Trustable Blockchain Interoperability: Securing Asset Transfers involving Permissioned Blockchains

Catarina Pedreira

Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—The blockchain technology has been drawing a great deal of attention since its arrival in 2008, with Bitcoin. It does not come as a surprise that this technology has an enormous potential to be applied in a vast number of areas. However, currently blockchains exist in silos, often competing when they could be cooperating and communicating. Interoperability is essential to allow for communication between blockchains and thus motivate mass adoption. In permissioned blockchains, interoperability is harder given their opaque nature. Some solutions have already been created in this context, however many require a trusted private third party, which may be insecure and it is not ideal given the nature of the technology. In this paper, we propose T-ODAP (Trustless Open Digital Asset Protocol), a secure multi-layered protocol that enables a trustless solution for permissioned blockchain interoperability. T-ODAP is more secure than centralized solutions given that it eliminates the need for trust in the protocol's participants. It provides a Decentralized View Storage (DVS), a Polkadot Connector that connects permissioned blockchains to the latter, and a trustless version of the Open Digital Asset Protocol which leverages the DVS and the connector. The protocol models the participants as rational agents and implements game theory techniques in order to punish them in case they deviate the protocol. T-ODAP is implemented using *Polkadot* and *Hyperledger Cactus*. We tested that the implemented solution worked properly. In the theoretical evaluation, we were able to evaluate the system's robustness and concluded that the system is resilient to attacks, having the same robustness level as Hashed Time-Lock Contract (HTLC)-based payment schemes.**

I. Introduction

Blockchains are becoming more and more relevant in today's world as they have been proven to have the potential to revolutionize applications and redefine the digital economy [13], and 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) [3].

Blockchains currently exist in silos - there are a large number of blockchain projects that encompass different characteristics and that 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 [9]. 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 [9], potentially fostering the technology's adoption.

Both permissioned and permissionless blockchains require blockchain *interoperability*, and interoperability looks different for each of the types. For permissionless blockchains, there are several solutions that provide interoperability while still maintaining the decentralized aspect that the technology defends, such as XCLAIM [15]. This is challenging to achieve, but still feasible due to the open nature of these chains. When it comes to permissioned blockchains, the situation is more challenging. These blockchains are opaque and thus it is against their nature to share internal states with the external world. This is challenging to solve - in order to know the internal state of a blockchain of this type, 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 completely secure.

This type of interoperability is very relevant given the fact that it enables new use cases leveraging permissioned blockchains, such as cross-border asset transfers between banks. These are, in general, still a very inconvenient form of payment given the high transaction fees, the lack of transparency and the high latency (usually around 2-3 days). In this scenario, with permissioned blockchain interoperability, each bank could be associated to a permissioned blockchain (given that a bank's data can not 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.

ODAP (Open Digital Asset Protocol) is a cross-communication protocol that operates between two gateway devices to transfer assets between blockchains represented by those gateways. This asset transfer is unidirectional and comparable to atomic swaps, where an asset is *locked* on one blockchain and it's representation is created on another [8].

A more decentralized, trustless and secure solution for permissioned blockchains' *interoperability* is needed. Thus, we propose 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 [4]. The second

layer comprises a connector built in Hyperledger Cactus [2], that is compatible with several permissioned blockchains and can connect the latter to the DVS. Hyperledger Cactus and Polkadot are interoperability mechanisms introduced in Section II. 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. Please note that the third layer consists of a theoretical model and was not implemented yet due to circumstances outside of our control; the implementation is intended for future work.

T-ODAP's biggest focus is providing a trustless, more secure solution than others that currently exist for permissioned blockchain interoperability. This entails a cost and complexity trade-off, which we are willing to accept as long as our work's objectives are met.

Finally, we present a theoretical evaluation for T-ODAP. We evaluate the full system'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 [16].

We tested the correct functioning of the first two layers of T-ODAP through Hyperledger Cactus, which also enables blockchain and smart contract testing. We also presented 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 should be 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?

