Resilient Gateway-Based N-N Cross-Chain Asset Transfers

André Augusto [1], Rafael Belchior [2,2,2,2], André Vasconcelos [1], Thomas Hardjono [1], and Miguel Correia [1]

[1]Affiliation not available
[2]Blockdaemon Ltd.

October 31, 2023

Abstract

New applications and solutions are emerging as blockchain technology continues to prosper in different industries. However, blockchain systems are considered isolated silos, especially when it comes to interoperability on systems putting restrictions on handling private data.

We propose ODAP-AS, a resilient N-N cross-chain asset transfer protocol that enables the execution of N transfers of assets in permissioned environments, leveraging the concept of gateways. Gateways act as the devices through which a blockchain network can be accessed. We build our protocol on top of the Open Digital Asset Protocol (ODAP), and its crash recovery mechanism, ODAP-2PC, a crash fault-tolerant protocol.

ODAP-AS also defines how one gateway is replaced by a backup in case of a crash. We implement a cross-chain asset transfer across Hyperledger Fabric and Hyperledger Besu using Hyperledger Cactus, which takes approximately 20 seconds. Additionally, we can conduct a sequential execution of ODAP-AS achieving 0.15 transactions/second throughput.

Multi-Party Cross-Chain Asset Transfers

André Augusto* Rafael Belchior*† Thomas Hardjono‡ André Vasconcelos* Miguel Correia*
*INESC-ID and Instituto Superior Técnico †Blockdaemon Ltd ‡MIT Connection Science & Engineering

***Abstract*—With the growing interest in blockchain technology, researchers and developers in different industries are shifting their attention to creating interoperability mechanisms. Existing mechanisms usually encompass asset exchanges, asset transfers, and general data transfers. However, most of the solutions based on these mechanisms only work for two permissionless blockchains falling short in use cases requiring more complex business relationships. Also, contrary to existing legacy systems, there is little standardization for cross-chain communication. Here we present MP-SATP, a resilient multi-party asset transfer protocol built on top of the Secure Asset Transfer Protocol (SATP). Furthermore, we enhance SATP's crash recovery mechanism that directly influences the reliability and performance of our solution. Using MP-SATP, we show how to perform N-to-N resilient asset transfers in permissioned environments by decoupling them into multiple 1-to-1 asset transfers. Our results demonstrate that the latency of the protocol is driven by the latency of the slowest 1-to-1 session; and how the usage of backup gateways avoids the overhead caused by rollbacks. Enterprise-grade environments such as supply-chain management systems can immediately leverage our solution to perform atomic multi-party asset transfers as shown by our use case.**

***Index Terms*—asset transfer, cross-chain, interoperability, multi-party, SATP**

I. Introduction

Interest in blockchain technology has risen since the appearance of Bitcoin [1] in 2008. Since then, many other cryptocurrencies and crypto-related projects were created, mainly in permissionless (public) networks. In the last few years, we have been witnessing a shift of attention to permissioned (or private) blockchains [2], where enterprises spread out across multiple industries have been adopting the technology [3]. Finance, healthcare, copyrights, and supply chain are some examples [4], [5].

A blockchain can be defined as an immutable distributed ledger, composed of a sequence of blocks that are cryptographically dependent on one another through cryptographic hash functions – in a way that if there is a change in one block, all the succeeding ones are invalidated. Blockchains can differ in multiple aspects, being the most obvious the security, privacy, and scalability guarantees [6].

To fully unlock the potential of a service provided to clients, it is essential to design a blockchain solution that can adapt to the unique requirements of each industry [7]. What may be deemed necessary in one industry may not hold the same significance in another, making the concept of a single blockchain governing the entire world impractical. Therefore, it becomes essential to incorporate interoperability mechanisms that facilitate cross-blockchain (or just cross-chain) communication, providing the foundational elements for seamless interaction and collaboration between different blockchain networks.

Blockchain Interoperability protocols fill this gap, allowing a source chain to change the state of a target chain through cross-chain transactions [3]. Several studies [3], [8], [9] categorize the different interoperability modes into *asset exchanges*, where assets in different blockchains are swapped between parties, such as atomic swaps [10]; *asset transfers*, where one asset is locked or burned (deleted) in the source chain and a representation is minted (created) in the target one [11]; and *data transfers*, where data is copied across blockchains, for instance, through the use of oracles [12].

We identify and address two gaps in the literature. Firstly, the majority of interoperability solutions focus on cross-chain communication between two parties at most. While some cross-chain communication protocols concentrate on 1-to-1 transfers, others that involve multi-party interactions are limited to asset exchanges through atomic swap protocols [10], [13], [14]. Secondly, we recognize a lack of research on interoperability within permissioned networks. The prevailing solutions are predominantly designed for permissionless networks, assuming that involved parties can access each other's internal state. In permissioned networks, unless explicit permission is given to all parties before the protocol, such protocols are deemed unpractical. Permissioned blockchains typically operate with decreased decentralization and employ consensus mechanisms that offer instant finality. These radically change the underlying trust assumptions and architectural design of cross-chain protocols. Notably, use cases like Delivery vs Payment (DvP) [15] or Central Bank Digital Currencies [16] are built upon permissioned networks, highlighting the importance of addressing interoperability challenges in this context.

We motivate the need for multi-party asset transfers using an example based on supply-chain management systems, which is further explored as a supply-chain use case in Section V. In this scenario, multiple business relationships are established within the supply chain network. For instance, a supplier may have individual agreements with various wholesalers, offering different pricing based on the products or volume of units purchased. Additionally, wholesalers may form partnerships with each other to devise tailored business strategies or collaborate on product creation. To elaborate, consider a scenario with two wholesalers and two producers. Each wholesaler initiates cross-chain transactions with their respective producer to facilitate payment for the goods shipped the other way around. Furthermore, the wholesalers rely on each other to assemble a final product, as different components (from both) are required.

In this scenario, one must address a potential issue where one party may refuse to pay its producer while the other already completed its transfer. This situation can lead to one party paying for goods that will be unutilized due to the lack of cooperation from the other wholesaler. In this complex supply chain network, consisting of multiple interconnected subnetworks or blockchains, there is a need to enable multi-party blockchain interoperability. Atomicity in this setting ensures that all parties involved can execute their asset transfers simultaneously, preventing one party from refusing to proceed when the others go through.

In response to the identified gaps in the literature, we present a novel solution called MP-SATP, which stands for Multi-Party Secure Asset Transfer Protocol. MP-SATP is specifically designed for permissioned networks and facilitates the transfer of multiple assets among N parties. Building upon the foundation of the Secure Asset Transfer Protocol (SATP), an ongoing work within the Internet Engineering Task Force (IETF), MP-SATP operates as a gateway-based protocol. Its primary objective is to enable atomic, fair, and consistent cross-chain asset transfers across participating ledgers.

