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 avoid 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 rose since the appearance of Bitcoin [26] 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, where enterprises spread out across multiple industries have been adopting the technology [10]. Finance, healthcare, copyrights, and supply chain are some examples [2], [12].

A blockchain can be defined as a distributed ledger, composed of a sequence of blocks that are cryptographically dependent on one another – in a way that if there is a change in one block, all the succeeding blocks are invalidated [15]. It offers properties like immutability and decentralization. Blockchains can differ in multiple aspects, being the most obvious the security, privacy, and scalability guarantees.

Given the uniqueness of each industry, the design of a blockchain solution must enable the full potential of the service being offered to the respective clients [13]. What is sometimes considered mandatory for a use case or industry is not for another, which makes having only one blockchain to rule the entire world impractical. One must work in interoperability mechanisms that provide the building blocks for cross-blockchain communication.

Blockchain Interoperability (BI) fills this gap, allowing a source chain to change the state of a target chain through cross-chain transactions [10]. Several studies [7], [10], [32] categorize the different interoperability modes into *asset exchanges*, where assets in different blockchains are swapped between parties, such as atomic swaps; *asset transfers*, where one asset is locked or burned (deleted) in the source chain and a representation is created in the target one; and *data transfers*, where data is copied across blockchains, for instance, through the use of oracles [5].

The majority of interoperability solutions have been focused on cross-chain communication between at most two parties; there are either solutions between two entities and not extendable to multiple parties, or those that introduce multi-party interoperability are only related to asset exchanges – via atomic swap protocols [16], [20], [27]. The additional gap identified in the literature is the lack of studies focused on interoperability in permissioned networks. Existing solutions are designed for permissionless networks given that all parties need access to the internal state of all the others, which is impossible unless explicit permission is given to them before each swap (which is not practical). This represents a major drawback for use-cases based on supply chain management systems (e.g. Delivery vs Payment), or Central Bank Digital Currencies [6].

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 such an environment multiple business relationships can be created. A supplier might have individual agreements with different wholesalers – e.g. set a lower price because they buy more units of the product – or even just between wholesalers, where they partner with one another to tailor-make business strategies or create products to be sold. Considering two wholesalers and two producers. Each wholesaler issues cross-chain transactions to the respective producer to pay for the goods being shipped the other way around; additionally, we assume the wholesalers depend on one another to build a final product (e.g. different hardware parts are needed to build a computer). One party might claim the unwillingness to perform its transfer if the others do not go through as well. In a supply chain network composed of these multiple subnetworks (i.e. blockchains), containing complex dependencies between parties, one must enable multi-party blockchain interoperability in a way that N transfers of assets can be performed atomically.

Having identified the different gaps in the literature, we propose MP-SATP, a multi-party asset transfer protocol that enables the transfer of multiple assets between N parties in permissioned networks. MP-SATP is based on the Secure

Asset Transfer Protocol (SATP), a work in progress in the scope of the Internet Engineering Task Force (IETF). SATP is a gateway-based protocol, whose main objective is performing cross-chain asset transfers while guaranteeing atomicity, fairness, and consistency in the participating ledgers. It is also paving the way for standardization in cross-chain communication, working towards making the interoperation of heterogeneous blockchains immediate and consistent. This appears as a key requirement in a time when interoperability solutions are single-purpose, due to being built for specific ledgers or use cases. To the best of our knowledge, this is the first multi-party blockchain interoperability solution focused on asset transfers between permissioned networks, with an additional focus on cross-chain standardization. Given that our proposal is built on top of SATP, we also design a new primary-backup mode to enhance the gateway crash recovery procedure, automatically enhancing the resiliency of our solution.

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 present 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 our solution. 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, an asset transfer protocol paving the way for standardization in cross-chain communication.