II. Related Work

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

A. Introduction to Blockchain

A blockchain can be described as a decentralized, tamper-proof distributed ledger [11], that allows trusted transactions among untrusted participants [12]. A *transaction* is proposed by a user (blockchain participant) and it is an essential component of the blockchain. The data it contains depends on the blockchain's scope - e.g. if it is financial, among other data, the transaction contains the value in concern and the addresses of the sender and receiver. More broadly, the transaction can also be called a *record*. A blockchain stores different sets of transactions into *blocks*, where each block is connected to others forming a chain (hence its name). The blockchain nodes form and maintain the network's infrastructure. Each node communicates with other nodes and contains a local replica of the chain - when a block is verified, each node attaches it to its local replica. This replica is usually the same on all nodes, although on some blockchains there may be temporary or even permanent distinct local replicas. This can either be a consequence of the nature of the blockchain (for example, due to a probabilistic consensus algorithm) or derived from a need to protect privacy - in a private blockchain, it is desirable for participants to be able to hide certain parts of the state they hold [7]. Though this may be the case for local replicas, the *global state* (the set of states that compose the blockchain) remains consistent. The network nodes reach *consensus* when they agree on a global state for the blockchain. There are several algorithms used to reach consensus, some more complex than others, depending on the type of blockchain - permissioned blockchains utilize Byzantine Fault Tolerant protocols (BFT) as *consensus* mechanisms while permissionless blockchains need more complex *consensus* algorithms such as Proof-of-Work, which consists in solving a hard and computationally expensive cryptographic puzzle [11].

B. 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" [14]. In the blockchain context, interoperability is a relatively new theme - interest from academia and industry did not start growing until about three years ago [9]. 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.

There are several existing blockchain interoperability mechanisms. These can be divided in different categories, which we illustrate in Figure 1, based in [9].

Particularly important for our work, is a Blockchain of Blockchains named **Polkadot** [4], a software 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 [15] 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

Category	Sub-category	Main use case
Public Connectors	Sidechains	Scalability, asset exchange
	Notary Schemes	Cryptocurrency exchanges
	Hashed timelocks	Cryptocurrency trading
	Hybrid	Enabling cross-chain assets
Blockchain of Blockchains	–	Creation of customized blockchains
Hybrid Solutions	Trusted Relays	Efficient interoperation
	Blockchain-Agnostic	General protocols
	Blockchain of Blockchains	Cross-blockchain dApps
	Blockchain Migrators	Risk reduction

Fig. 1: Blockchain Interoperability Solutions (adapted from [9])

techniques. This work cannot resolve this paper's problem since it only supports a limited type of permissioned blockchains. Moreover, the use of a chain relay to provide external data from a permissioned blockchain might not be secure - as these blockchains are opaque, one can not know if the information directly provided from the inside is trustworthy. However, the game theory techniques may be useful for our work.

Additionally, this recent protocol [6] 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. This is extremely useful to our work given that it provides a more decentralized solution for permissioned blockchains' interoperability.

Finally, another very relevant work is ODAP, mentioned previously. To recall, it 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. This corresponds to its most significant disadvantage, since it might not be very secure to assume this. Moreover, it is not able to verify permissioned blockchain's internal state, which might not be secure for the same reasons presented when talking about the disadvantages of XCLAIM.

C. Game Theory

Game theory has the goal of modeling a strategic interaction between different players in a context with predefined outcomes [10] - a *game*. A player's *utility* reflects the outcome expected by that player, acting as way to quantify its preferences. A strategy is a set of actions that the player can choose from, where the ultimate goal is to achieve its expected outcome. It is also based in the other players' strategies. If a player is rational, it has clear preferences, meaning it always desires to maximize its own utility by choosing the strategy that comprises the optimal expected outcome.

Game theory can be applied to the most significant challenges in the Blockchain technology, such as security problems, mining management, economical issues with the technology and energy trading [11].