One notable contribution of MP-SATP is its emphasis on cross-chain standardization, as it paves the way towards establishing immediate and consistent interoperation among diverse blockchains. This addresses a critical requirement in the current landscape where interoperability solutions are tailored for specific ledgers or use cases. To the best of our knowledge, MP-SATP represents the first multi-party blockchain interoperability solution dedicated to asset transfers between permissioned networks while also focusing on cross-chain standardization.

To further enhance the resilience of our proposed solution, we introduce a new primary-backup mode that enhances the gateway crash recovery procedure. This additional feature ensures a more robust and reliable system in the face of gateway failures, thereby augmenting the overall resiliency of MP-SATP.

This paper is structured in the following way. The background knowledge necessary for the understanding of this paper is presented in Section II. We present MP-SATP in Section III, and the primary-backup mode of SATP in Section IV. A use case using promissory notes is presented next in Section V. Section VI presents the implementation and evaluation details. Lastly, we present the Related Work and draw our conclusions in Sections VII and VIII.

II. Background

This section introduces the building blocks for MP-SATP. We walk through some of the most important blockchain interoperability concepts, the gateway-based architecture for interoperability; and finally SATP and its crash recovery mechanism.

A. Cross-Chain Asset Transfers

Solutions that perform cross-chain asset transfers follow roughly the same scheme: an asset is locked or burnt in the source chain, and either some asset is unlocked in the target chain, or a representation of the original one is minted there. The difference between locking and burning is tied to the concept of permanent or temporary transfers. In a permanent transfer the asset is permanently deleted in the source chain, whereas in a temporary transfer, it remains locked until it is transferred back. We represent a cross-chain transfer of an asset a between party $\mathcal{A}$ and party $\mathcal{B}$ as $\mathcal{A} \xrightarrow{a} \mathcal{B}$.

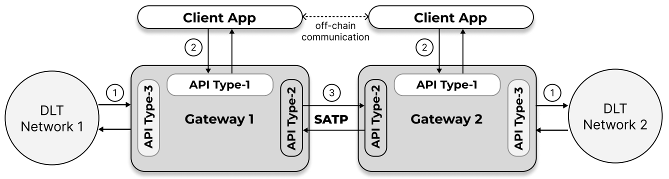

Fig. 1. Gateway-based architecture for blockchain interoperability. (1) Gateways have read and write access to a network; (2) Client applications can request gateways to perform asset transfers; (3) Gateways initiate sessions with other gateways to run a gateway-to-gateway protocol, such as SATP.

In this paper, we consider the case of permanent asset transfers, where an asset is burnt in the source chain, and a representation is created in the target one – i.e., with no obligation of being brought back to the original one.

B. Gateway-Based Blockchain Interoperability

Reference [17] proposes a singular perspective when thinking about interoperability architectures. The authors apply the same fundamental goals and architecture as in the early days of the Internet. At the time, the solution proposed to scale up and interconnect different networks was to implement *border gateway routers*. These routers provided entry points to each network. Mapping the concept to blockchains, a gateway-based architecture assumes one or more gateways deployed in front of each network that mediates traffic to/from each blockchain the same way routers forward data packets between networks. Gateways can thus be considered facilitators for performing cross-chain transactions. These gateways are owned and operated by specific entities that bear legal responsibility and adhere to regulatory compliance requirements relevant to the transferred assets. As trusted entities, gateways are well-suited for implementation within permissioned environments, where strict control and accountability are essential.

Figure 1 depicts the gateway architecture. There are three different APIs defined for a gateway: (1) a DLT-specific to interact with the ledger it has access to; (2) a client-specific API to receive requests from client applications; and (3) one that is reached by other gateways, to initiate gateway-to-gateway interactions. As an example, SATP is a gateway-to-gateway protocol. Note that the execution of the gateway-to-gateway protocol is always initiated by client applications.

C. Secure Asset Transfer Protocol (SATP)

The Secure Asset Transfer Protocol (SATP – previously called Open Digital Asset Protocol) [18] is being worked on at the Internet Engineering Task Force (IETF). It appears as the *"first cross-chain communication protocol handling* *multiple digital asset cross-border transactions by leveraging asset profiles (the schema of an asset) and the notion of gateways."* [19]. One of the goals for SATP is to enable communication across different domains – distributed ledgers, databases, or legacy systems.

In SATP, clients instantiate gateway-to-gateway interactions to perform asset transfers. Considering a source gateway $\mathcal{G}_S$, and a recipient gateway $\mathcal{G}_R$, an SATP session can be represented as $\mathcal{G}_S \overset{satp}{\rightarrow} \mathcal{G}_R$.

There are four phases in the protocol:

0) ***Identity and Asset Verification Flow***: gateways mutually verify their identities and the identities of their owners, ensuring that both gateways are valid (if gateways use trusted hardware this can be performed through attestation techniques);
1) ***Transfer Initiation Flow***: gateways exchange the communication terms and rules, making verifications regarding their jurisdictions and the asset that is being transferred;
2) ***Lock-Evidence Verification Flow***: the asset being transferred is locked, and a piece of evidence is presented to the other party;
3) ***Commitment Establishment Flow***: the involved gateways commit the changes and terminate the asset transfer. The commitment corresponds to the deletion of the asset in the source blockchain, and the creation of a representation in the target blockchain.

Note that all communication is done through a trusted communication channel using, for example, TLS.

D. SATP Crash Recovery Protocol

HERMES [19] proposed the crash recovery mechanism that allows any party running SATP to recover from a crash when exchanging messages to guarantee consistency across both blockchains. This mechanism leverages logs generated before and after each sent and received message. However, it is important to note that it focuses solely on crash failures and does not address Byzantine behaviour, which involves malicious deviations from expected protocol execution.

According to the authors, when a crash occurs, the Recovery Procedure must be triggered by the recovered gateway or a backup one. Therefore, there are two possibilities when a gateway crashes: 1) *self-healing*: the crashed gateway recovers and re-establishes communication with the counter-party gateway; or 2) *primary-backup*: a backup gateway resumes the execution of the protocol if the crashed gateway does not recover within a bounded time δ_t. The existing specification only mentions the necessity of such procedures, not proposing any concrete solution. In Section IV, we address this gap.

On the other hand, if there is no response from a gateway or its backup within $\delta_{rollback}$, s.t. $\delta_t < \delta_{rollback}$, there must be a rollback to ensure termination in a consistent state. A rollback is equivalent to issuing transactions with a contrary effect to the ones already issued [19]. When the crashed gateway or its backup is finally alive, it runs the Recovery Procedure, in which it learns the rollback performed by the other gateway. A rollback is triggered as well to guarantee consistency.

