not creating the asset in the Quorum blockchain. The transfer is rolled back and the blocked asset in B_s is set to a pure state again, so

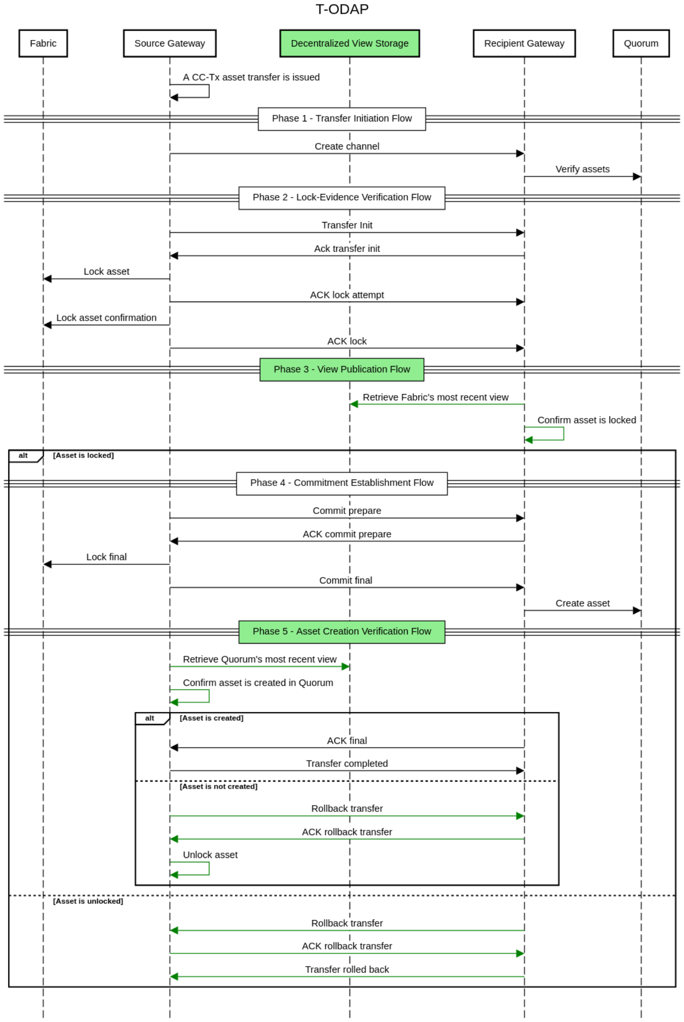

Fig. 2. T-ODAP Protocol Flow example

that it is not lost.

- If the asset is unlocked, this means that G_s opted for an undesired action. The transfer has to be rolled back; otherwise, by creating a representation of the asset in B_s, double spend would occur.

E. Threat Model

We now present the threat model and security analysis for T-ODAP. Note that each threat corresponds to an action that can be performed by a malicious node. There are several threats included:

Threat 1 - *The source gateway G_s steals the asset to be transferred (it does not lock the asset before transferring it to G_r).*

Let us imagine G_s is meant to transfer an asset to G_r, so that the latter creates the asset's representation in B_r. G_s can try to steal the asset by not providing instructions for B_s to lock the asset, while lying to G_r about locking it. This way, B_r will still end up creating the asset's representation although the asset has not been locked in the source ledger.

T-ODAP mitigates this attack through the use of the Decentralized View Storage. As we have previously seen in the protocol's flow, the latter allows for the removal of trust between the gateways, since the recipient gateway G_r can observe the source ledger's internal state (including the asset's state) prior to the transfer, so that it can stop the latter in case the asset's is incorrect.

Threat 2 - *The source gateway G_s steals part of the asset to be transferred but transfers the remaining portion.*

This threat is a slight variation of the previous. In this context, imagine the asset transfer comprises transferring 5 units of token of type A to be created as x units of token of type B in G_r. The source gateway can try to lock only 3 of those units and steal the remaining 2. The transfer will still take place, since G_r believes that B_s locked the entire asset.

T-ODAP mitigates this threat as it mitigates threat 1 - the recipient gateway can verify B_s the exact amount of token units that must be locked. If this number does not match what is expected, the transfer is rolled back.

Threat 3 - *The recipient gateway G_r does not create the assets in the recipient ledger.*

Here, the threat is focused on the recipient gateway, which performs a denial-of-service attack by not creating the assets in B_r. The latter can be executed by a malicious G_r that desires to harm the users of the source gateway, the source gateway or both by causing them to lock funds that will never be created in B_r. Despite not having a monetary incentive (given that the assets are not created), the malicious intent towards the participants can suffice as an incentive for the attack (i.e. the valuation value is high for this attacker).

T-ODAP mitigates threat 3 through the fifth phase of the protocol (see Figure 2), in which the internal state of the recipient ledger is verified after G_r claims that the assets were created. In case the gateway is malicious and the assets are not created, the transfer suffers a rollback and the asset's state in B_s goes back to pure state.