III. T-ODAP: Decentralizing Asset Transfers

In this section, we propose a solution for the problem introduced in Section I - T-ODAP (Trustless Open Digital Asset Protocol).

A. Requirements

T-ODAP must provide a secure and trustless protocol that enables asset transfers via gateways.

Several non-functional requirements are desired:

- *Security* - The protocol should ensure that the processing and delivery of the assets is secure, namely assuring that gateways follow the protocol and do not tamper, drop or re-direct the assets to wrong gateways. In order to do this, the protocol enforces the use of the DVS.
- *Compatibility* - T-ODAP should be compatible with several permissioned (and permissionless) blockchains that support smart contracts with functionality for locking and unlocking assets.
- *Trustlessness* - The protocol should provide a trustless solution, i.e. a solution in which a participant does not need to trust any other participants in the system in order to maintain security of its assets and other expectations of functionality, only needing to trust the protocol, mathematics, cryptography, code and economics [5]. In our case, this means that the two gateways in any protocol instance do not need to trust each other in order to perform an asset transfer;
- *Availability* - T-ODAP should be working in proper conditions, with a minimal downtime;
- *Testability* - It needs to be possible to test the system in a safe environment (e.g. a non-production environment emulating a high workload);
- *Privacy* - The protocol should only be able to provide state proofs about the internal state of each participating gateway, and not any other that is not involved.
- *Efficiency* - The solution should be efficient, i.e. despite encompassing additional steps in relation to ODAP, these additional steps should not affect performance severely.

The solution should also tackle functional requirements. These are illustrated in Figure 2.

B. Assumptions

Similarly to related work [15], we assume that adversaries (in this case, the gateways) are computationally bounded and rational agents, motivated by actions that increase their utility and avoiding actions that decrease their utility. 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

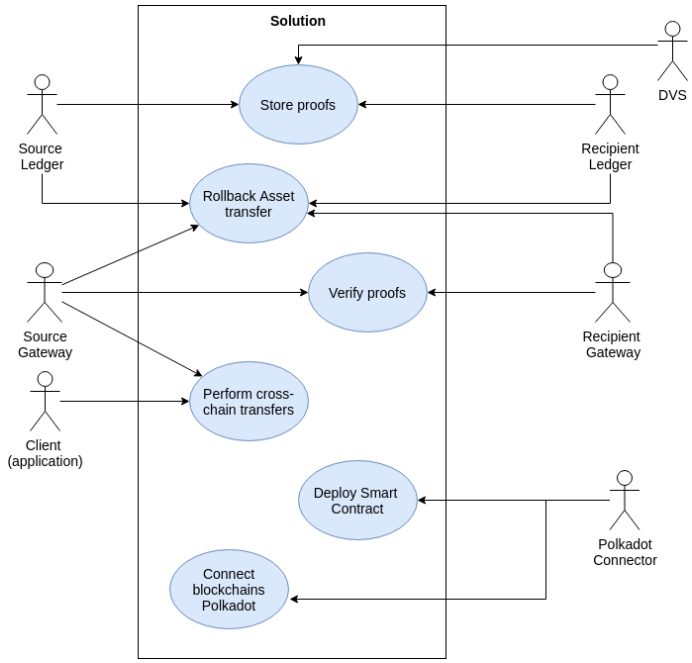

Fig. 2: Functional Requirements T-ODAP

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. Thus, we assume all views published in the DVS are valid (i.e. correspond to the correct internal state).

C. System Overview

Our solution is composed of several layers that stack on each other - the DVS, the Polkadot Connector and T-ODAP. This scheme is depicted in Figure 3.