III. MP-SATP: Multi-Party Cross-Chain Asset Transfers

In this section, we present the building blocks for MP-SATP, a multi-party asset transfer protocol built on top of SATP. MP-SATP performs N-to-N transfers of assets through their decomposition in coordinated 1-to-1 SATP transfers.

A. General Assumptions

In this section we present the general assumptions in which we model MP-SATP:

- Gateways abide by regulatory compliance concerning the assets being transferred, being suitable for permissioned environments. In case of disputes, third-party audit entities can request asset transfer evidence.
- The underlying blockchains are *live* and *safe* – i.e. any valid transaction broadcast will eventually be added to the blockchain, and every correct node will eventually converge to a single truth.
- For a successful transfer, all clients involved in a multi-party asset transfer have previously agreed on transferring their assets, and authorize the respective gateway to act on their behalf. This is done through a shared transfer context.
- We bound the latency of any message by δ_t. This means that if no message is received within δ_t, it is assumed that a gateway has crashed.

B. Notation

Here we define the notation used to model our protocol. Furthermore, hereafter the concepts are presented based on an example, which is depicted in Figure 2. Let us consider a set of clients $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$ that want to engage in a multi-party asset transfer. Each client $\mathcal{C}_i$ has a wallet in blockchain $\mathcal{B}_i$, thus, we consider blockchain $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$. For simplicity, but without loss of generality, we only consider two transfers between elements of $\mathcal{C}$, $\mathcal{C}_1 \overset{a_1}{\rightarrow} \mathcal{C}_2$ and $\mathcal{C}_3 \overset{a_2}{\rightarrow} \mathcal{C}_4$; these represent the transfer of asset a_1 from $\mathcal{C}_1$ to $\mathcal{C}_2$, and the transfer of asset a_2 from $\mathcal{C}_3$ to $\mathcal{C}_4$. The goal is to ensure the atomicity of both transfers – i.e. either both succeed or both fail.

We also leverage gateways as entry points for the underlying blockchains, therefore, we denote as $\mathcal{G}_i$ a gateway with read and write access to $\mathcal{B}_i$, that will be reached by $\mathcal{C}_i$ to initiate cross-chain transfers. There may be multiple gateways connected to the same network which are used to parallelize cross-chain transactions and can also serve as backups to one another. We represent a backup gateway for $\mathcal{G}_1$ as $\mathcal{G}'_1$ (not represented in Figure 2).

C. System Model

As mentioned in Section III-A, clients are assumed to agree on the assets being transferred off-chain (e.g. match orders in an off-chain forum), building a graph $\mathcal{D}_1 = (\mathcal{V}_1, \mathcal{E}_1)$, where $\mathcal{V}_1$

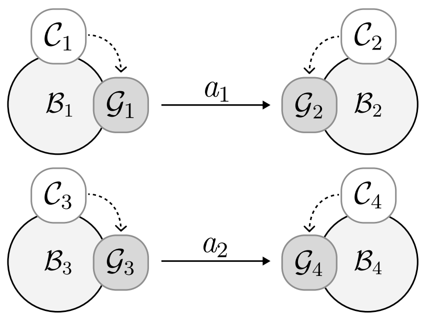

Fig. 2. Example of multi-party asset transfers between clients ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$) through the respective gateways. Asset a_1 is initially owned by $\mathcal{C}_1$ and a_2 by $\mathcal{C}_3$

is a finite set of vertexes, and $\mathcal{E}_1$ is a finite set of edges between elements of $\mathcal{V}_1$. $\mathcal{V}_1$ is the set of parties (clients $\mathcal{C}$) involved in the multi-party cross-chain asset transfer. Additionally, $\mathcal{E}_1$ is the list of cross-chain asset transfers between elements of $\mathcal{V}_1$. Each cross-chain transfer is a tuple ($\mathcal{C}_S$, $\mathcal{C}_R$, a), where $\mathcal{C}_S$ is the source client, $\mathcal{C}_R$ is the recipient client, and a is the profile of the asset being transferred. In Figure 2, $\mathcal{E}_1 = \{(\mathcal{C}_1, \mathcal{C}_2, a_1), (\mathcal{C}_3, \mathcal{C}_4, a_2)\}$.

The communication between clients, and consequently, between blockchains must be enabled through an interoperability mechanism [8]. In this solution, we leverage gateways as this component. Given that gateways run a gateway-to-gateway protocol, a mapping between each client and their respective gateways must exist. Therefore, in the gateway layer, the graph $\mathcal{D}_1$ must be translated into a graph $\mathcal{D}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ where $\mathcal{V}_2$ is the set of gateways that represent each client, and $\mathcal{E}_2$ is the previous list of cross-chain asset transfers concatenated with the respective gateways. This time, each cross-chain transfer is a tuple ($\mathcal{C}_S$, $\mathcal{C}_R$, $\mathcal{G}_S$, $\mathcal{G}_R$, a), where $\mathcal{C}_S$ is the source client, $\mathcal{C}_R$ is the recipient client, $\mathcal{G}_S$ is the source gateway, $\mathcal{G}_R$ is the recipient gateway, and a is the profile of the asset being transferred. In the given example, one would have $\mathcal{E}_2 = \{(\mathcal{C}_1, \mathcal{C}_2, \mathcal{G}_1, \mathcal{G}_2, a_1), (\mathcal{C}_3, \mathcal{C}_4, \mathcal{G}_3, \mathcal{G}_4, a_2)\}$.

Note that the assets being transferred in a single multi-party cross-chain asset transfer session can be heterogeneous; they might concern different fungible or even non-fungible assets.

D. MP-SATP Session Context

We have previously stated that clients authorize their gateways to act on their behalf for that specific asset transfer. This is done leveraging the concept of a session context *ctx*. It is calculated by hashing (through a cryptographic hash function $\mathcal{H}$) the concatenation of the graph $\mathcal{D}_2$, with the current timestamp ts, which avoids replay attacks. This context is then sent by each $\mathcal{C}_i$ to the corresponding $\mathcal{G}_i$ along with the graph $\mathcal{D}_2$.

$$ctx = \mathcal{H}(\mathcal{V}_2 \mid\mid \mathcal{E}_2 \mid\mid ts)$$

E. Protocol

MP-SATP is illustrated in Figure 3. Any client can initiate the protocol within its local gateway. In the above example, we assume that client $\mathcal{C}_4$ initiates MP-SATP by sending a request to $\mathcal{G}_4$. This particular gateway is referred to as the *coordinator* throughout the protocol. To initiate the transfer, the coordinator sends an *mp-satp-init* message containing the relevant context (ctx) to all the gateways involved in the asset transfer.