A. Cross-Chain Asset Transfers

Currently, solutions that perform cross-chain asset transfers follow roughly the same scheme: an asset is locked or burnt in a source chain; and either some asset is unlocked in the target chain, or a representation of the original one (it can be some form of a wrapped asset) is minted there. The difference between these approaches is tied to the concept of permanent or temporary transfers – an asset is expected to be burnt in the source chain, or remains locked until it is transferred back, respectively. We represent a cross-chain transfer of an asset a between party $\mathcal{A}$ and party $\mathcal{B}$ as $\mathcal{A} \overset{a}{\rightarrow} \mathcal{B}$.

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, with no obligation of being brought back to the original.

B. Gateway-Based Blockchain Interoperability

Hardjono T. [18] proposes a singular perspective when thinking about interoperability architectures. An analogy between the early days of the Internet and the adoption of blockchains is made, which can be modeled by applying the same fundamental goals. At the time, the solution proposed to scale up and interconnect different networks was to implement *border gateway routers*, which would provide an entry point to each network. This new device would have distinct communication methods depending on if it was intra- or inter-domain routing. Mapping the concept to blockchains, a gateway-based blockchain interoperability 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 therefore be thought of as facilitators for performing cross-chain transactions. Gateways are owned by a certain entity that is legally responsible for them, abiding by regulatory compliance concerning the assets being transferred. Hence, we assume gateways are trusted entities, which makes the gateway-based architecture suitable for permissioned environments.

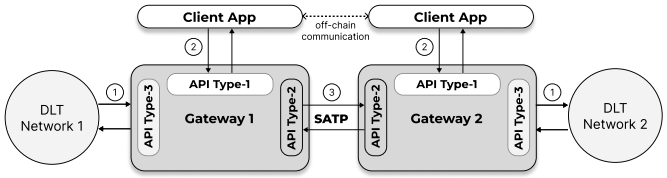

Fig. 1. Gateway-based architecture for blockchain interoperability. (1) Gateways have read and write access to a network; (2) Client applications can request gateways to perform asset transfers; (3) Gateways initiate sessions with other gateways to run SATP, a gateway-to-gateway protocol.

Figure 1 depicts the gateway architecture. There are three different APIs defined for a gateway: (1) one that is reached by other gateways, where gateway-to-gateway protocols are run; (2) a client-specific API to receive requests from the respective clients; and (3) a DLT-specific one, to interact with the ledger in which it is deployed. A client is always responsible for initiating the execution of a gateway-to-gateway protocol between its respective gateway and the corresponding one. During the execution of such protocol, transactions are issued to the underlying ledgers.

C. Secure Asset Transfer Protocol (SATP)

The Secure Asset Transfer Protocol (SATP – previously called Open Digital Asset Protocol) [19] is currently in specification in the scope of 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."* [9]. The main goal of the IETF working group is for SATP to become a standard in the communication across different domains - being them DLTs, distributed databases, legacy systems and so on.

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 [9] 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. It is based on the logs generated before and after each sent and received message. Currently, these protocols are focused only on failures by crashing and are not concerned with Byzantine behavior.

According to the authors, when a crash occurs, the Recovery Procedure needs to be triggered by the recovered gateway or by a backup one. Therefore, there are two possibilities when a gateway crashes: *self-healing*: the crashed gateway recovers and re-establishes communication with the other gateway; or a *primary-backup mode*: if the crashed gateway does not recover within a bounded time δ_t, a backup gateway resumes the execution of the protocol. The existing specification only mentions the necessity of such procedures, not proposing any concrete solution. We, therefore, explore in section IV a solution to the problem.

On the other hand, if there is no response from a gateway (or a 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 [9]. When the crashed gateway is finally alive, it runs the Recovery Procedure, in which it learns the rollback performed by the other gateway, rolling back as well.

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 nodes of each blockchain comply with the consensus protocol of the networks they belong to. We assume blockchains are secure – i.e. any valid transaction broadcast will eventually be added to the blockchain, and that 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 behalf of them (an example is the use of the *approve()* function in ERC-20[1] token standard for smart contracts).
- We assume the latency of any message is bounded 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 $\mathcal{C}$, $\mathcal{C}_1 \xrightarrow{a_1} \mathcal{C}_2$ and $\mathcal{C}_3 \xrightarrow{a_2} \mathcal{C}_4$; 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 $\mathcal{V}_1$ 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, $\mathcal{E}_1 = \{(\mathcal{C}_1, \mathcal{C}_2, a_1), (\mathcal{C}_3, \mathcal{C}_4, a_2)\}$.