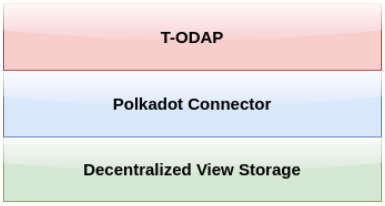

Fig. 3: Layers that build T-ODAP

In the bottom layer of Figure 3, we have the DVS. The latter 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 [6], 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. We understand that the existence of this validator node can be seen as contradictory having our goal of trustlessness in mind, however it was the solution we found within our limited time. This is an aspect that can be improved in future work. 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). 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.

Finally, 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 it is 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. The latter has a negative value associated to it, which will decrease the player's overall utility.

D. Protocol

We now discuss the design and architecture of T-ODAP protocol. Figure 4, 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.

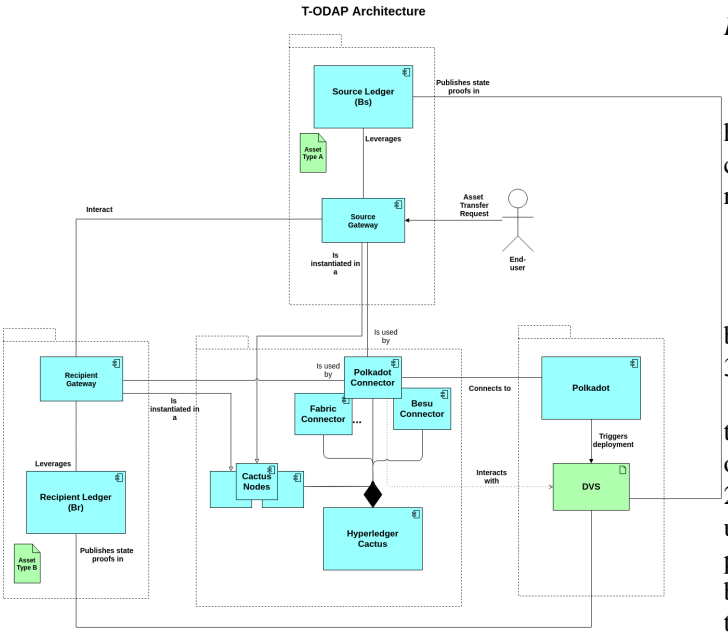

Fig. 4: Archimate T-ODAP Protocol Architecture

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" [**?**]. 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.

E. Protocol Flow

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

Figure 5 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 2 (Lock-Evidence Verification Flow) take place; these remain unchanged from what was presented in Section II - the first phase leverages 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