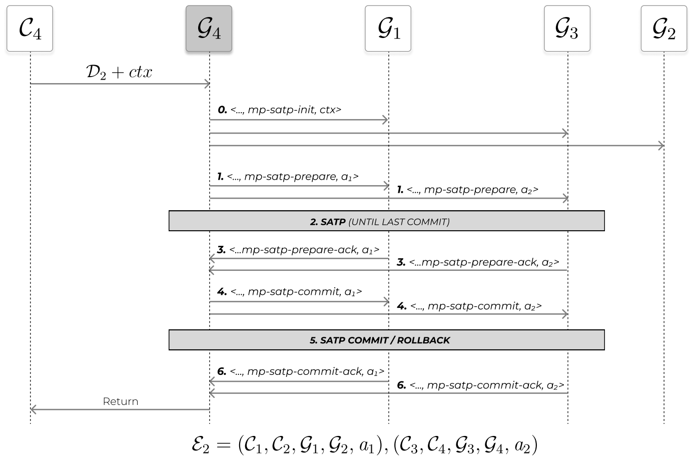
$\mathcal{C}_4$
$\mathcal{G}_4$
$\mathcal{G}_1$
$\mathcal{G}_3$
$\mathcal{G}_2$
$\mathcal{D}_2 + ctx$
0. <..., mp-satp-init, ctx>
1. <..., mp-satp-prepare, a_1>
1. <..., mp-satp-prepare, a_2>
2. SATP (UNTIL LAST COMMIT)
3. <...mp-satp-prepare-ack, a_1>
3. <...mp-satp-prepare-ack, a_2>
4. <..., mp-satp-commit, a_1>
4. <..., mp-satp-commit, a_2>
5. SATP COMMIT / ROLLBACK
6. <..., mp-satp-commit-ack, a_1>
6. <..., mp-satp-commit-ack, a_2>
Return
$\mathcal{E}_2 = (\mathcal{C}_1, \mathcal{C}_2, \mathcal{G}_1, \mathcal{G}_2, a_1), (\mathcal{C}_3, \mathcal{C}_4, \mathcal{G}_3, \mathcal{G}_4, a_2)$

Fig. 3. MP-SATP session initiated by a client $\mathcal{C}_i$. In this example we consider $\mathcal{G}_1 \overset{satp}{\rightarrow} \mathcal{G}_2$ and $\mathcal{G}_3 \overset{satp}{\rightarrow} \mathcal{G}_4$, where $\mathcal{G}_4$ was elected as the coordinator.

The protocol is further divided into two phases, assimilating with two-phase commit protocols: the *prepare* and *completion phases*. For clarity, we divide the *completion phase* into the *commit* and *rollback phases* according to the result of the *prepare phase*. Note that in the worst-case scenario, every client application requests its local gateway to initiate MP-SATP, however, the *prepare phase* guarantees that only one can be run successfully.

1) ***Prepare Phase****:* The coordinator is responsible for initiating an MP-SATP session with every source gateway $\mathcal{G}_S$ in the asset transfers list $\mathcal{E}_2$, through a *mp-satp-prepare* message. In Figure 2, $\mathcal{G}_4$ sends a *mp-satp-prepare* to $\mathcal{G}_1$ and $\mathcal{G}_3$, the source gateways in each cross-chain asset transfer. This first message includes the data necessary for each gateway to start its SATP 1-to-1 session with the corresponding counterparty gateway. This assimilates to the message sent by clients in a normal SATP 1-to-1 session when initiating a gateway-to-gateway interaction. Hence, after receiving the message, $\mathcal{G}_1$ and $\mathcal{G}_3$ initiate an SATP session with $\mathcal{G}_2$ and $\mathcal{G}_4$, respectively. This can be translated into $\mathcal{G}_1 \overset{satp}{\rightarrow} \mathcal{G}_3$ and $\mathcal{G}_2 \overset{satp}{\rightarrow} \mathcal{G}_4$. At this point, we are under the assumptions of SATP and its crash recovery mechanism, i.e. if there is a crash in one gateway the crash recovery procedure is executed, or in the worst case scenario the rollback. These gateways run SATP only until the end of phase 2, the Lock-Evidence Verification Flow. Every a_i in $\mathcal{E}_2$ should be locked in the corresponding source blockchains, which marks the end of the *prepare phase*. To indicate their readiness to proceed to the next phase, each source gateway in every SATP session acknowledges the coordinator, which gathers a set of *mp-satp-prepare-ack* responses with a boolean indicating the success or failure of the initial stage. If every gateway responds positively, the *commit phase* is initiated, otherwise, the *rollback phase*.

*2) **Commit Phase**:* If MP-SATP reaches the *commit phase*, every SATP session is ready to commit. Committing in a SATP session corresponds to deleting the locked asset in the source chain, and creating a representation of that asset in the target one. Therefore, the coordinator sends a *mp-satp-commit* message to every client gateway in $\mathcal{E}_2$. In each SATP session, the third (and last) phase is run. After the completion, a final *mp-satp-commit-ack* message is sent to the coordinator stating the success of the cross-chain transfer. Upon receiving a success message from every 1-to-1 SATP session, the coordinator can return to the client.

Note that the only reason why we need the first *mp-satp-init* message is for the recipient gateways (i.e., $\mathcal{G}_2$) to have knowledge of the initiation of the transfer and accept incoming SATP transfers according to the context provided by the client.

*3) **Rollback Phase**:* When MP-SATP reaches the *rollback phase*, at least one SATP session is not ready to commit. The coordinator sends a *mp-satp-rollback* message, which initiates the rollback in each SATP session. This includes issuing transactions that have the inverse effect of those that have previously been issued (e.g. if the asset was previously locked in the source blockchain, a transaction unlocking it must be issued to guarantee a consistent state across blockchains).

The communication between the coordinator gateway and every other gateway is done through the exchange of MP-SATP messages, whose format is specified in the following box.

MP-SATP Message Format

1) **Version**: MP-SATP protocol version;
2) **Message Type**: each message has a predefined schema (e.g., urn:ietf:mp-satp:msgtype:mp-satp-prepare);
3) **Context ID**: unique identifier representing an MP-SATP session;
4) **MP-SATP Phase**: the current phase of the protocol (init, prepare, commit, rollback);
5) **Sequence Number**: an increasing counter that uniquely represents a message from a session;
6) **Coordinator Gateway ID**: the public key of the coordinator;
7) **Recipient Gateway ID**: the public key of the gateway interacting with the coordinator;
8) **Payload**: any necessary payload including the profiles of the assets subject to transfer;
9) **Message Hash**: the cryptographic hash of this message;
10) **Signature**: the signature of this message;

IV. Enhancing SATP Crash Recovery

Given that our protocol is built on top of SATP, we also propose an enhancement to its crash recovery mechanism, directly impacting the guarantees of our solution. HERMES [19] proposes a crash fault-tolerant protocol for SATP. The authors of the paper assume that any gateway recovers from crashes within a defined bound of time, but in case of severe malfunctions (e.g. hardware failure), it might not be possible to recover within the defined amount of time, compromising atomicity. We remove the assumption that no gateway will ever crash indefinitely, and introduce backup gateways that are capable of resuming the execution of protocol on behalf of the crashed. Hence, we propose an extension to the existing protocol, where the main goal is to define *how a backup gateway can build trust with the counterparty's gateway to resume the execution of the protocol*.