The attacks described in threat 1, threat 2 and threat 3 can still be successfully executed during the attack windows - i.e. during the intervals between view publications, since during the latter the attack is not registered and thus can not be proven to have happened. In order to diminish the attack window as much as possible, the view publishing frequency should be high (i.e. k should be low). However, this is a trade-off - highly frequent publications incur higher costs.

IV. Evaluation

In this section, we evaluate T-ODAP both from a theoretical perspective. In order to do this, we leverage the game theoretical framework in [14].

The latter evaluates a blockchain protocol's robustness by first identifying the players involved, the actions they can perform (tied with specific utilities) and the game or games that better represent that protocol.

In T-ODAP, we have two players - the source gateway G_s, and the recipient gateway G_r. These are considered to be rational players, meaning that they both always desire to maximize their own utility. Based on the protocol flow of T-ODAP, we divided the protocol into three different games, the first (A) corresponding to Phases 1, 2 and 3, the second (B) corresponding to the scenario where the asset is locked and the third (C) to the remaining scenario.

In each game, the order of the actions performed matters. In this context, if any player deviates the protocol, the game goes back to the initial state, with a null outcome (0) for each (i.e. (0,0), where the first position corresponds to G_s and the second one to G_r). The initial state corresponds to the state before the asset transfer. If they follow each step correctly, they receive a positive utility of (1,1). If a player is harmed by another player's action, the harmed player receives a negative utility of -1, similarly to authors in [14]. The values of 1 and -1 were chosen by convention.

Figure 3, Figure4 and Figure 5 correspond to Game A, Game B and Game C, respectively, and are presented below.

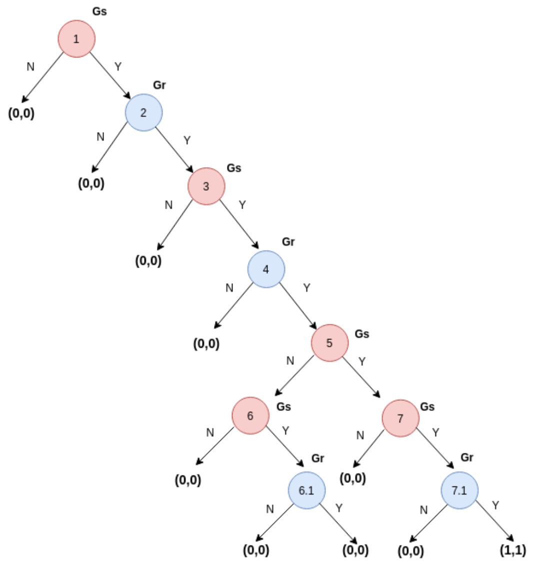

Fig. 3. Game A - Diagram

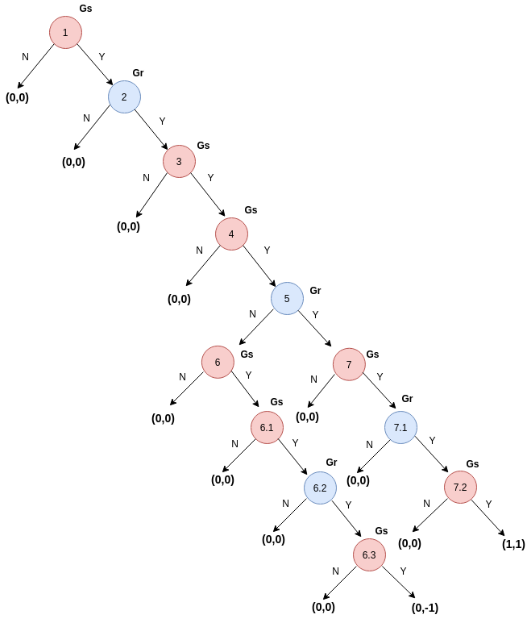

Fig. 4. Game B - Diagram

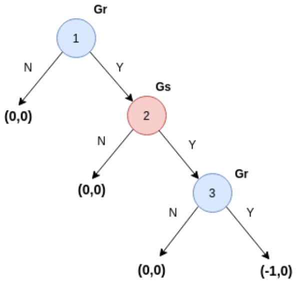

Fig. 5. Game C - Diagram

An instance of T-ODAP can be composed by Game A and Game B forming mechanism AB or by Game A and Game C, forming mechanism AC. In the latter case, the outcome will never be the best outcome possible for any of the players - it will either be 0 or -1. In Game A, if the players reach the outcome (1,1), the next game will be Game B. Else, the next game will be Game C. This means that the mechanism AB is the only one which can lead to an optimal outcome (i.e. if both players follow the protocol in every step). Therefore, mechanism AB is the one relevant to decide if T-ODAP is resilient to attacks given the fact that, out of the two, AB is the only one that presents a possibility of players following the protocol from the beginning to the end.