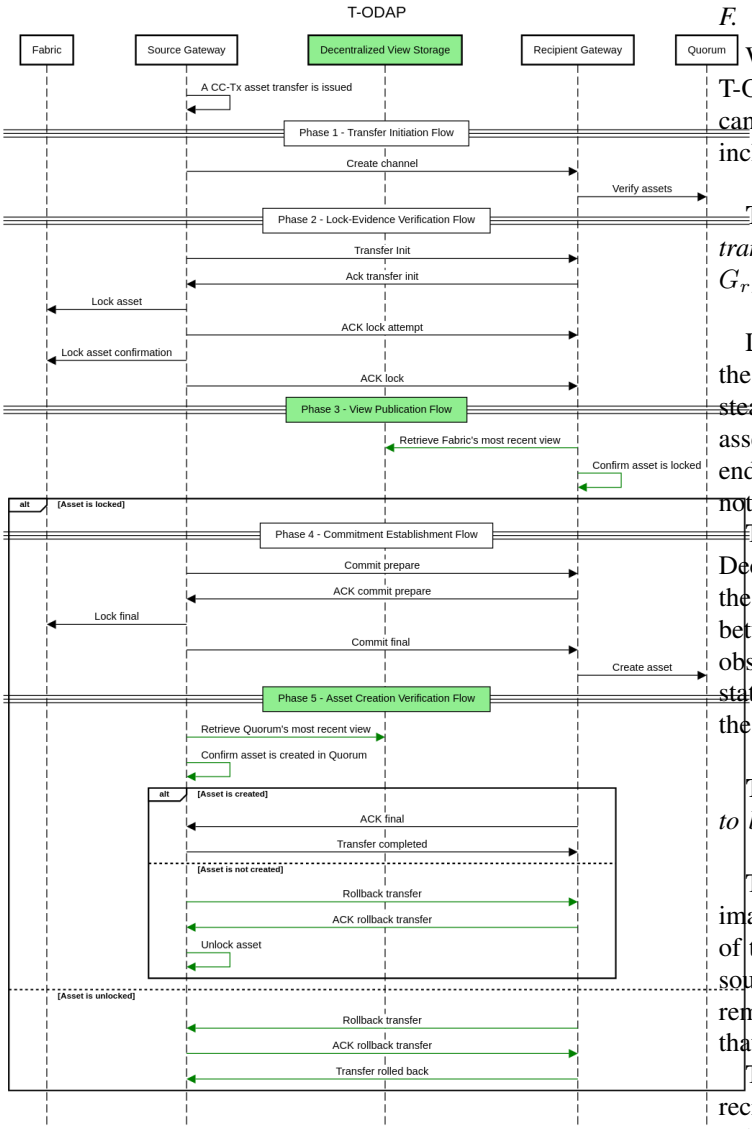

Fig. 5: T-ODAP Protocol Flow example

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 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.

F. 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 5), 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, 2 and 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 high costs for the solution.

IV. Implementation

In this section, we present implementation details of the solution.

Only the first two layers of T-ODAP were implemented - we were not able to implement the third layer due to circumstances outside of our control. The DVS layer was implemented as a Polkadot smart contract, written in the Parity language ink!, based on Rust, while the Polkadot connector was implemented as a new connector on Hyperledger Cactus.

The implementation of these layers contributed to both open-source projects of Polkadot and Hyperledger Cactus.

V. Evaluation

In this section, we evaluate T-ODAP, being divided into two sections: theoretical and practical evaluation. Note that, as we were not able to implement the last layer of the solution, the practical evaluation could not be completed. We did, however, perform several tests to both the DVS and the Polkadot connector's implementation, in order to guarantee the correct functioning of the latter.

A. Theoretical Evaluation

In order to perform a theoretical evaluation to T-ODAP, we leverage the game theoretical framework in [16].

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 do 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 [16]. The values of 1 and -1 were chosen by convention.

Figures 6,7 and 8 correspond to Game A, Game B and Game C, respectively, and are presented below.