A. Data Replication

To guarantee that a gateway can resume the execution of a SATP session, it needs to be up-to-date with the latest logs. The log storage infrastructure can be either centralized (e.g., locally or in the cloud) or decentralized (e.g., an IPFS network). Either way, given the possibility of dealing with private information, data must not be stored in cloud providers or IPFS networks unencrypted. To avoid the leakage of this information we leverage a primary-secondary scheme in which the primary gateway replicates log entries to all the backup gateways, and only the hashes of the logs are published to other platforms for integrity checks and eventual auditabilities.

B. Protocol Description

To demonstrate the solution we assume only one SATP session, given by $\mathcal{G}_1 \xrightarrow{\text{satp}} \mathcal{G}_3$. The proposed enhancement is based on X.509 certificates, therefore, we consider that every gateway has a valid X.509 certificate that was issued by its owner – the entity legally responsible for the gateway. Moreover, in the *extensions* field of the certificate, there is a list containing the hash of the authorized backup gateways. Assuming $\mathcal{G}_1'$ and $\mathcal{G}_1''$ backup gateways for $\mathcal{G}_1$, the *extensions* field of $\mathcal{G}_1$'s X.509 certificate is given by $\mathcal{L}_{\mathcal{G}_1} = [\mathcal{H}(Cert(\mathcal{G}_1')), \mathcal{H}(Cert(\mathcal{G}_1''))]$, where $\mathcal{H}(m)$ represents the cryptographic hash of m, and $Cert(\mathcal{G})$ represents the X.509 certificate for gateway $\mathcal{G}$.

As mentioned in Section III-A, if $\mathcal{G}_1$ does not send any message for δ_t, $\mathcal{G}_1'$ assumes the crash of $\mathcal{G}_1$. To avoid rollbacks, $\mathcal{G}_1'$ contacts $\mathcal{G}_3$ before $\delta_{rollback}$ to resume the execution of the open SATP session. The main issue here is how $\mathcal{G}_3$ knows that $\mathcal{G}_1'$ is authorized to replace $\mathcal{G}_1$ and resume the execution of the protocol. The solution proposed is based on three validations conducted by $\mathcal{G}_3$:

1) The certificate of $\mathcal{G}_1'$ must be valid – checked by running a certification path validation algorithm [20], which includes validating all the intermediate certificates up to a trusted root.
2) the parent certificate of both $\mathcal{G}_1$ and $\mathcal{G}_1'$ certificates is the same. In other words, both certificates must have been issued by the same institution, which proves they belong to the same legal entity.
3) $\mathcal{G}_1'$'s certificate hash belongs to the list specified in $\mathcal{G}_1$'s certificate extensions which indicates a set of gateways that are eligible to be the backup gateway in the case of a crash – i.e., $\mathcal{H}(Cert(\mathcal{G}_1')) \in \mathcal{L}_{\mathcal{G}_1}$. This is set by each entity when issuing a certificate for a gateway.

V. Use Case using Promissory Notes

We present a supply chain use case that would benefit from the implementation of our proposals, where multiple parties engage in a multi-party asset transfer.

A promissory note can be defined as a promise *"made by one or more persons to another, engaging to pay a certain sum of money subject to certain requirements as to the promise"* [21]. Replacement bills or notes issued by central banks can be substituted and must be signed by the promisor [22]. Recent advances in the financial industry have focused on the digitalization of promissory notes and their integration into blockchains, given that paper promissory notes are hard to track and require hand signatures [23], [24]. Furthermore, the concept of promissory notes in the interoperability of the blockchain-based on gateways was already proposed by [19], [24].

As we have been remarking through this paper, gateway-based interoperability solutions fit in the permissioned environment of enterprise solutions. Gateways are identified entities within an organization and comply with the existing regulations/legal frameworks imposed by the organization's home jurisdiction, making them suitable for this use case. We, therefore, present an example where a gateway-to-gateway protocol provides the building blocks for inter-jurisdiction asset transfers.

We leverage the example provided by [19] and extend it to realize an N-to-N atomic cross-jurisdiction asset transfer, using MP-SATP. The base example consists of two entities, a Producer (P) that sells goods to a Wholesaler (W). P issues an invoice for value V to W, which should be paid in a maximum of 90 days. Since P might not want to wait 90 days for the payment, it can request a promissory note stating that W will pay V to P in 90 days. This promissory note can now be sold by P to a third party.

Gateways can facilitate the transfer of promissory notes between different jurisdictions while abiding by the regulations on each end. Given this base illustration, we extend it to demonstrate MP-SATP in a similar supply chain example as depicted in Figure 4. Two wholesalers – W_1 and W_2 – form a consortium that, among other products sold individually, sells products in partnership. W_1 and W_2 depend on the products sold by two producers – P_1 and P_2 – respectively.

When P_1 sells goods to W_1, P_1 issues an invoice for value V_1, and requests a promissory note PN_1 stating the debt. The same happens between P_2 and W_2, with respect to a value V_2.

Given that W_1 and W_2 depend on one another to sell their final products, W_1 might not want to go into debt (buying and issuing a PN_1 to P_1) unless W_2 also buys the necessary amount of goods from P_2. We can therefore represent this problem as two independent asset transfers that need to be performed atomically: $W_1 \overset{PN_1}{\rightarrow} P_1$, and $W_2 \overset{PN_2}{\rightarrow} P_2$.

Considering $\mathcal{G}_{P1}$ as P_1's gateway, $\mathcal{G}_{P2}$ as P_2's gateway, $\mathcal{G}_{W1}$ as W_1's gateway, and $\mathcal{G}_{W2}$ as W_2's gateway we can leverage MP-SATP to perform multi-party cross-jurisdiction asset transfers $\mathcal{G}_{W1} \overset{satp}{\rightarrow} \mathcal{G}_{P1}$ and $\mathcal{G}_{W2} \overset{satp}{\rightarrow} \mathcal{G}_{P2}$, atomically.