Through the analysis of the AB mechanism, and through the theorems presented in [14] - more specifically, due to the fact that the mechanism is both optimal-resilient (no subset of players has an incentive to deviate from the protocol) and t-weak-immune (for any altruistic player, the lowest possible outcome is the initial state's outcome, even if other players misbehave), we can conclude that T-ODAP is (k,t)-weak-robust.

In a simplified manner, when analyzing game A and game B, we can see that in both cases both players are incentivized to follow the protocol given that the best possible outcome is (1,1). This means that the players have no incentive to deviate from the protocol, which makes both games optimal-resilient. Moreover, the worst possible utility for an honest player is the initial state's utility - (0,0). Only deviant players can receive negative utility when they misbehave (in Game B). Therefore, the games are also t-weak-immune [14]. Since both games are optimal resilient and weak immune, by applying the Theorems 2, 3 and 4 of [14] which ensure the invariance of properties once the composition operator is applied, we get that the resulting mechanism AB is both optimal resilient and weak immune, which makes it (k,t)-weak-robust, the same level of robustness as a cross-chain swap protocol.

This means that the honest players can trust that they will never lose utility by taking part in the protocol. Their utility will either remain the same, if there is a malicious player in the game, or be increased if both players follow the protocol. Players are incentivized to cooperate given that that gives them the best possible outcome, and are at the same time discouraged from misbehaving given that, if they do, their utility will either stay the same or be reduced.

V. Limitations and Future Work

T-ODAP has some limitations, the first being that it is more costly than ODAP. This makes sense since T-ODAP adds complexity to the latter, as well as several transactions in blockchains, which causes this higher cost. However, as stated before, the biggest focus of T-ODAP is in a trustless and secure solution, so the higher cost comes as a trade-off.

Additionally, in future work, other features may be added to T-ODAP such as the support of slashing to punish participants that deviate the protocol. This can correspond, for example, to the use of a collateral ([7], [13]), which is removed in case the participants misbehave. This mechanism involves, however, some challenges: how to assure that an internal state proof is actually invalid, in order to punish a participant fairly? The latter is hard to achieve due to the opaqueness of permissioned blockchains.

It is also interesting to leverage a crash-recovery mechanism [5] for the T-ODAP gateways, given that one of them can crash in the middle of an asset transfer and it is not desirable to have to rollback that transfer every time this happens. [6] presents a first approach to this problem, however it is not implemented yet.

VI. Conclusion

Blockchain interoperability is essential for mass adoption. Some projects have been studying and presenting ways to achieve this, however many require a centralized trusted third party, which is not desirable.

This paper presents T-ODAP, a multi-layered secure and trustless system leveraging a DVS to publish internal state proofs, a Polkadot Connector to interact with the latter and a trustless adaptation of the ODAP protocol. T-ODAP has the goal to arise as an alternative to the centralized interoperability solutions currently offered to permissioned blockchains, providing stronger levels of security in relation to other protocols such as ODAP due to being trustless.

We evaluated the full system's robustness in face of attacks and concluded that the system is (k,t)-weak-robust, similarly to popular mechanisms such as HTLC-based payment schemes. We performed tests to the implemented layers of our solution, which were successful. The latter were realized through Hyperledger Cactus, which enables blockchain and smart contract testing.

Our solution contributes to the development of permissioned blockchain interoperability, which in turn will hopefully contribute to the widespread adoption of blockchain technology in enterprises.

References

[1] Archi – open source archimate modelling.

[2] On public and private blockchains — ethereum foundation blog.

[3] Polkadot: Vision for a heterogeneous multi-chain framework.

[4] E. Abebe, Y. Hu, A. Irvin, D. Karunamoorthy, V. Pandit, V. Ramakrishna, and J. Yu. Verifiable observation of permissioned ledgers. *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2021.

[5] R. Belchior, M. Correia, and T. Hardjono. DLT Gateway Crash Recovery Mechanism draft 02. Internet-Draft draft-belchior-gateway-recovery-02, Internet Engineering Task Force, 2021.

[6] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono. HERMES: Fault-Tolerant Middleware for Blockchain Interoperability. *Future Generation Computer Systems*, mar 2021.

[7] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. A survey on blockchain interoperability: Past, present, and future trends. *ACM Comput. Surv.*, 54(8), Oct. 2021.

[8] T. Hardjono. Blockchain gateways, bridges and delegated hash-locks. *ArXiv*, abs/2102.03933, 2021.

[9] M. Hargreaves, T. Hardjono, and R. Belchior. Open Digital Asset Protocol draft 02. Internet-Draft draft-hargreaves-odap-02, Internet Engineering Task Force, 2021.

[10] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior. Hyperledger Cactus Whitepaper, 2020.

[11] S. Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59:15–17, 10 2016.

[12] P. Wegner. Interoperability. *ACM Computing Surveys*, 28:285–287, 3 1996.

[13] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 193–210, 2019.

[14] P. Zappalà, M. Belotti, M. Potop-Butucaru, and S. Secci. Game Theoretical Framework for Analyzing Blockchains Robustness. In S. Gilbert, editor, *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.