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

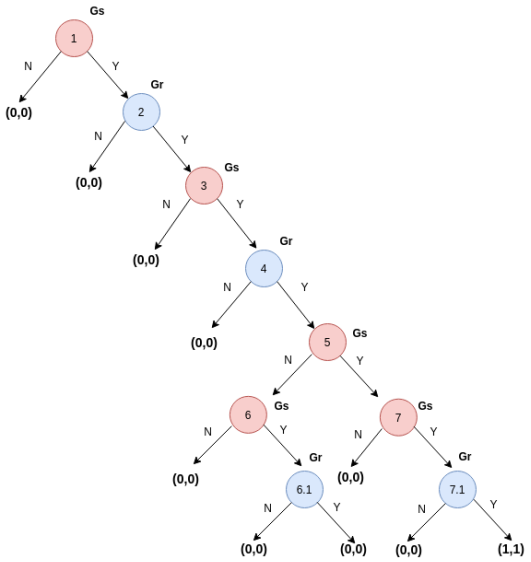

Fig. 6: Game A - Diagram

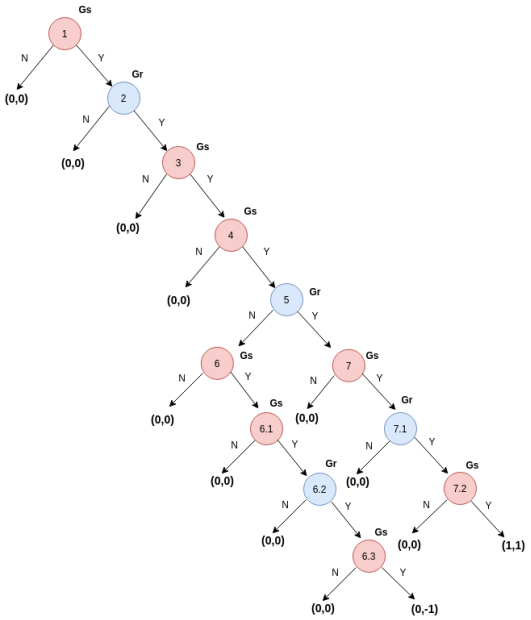

Fig. 7: Game B - Diagram

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 (if both players follow the protocol in every step).

Through the analysis of the AB mechanism, and through the theorems presented in [16], we conclude that T-ODAP is (k,t)-weak-robust - the same level of robustness as a cross-chain swap protocol.

B. Practical Evaluation

Since we did not implement all the layers of T-ODAP (due to depending on the implementation of ODAP to be finished, and due to the fact that it got delayed), the practical evaluation could not be completed. These are the metrics we planned to evaluate (some of which would be evaluated in relation to the work's baseline, ODAP):

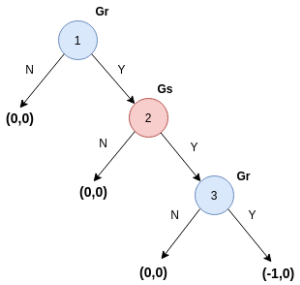

Fig. 8: Game C - Diagram

- *Throughput* - The solution must be able to deal with several cross-chain transaction requests for asset transfers without causing great harm in latency and performance. ion to ODAP. As T-ODAP is built on top of ODAP and adds extra steps (and thus, more complexity), it is certain that our solution will be somewhat slower than the baseline;
- *Latency* - We would test T-ODAP against distinct workloads of asset transfer request rates. Similarly to before, T-ODAP is expected to have higher latency when compare to its baseline;
- *Cost* - The monetary cost of running an instance of the protocol, in terms of transaction fees. It is expected that the cost of T-ODAP would be slightly higher given that there are extra transactions involved.
- *DVS Publishing Frequency* - We would test the solution against different publishing frequencies and observe its behavior in terms of robustness to attacks, trying to understand what is the ideal value or range of values for k of publishing frequency in terms of a having a small attack window, as well as not having a great loss in performance at the same time.

We understand that there is a trade-off between performance and security yet we choose to give more weight the trustlessness and security of our solution.

VI. Conclusion

Blockchain interoperability is essential for mass adoption. This applies to both permissioned and permissionless blockchains, however we focused on the first due to the small amount of solutions that currently exists. Some projects have been studying and presenting ways to achieve this, however many require a centralized trusted third party, which may not be completely secure. By trusting individual entities, these systems are vulnerable to attacks such as incorrect state sharing by malicious members of permissioned chains or by a malicious centralized third-party. Therefore, there is an opportunity for a new, trustless solution which makes the process more secure. The latter ensures that trust is moved from one entity to a protocol; this provides a higher level of security given that the protocol itself is secure.

We explored and analyzed several state-of-the-art solutions, including XCLAIM, ODAP and Public Bulletin. The aforementioned studies comprise highly valuable aspects which can be applied to a solution for permissioned blockchain interoperability. This includes mechanisms based on game theory that incentivize the participants to follow the established protocol, a mechanism to publish internal state proofs and a mechanism to perform unilateral asset transfers between gateways. All were extremely valuable for the construction of the solution, however none of them encompasses all the desired characteristics for our solution - e.g. XCLAIM uses possibly insecure chain relays and ODAP requires gateways to trust each other which can be insecure.

We presented 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.