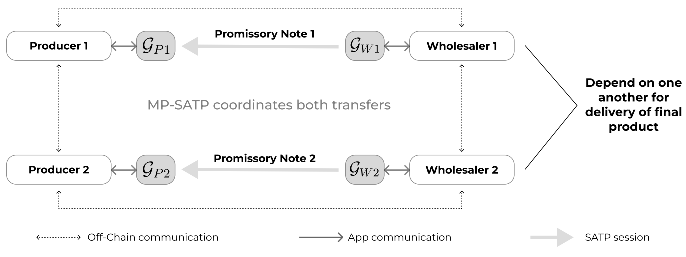

Fig. 4. Producer 1, Producer 2, Wholesaler 1, and Wholesaler 2 engaging in a multiparty asset transfer using MP-SATP.

VI. Implementation & Performance Evaluation

We implement MP-SATP in Hyperledger Cacti [25] in the form of a business logic plugin. We also develop the core SATP plugin and its crash recovery mechanism, given that our work is dependent on them. The total implementation reaches approximately 30k lines (including tests) of code and is expected to be merged into the main code base of the project in the near future. Furthermore, we present an initial evaluation of our proposals, including the overall latency of MP-SATP with asset transfers between Hyperledger Fabric and Hyperledger Besu networks. Finally, we show the performance gained through our primary-backup solution.

A. Hyperledger Cacti

Cacti is a project under the Hyperledger ecosystem. It allows users to make an adaptable and secure integration of different blockchains and provides a pluggable architecture that enables the execution of ledger operations across as many blockchains as needed. It leverages ledger connectors that serve as APIs to the underlying ledgers. One major advantage of using Cacti is that it is capable of handling the integration of both public and private blockchains. At the date of writing, the project has more than 1.5 million lines of code, 264 stars, and 223 forks on GitHub.

B. Architecture

We present a simplified architecture of the solution in Figure 5. Cacti offers support for API Servers, that receives a list of plugins – a plugin registry – and exposes the endpoints provided by those plugins. Communication between clients is performed off-chain. In this implementation, each gateway is represented by a business logic plugin (SATP plugin) that has multiple connections: 1) the local database to store logs generated by the execution of the protocol; 2) an IPFS connector to an IPFS network, which is used as decentralized log storage to guarantee availability and integrity of the logs; 3) ledger connectors that make possible the interaction with the underlying blockchains in the form of transactions. Each gateway has a different blockchain connector, providing interfaces for different blockchains.

We enhanced the SATP plugin and developed from scratch the MP-SATP plugin. The latter has a direct connection to the SATP plugin that initiates 1-to-1 transfers. The MP-SATP plugin contains the logic for every stage of the protocol

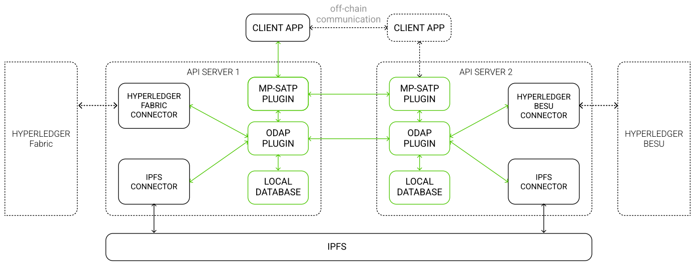

Fig. 5. MP-SATP and SATP Implementation architecture in Hyperledger Cacti. We represent in green our contributions. We leverage the Hyperledger Besu connector and the IPFS connector available in Cacti.

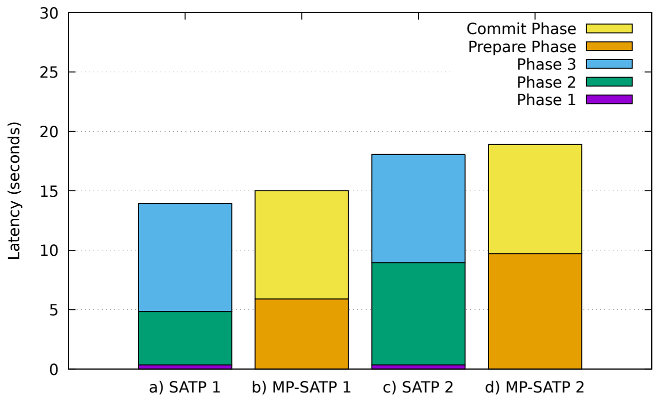

Fig. 6. (a) latency of running a single SATP session between Besu networks; (b) latency of running MP-SATP transferring 5 assets between Besu networks; (c) latency of running a single SATP session between a Fabric and a Besu network; (d) latency of running MP-SATP transferring 5 assets: 4 between Besu networks and 1 between a Fabric and a Besu network.

specified in Section III, and its algorithm is depicted in Algorithm 1. i stands for the index in the array $\mathcal{E}_2$, and t for every asset transfer tuple. Recall from Section III-C that each asset transfer in $\mathcal{E}_2$ is represented as ($\mathcal{C}_S$, $\mathcal{C}_R$, $\mathcal{G}_S$, $\mathcal{G}_R$, a).

Algorithm 1: MP-SATP algorithm

```
Input: $\mathcal{E}_2$
Result: True
$numberTransfers \leftarrow \mathcal{E}_2.length()$;
$prepareResponses \leftarrow [0..numberTransfers]$;
foreach $(i, t) \in \mathcal{E}_2$ do
    $prepareResponses[i] \leftarrow t.\mathcal{G}_S.initSATPAsync(t)$
wait() ;          // wait for every response
for $i \leftarrow 0$ to $numberTransfers$ do
    if $prepareResponses[i] \neq true$ then
        foreach $t \in \mathcal{E}_2$ do
            $t.\mathcal{G}_S.rollbackSATPAsync(t)$;
        wait() ;  // wait for every rollback
        return $False$;
foreach $t \in \mathcal{E}_2$ do
    $t.\mathcal{G}_S.commitSATPAsync(t)$
wait() ;              // wait for every commit
return $True$;
```

C. Testing Environment

All tests were run in a Google Cloud Compute Engine VM instance composed of 4 vCPUs, and 20 GB of memory, having a Boot Disk mounted using an Ubuntu 20.04 image, and a 100 GB SSD. As previously mentioned in Section VI, we leverage a Besu and a Fabric connector in Cacti. Hence, for testing purposes, we utilized the respective all-in-one Docker images available in Docker Hub. Every result presented in this section is the average of 100 independent runs.

D. MP-SATP Evaluation

We perform the evaluation of the protocol in two experiments using at most 10 different networks given the constraints of running multiple blockchains in a single machine. We start by creating an MP-SATP session composed of 5 asset transfers between Hyperledger Besu networks. Figure 6 *(a)* depicts the latency of one 1-to-1 SATP session between two different Besu networks; Figure 6 *(b)* depicts the latency of one MP-SATP session composed of 5 asset transfers between different Besu networks. The MP-SATP session has a slight overhead compared to the single 1-to-1 session, which is caused by the communication between the coordinator and every participant. The *prepare phase* in MP-SATP takes around one more second than phases 1 and 2 together in a single SATP session because the coordinator waits for all SATP sessions to return before sending the *mp-satp-commit* message. We hypothesise that the overall latency will always be tied to the latency of the slowest SATP session – i.e., with the highest latency.

In the second experiment, we replaced one of the 5 SATP transfers between Besu networks with an asset transfer between Fabric and Besu, to observe the change in the overall latency. Figure 6 *(c)*) depicts the latency of one 1-to-1 SATP session between a Fabric and a Besu network. Transactions take longer to be confirmed in the Fabric network, which explains the difference to Figure 6 *(a)*). To confirm our previous hypothesis we run MP-SATP where 4 transfers are still between Besu networks, and one is between a Fabric and a Besu network. Figure 6 *(d)* confirms our hypothesis.

These findings lead us to the predicted conclusion that the latency of such a protocol is strongly related to the confirmation times of the ledgers – i.e., the SATP session with the highest latency drives the total latency of an MP-SATP session. Formally, the latency of an MP-SATP session is given by $\max([Lat(\mathcal{E}_2^1), Lat(\mathcal{E}_2^2), ..., Lat(\mathcal{E}_2^n)])$, where Lat($\mathcal{E}_2^i$) is the latency of the i^{th} cross-chain asset transfer in $\mathcal{E}_2$.

E. SATP's Crash Recovery Enhancement

To understand the importance of our contribution to SATP's crash recovery mechanism through the primary-backup mode, we analyze the worst-case scenario with and without our solution. The worst-case scenario is a crash happening at the end of SATP's last phase, which would require triggering the rollback procedures. Firstly, we ran SATP without our solution.

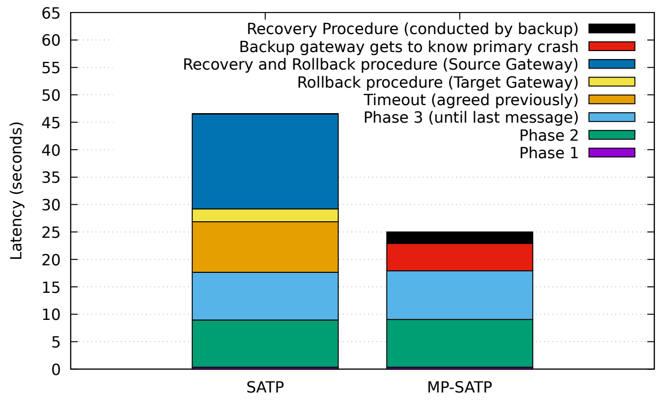

Fig. 7. Latency of an SATP session having a source gateway crash. When our proposal is not implemented both gateways rollback – i.e., every action is reverted. When using a backup gateway, it gets to know the crash of the original one and resumes the execution of the protocol – i.e., the asset is successfully transferred to the target chain.

We simulated the crash of the source gateway and let the target gateway timeout triggering the rollback procedure. We set it to 9 seconds – i.e., $\delta_{rollback} = 9sec$. When the crashed gateway recovers, it learns the rollback performed by the other gateway through the recovery procedure and rolls back as well.

In the second experiment we simulate the crash of the source gateway, however, this time we ensure there is a backup gateway that resumes the execution of the protocol right after $\delta_t = 5sec$ (before $\delta_{rollback}$). The backup gateway, up to date with the logs, only needs to run the recovery procedure and continue the execution of the protocol. Given that we simulate the crash in the last messages exchanged by gateways, as soon as the recovery procedure terminates, the protocol terminates as well. Without our proposal, the protocol terminates as it started (because all transactions were reverted) and takes, on average, around 46.3 seconds. With backup gateways, no rollback shall ever be triggered due to gateway crashes and the asset is successfully transferred to the target chain taking, on average, 25 seconds. The results are depicted in Figure 7. Choosing the right parameters δ_t and $\delta_{rollback}$ is critical for the success of this proposal. The difference between timeouts ($\delta_{rollback} - \delta_t$) must be enough for a backup gateway to retrieve the stored logs, reconstruct the SATP session, and send the initial recovery message to the counterparty gateway, to avoid the latter to rollback.

VII. Related Work

Here we present similar work that has been done to interoperate multiple blockchains.

Polkadot [26] and Cosmos [27] enable the interconnection of different chains through the Cross-Chain Message Passing Protocol (XCMP) [28], and the Inter-Blockchain Communication protocol (IBC) [29], respectively. XCMP enables the interoperation with more than two heterogeneous blockchains, however, IBC can only interoperate with up to two heterogeneous blockchains. These solutions can only interoperate blockchains in the same ecosystem which limits communication with the rest of the world.

Herlihy [10] proposed multi-party atomic cross-chain swaps, using HTLCs; however, it requires some assumptions. A cross-chain swap is modelled as a strongly connected directed acyclic graph, whose vertexes are parties and arcs are proposed asset transfers. The solution requires 1) a specific deployment order of smart contracts; 2) there can not be a cycle in the graph of transactions; 3) and most importantly, it cannot guarantee that an honest party does not lose its assets in case of a temporary crash. ACW^{3}N [30] appears as a solution for some of these issues with the introduction of a witness network where proofs are published, and where the global state is stored which can only be changed with a multi-signature from all involved parties. Lilac [13] proposes a multi-party asset exchange scheme, where assets can be locked in parallel. These solutions do not seamlessly work in permissioned blockchains unless access to those chains has been granted beforehand to every party involved in the swap.

Luo et al. [31] suggest an inter-blockchain architecture for routing management and transfer of messages between blockchains that requires a third-party blockchain. Fynn et al. [32] propose Move that enables the transfer of smart contracts between blockchains built on top of the EVM, by leveraging 2PC; however, it only focuses on 1-to-1 interactions. Wang et al. [14] also leverage 2PC to conduct transactions across N blockchains, however, the safety and liveness properties are not yet theoretically proved. If the coordinator crashes, atomicity is only guaranteed through the assumption that eventually a new coordinator is elected. Reference [14] presents a centralized component that performs actions in multiple blockchains based on a 2PC.

Some solutions also leverage Trusted Execution Environments (TEEs) [33]–[35] to perform cross-chain transactions, but they are limited to only two blockchains. These solutions provide more security guarantees in the relayers but lack scalability guarantees due to the physical restrictions imposed by the trusted hardware.