Theoretically, 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. Unfortunately, we were not able to perform an experimental evaluation given the fact that the third layer of T-ODAP was not implemented. However, we presented information about the tests we did to the first two layers of our solution, which indicated the correct functioning of the system. Finally, we presented the metrics we would have evaluated if we had had the opportunity, as well as expressing our predictions for the results to expect, relatively to ODAP, which is our work's baseline.

Our solution contributes to the development of permissioned blockchain interoperability, which in turn will hopefully contribute to the widespread adoption of permissioned blockchains in enterprises, affecting the latter and society as a whole.

A. Contributions

We were able to deliver several contributions to the scientific community and the open-source community, that have value not only together but also individually.

As primary contributions, we delivered:

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 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.

Note that the implementation of the Polkadot connector and the DVS smart contract can be found in https://github.com/hyperledger/cactus/pull/1490.

VII. 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 steps and more 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.

As mentioned before, the third layer of T-ODAP was not implemented. The implementation was not possible on time due to circumstances outside of our control; however, the layer is ready for implementation in future work. This work can be done through Hyperledger Cactus, which has open-source code, where ODAP is implemented as well. The adaptation consists on connecting ODAP to the DVS through the Polkadot Connector, and then adjusting the protocol to verify the asset's state before and after an asset transfer (as described in the solution's Flow).

Additionally, in future work, other features may be added to T-ODAP such as the support of slashing to punish participants in case they deviate the protocol. This can correspond, for example, to the use of a collateral (as in XCLAIM [15]), 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.

The Decentralized View Storage may also support more gateways simultaneously, instead of only two. This leads to much more possible synergies between different blockchains, yet it also entails much more complex logic and a broader array of attacks. Moreover, the assumption described in Section III-B which states that our solution assumes all views published in the DVS are valid can possibly be removed in future work. This means accepting the fact that auditor nodes can be malicious or assume that there are no auditor nodes at all, and present a solution for this challenge.

It is also interesting to leverage a crash-recovery mechanism 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. [8] presents a first approach to this problem, however it is not implemented yet.

An additional feature for future work can be the attestation of the smart contract code running on each gateway, or having nodes checking the smart contracts and guaranteeing they do not change. This is due to the fact that the contracts may contain malicious code that tries to unlock an asset right after locking it, for example. Finally, it can be interesting to see study if it is possible to decentralize the solution further, and see the limitations of the latter.

Acknowledgments: The authors would like to express a deep appreciation for everyone in the Hyperledger Cactus and Polkadot open-source communities who offered to help in this work. A special thanks to Peter Somogyvari for his unlimited patience, insight and support.

REFERENCES

[1] Archi – open source archimate modelling.
[2] hyperledger/cactus: Hyperledger cactus is a new approach to the blockchain interoperability problem.
[3] On public and private blockchains — ethereum foundation blog.
[4] Polkadot: Vision for a heterogeneous multi-chain framework.
[5] A trustless, general-purpose polkadot — ethereum bridge — by snowfork - polkadot — ethereum bridge.
[6] 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.
[7] R. Belchior, S. Guerreiro, A. Vasconcelos, and M. P. Correia. A survey on business process view integration. *ArXiv*, abs/2011.14465, 2020.
[8] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono. Hermes: Fault-tolerant middleware for blockchain interoperability, Mar 2021.
[9] 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.
[10] P. P. Ippolito. Game theory in artificial intelligence — towards data science.
[11] Z. Liu, N. C. Luong, W. Wang, D. Niyato, P. Wang, S. Member, Y.-C. Liang, and D. I. Kim. A survey on applications of game theory in blockchain. *arXiv*, 2019.
[12] P. Taylor, T. Dargahi, A. Dehghantanha, R. Parizi, and K.-K. R. Choo. A systematic literature review of blockchain cyber security. *Digital Communications and Networks*, 6, 02 2019.
[13] S. Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59:15–17, 10 2016.
[14] P. Wegner. Interoperability. *ACM Computing Surveys*, 28:285–287, 3 1996.
[15] 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.
[16] 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.