VIII. Conclusion

Because there are no solutions for the multi-party asset transfer problem focused on permissioned environments, this paper proposes MP-SATP, a protocol based on a 2PC to ensure coordination between the various entities and built on top of the Secure Asset Transfer Protocol (SATP). MP-SATP launches and coordinates multiple SATP sessions on multiple assets agreed upon by the clients. Additionally, we propose an improvement to the existing SATP's crash recovery procedure, in the primary-backup mode. From the implementation and evaluation of our proposals, we show that MP-SATP guarantees atomicity and finality properties. Additionally, with the use of gateways, one can guarantee the auditability of transfers of assets performed between gateways and compliance with legal frameworks. We also present a supply chain use case that would benefit from these new proposals.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System."

[2] B. C. Ghosh, T. Bhartia, S. K. Addya, and S. Chakraborty, "Leveraging Public-Private Blockchain Interoperability for Closed Consortium Interfacing," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, May 2021, pp. 1–10.

[3] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Computing Surveys*, vol. 54, no. 8, pp. 168:1–168:41, Oct 2021.

[4] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[5] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab, "Blockchain for Industry 4.0: A Comprehensive Review," *IEEE Access*, vol. 8, pp. 79 764–79 800, 2020.

[6] K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018, pp. 45–54.

[7] W. Chen, Z. Xu, S. Shi, Y. Zhao, and J. Zhao, "A survey of blockchain applications in different domains," in *Proceedings of the 2018 International Conference on Blockchain Technology and Application*, Dec 2018, p. 17–21. [Online]. Available: http://arxiv.org/abs/1911.02013

[8] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do You Need a Distributed Ledger Technology Interoperability Solution?" *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 1, pp. 1–37, Sep. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3564532

[9] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: Communication across distributed ledgers," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, N. Borisov and C. Diaz, Eds. Berlin, Heidelberg: Springer, 2021, p. 3–36.

[10] M. Herlihy, "Atomic Cross-Chain Swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, ser. PODC '18. New York, NY, USA: Association for Computing Machinery, Jul. 2018, pp. 245–254. [Online]. Available: https://dl.acm.org/doi/10.1145/3212734.3212736

[11] 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)*, May 2019, pp. 193–210.

[12] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges," *IEEE Access*, vol. 8, pp. 85 675–85 685, 2020.

[13] D. Ding, B. Long, F. Zhuo, Z. Li, H. Zhang, C. Tian, and Y. Sun, "Lilac: Parallelizing Atomic Cross-Chain Swaps," in *2022 IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2022, pp. 1–8.

[14] X. Wang, O. T. Tawose, F. Yan, and D. Zhao, "Distributed Nonblocking Commit Protocols for Many-Party Cross-Blockchain Transactions," Jan. 2020. [Online]. Available: http://arxiv.org/abs/2001.01174

[15] E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, and C. Vecchiola, "Enabling enterprise blockchain interoperability with trusted data transfer (industry track)," in *Proceedings of the 20th International Middleware Conference Industrial Track*, ser. Middleware '19. New York, NY, USA: Association for Computing Machinery, Dec 2019, p. 29–35. [Online]. Available: https://doi.org/10.1145/3366626.3368129

[16] A. Augusto, R. Belchior, A. Vasconcelos, I. Kocsis, G. László, and M. Correia, "CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains," Mar. 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/CBDC_bridging_between_Hyperledger_Fabric_and_permissioned_EVM-based_blockchains/21809430/2

[17] T. Hardjono, A. Lipton, and A. Pentland, "Toward an Interoperability Architecture for Blockchain Autonomous Systems," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1298–1309, Nov. 2020.

[18] M. Hargreaves, T. Hardjono, and R. Belchior, "Secure Asset Transfer Protocol (SATP)," Internet Engineering Task Force, Internet Draft draft-ietf-satp-core-01, Jun. 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-satp-core

[19] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, "Hermes: Fault-tolerant middleware for blockchain interoperability," *Future Generation Computer Systems*, vol. 129, pp. 236–251, Apr. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21004337

[20] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Internet Engineering Task Force, Request for Comments RFC 5280, May 2008. [Online]. Available: https://datatracker.ietf.org/doc/rfc5280

[21] D. M. Lobl, "Promissory notes," 2013.

[22] J. S. Waterman, "The Promissory Note as a Substitute for Money," *MINNESOTA LAW REVIEW*, 2019.

[23] "Electronic promissory notes on blockchain," https://www.gov.pl/attachment/e3ff4c9d-72f0-4ae4-89ac-f952f8ea666f, 2018, [Online].

[24] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, "Enabling Cross-Jurisdiction Digital Asset Transfer," in *2021 IEEE International Conference on Services Computing (SCC)*, Sep. 2021, pp. 431–436.

[25] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior, "Hyperledger cactus whitepaper." [Online]. Available: https://github.com/hyperledger/cactus/blob/main/docs/whitepaper/whitepaper.md

[26] D. G. Wood, "POLKADOT: VISION FOR A HETEROGENEOUS MULTI-CHAIN FRAMEWORK."

[27] J. Kwon and E. Buchman, "Cosmos whitepaper," *A Netw. Distrib. Ledgers*, p. 27, 2019.

[28] "Introduction to Cross-Consensus Message Format (XCM) · Polkadot Wiki," Jun. 2023. [Online]. Available: https://wiki.polkadot.network/docs/learn-xcm

[29] C. Goes, "The Interblockchain Communication Protocol: An Overview," Jun. 2020. [Online]. Available: http://arxiv.org/abs/2006.15918

[30] V. Zakhary, D. Agrawal, and A. El Abbadi, "Atomic commitment across blockchains," *Proceedings of the VLDB Endowment*, vol. 13, no. 9, pp. 1319–1331, May 2020. [Online]. Available: https://doi.org/10.14778/3397230.3397231

[31] L. Kan, Y. Wei, A. Hafiz Muhammad, W. Siyuan, L. C. Gao, and H. Kai, "A Multiple Blockchains Architecture on Inter-Blockchain Communication," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Jul. 2018, pp. 139–145.

[32] E. Fynn, A. Bessani, and F. Pedone, "Smart Contracts on the Move," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2020, pp. 233–244.

[33] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-Time Cryptocurrency Exchange Using Trusted Hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1521–1538. [Online]. Available: https://dl.acm.org/doi/10.1145/3319535.3363221

[34] Y. Lan, J. Gao, Y. Li, K. Wang, Y. Zhu, and Z. Chen, "Trustcross: Enabling confidential interoperability across blockchains using trusted hardware," in *Proceedings of the 2021 4th International Conference on Blockchain Technology and Applications*, ser. ICBTA '21. New York, NY, USA: Association for Computing Machinery, Feb 2022, p. 17–23. [Online]. Available: https://doi.org/10.1145/3510487.3510491

[35] Z. Yin, B. Zhang, J. Xu, K. Lu, and K. Ren, "Bool Network: An Open, Distributed, Secure Cross-Chain Notary Platform," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3465–3478, 2022.