[1]https://ethereum.org/en/developers/docs/standards/tokens/erc-20, accessed on December 18, 2022

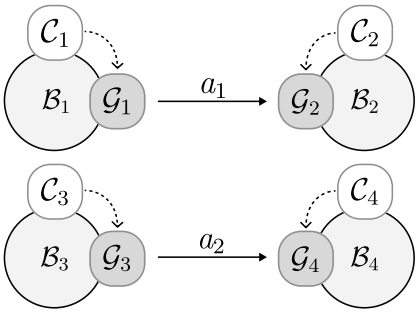

Fig. 2. Example of multi-party asset transfers between clients ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$) through the respective gateways. Asset a_1 is initially owned by $\mathcal{C}_1$ and a_2 by $\mathcal{C}_3$

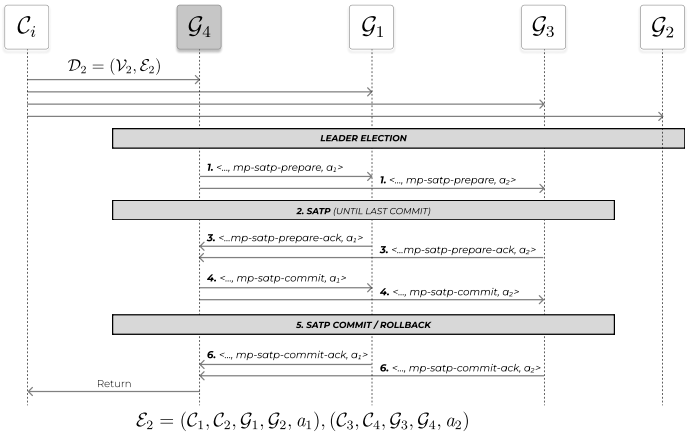

Fig. 3. MP-SATP session initiated by a client $\mathcal{C}_i$. In this example we consider $\mathcal{G}_1 \overset{satp}{\rightarrow} \mathcal{G}_2$ and $\mathcal{G}_3 \overset{satp}{\rightarrow} \mathcal{G}_4$, where $\mathcal{G}_4$ was elected as the coordinator.

The communication between clients, and consequently, between blockchains must be enabled through an interoperability mechanism [7]. 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 assets or even non-fungible ones.

D. Protocol

MP-SATP is depicted in Figure 3. A client in $\mathcal{C}$ is responsible for sending the graph $\mathcal{D}_2$ to all gateways in $\mathcal{V}_2$. A leader election protocol is initiated between all gateways to elect a gateway to manage the set of transfers specified in $\mathcal{E}_2$. Considering our model we leverage a simple leader election algorithm that uses the gateway identifier as a tiebreaker. The gateway with the highest ID is elected, and is therefore called the coordinator.

The core of the protocol, after the leader election, is 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 first one.

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 $\mathcal{E}_2$. The session is initiated through a *mp-satp-prepare* message. In the example depicted in Figure 2, whoever is elected as the coordinator 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. At this point, every a_i in $\mathcal{E}_2$ should be locked in the corresponding source blockchains, which marks the end of the *prepare phase*. If everything goes well until this stage, gateways are ready to commit. 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 successfulness of this first stage. If every gateway responds positively, the *commit phase* is initiated. Otherwise, the *rollback phase* is triggered.

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 by all source gateways in each SATP session 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 gateway can return to the client. Note that the return can be done in a previous stage according to the consistency model required by the client.

3) ***Rollback Phase****:* When MP-SATP reaches the *rollback phase*, at least one SATP session is not ready to commit.

This can be either because of an issue when issuing the lock transaction, or any disagreement between gateways when running SATP. In this situation, the coordinator is responsible for sending a *mp-satp-rollback* message, triggering the rollback in each SATP session. This encompasses issuing transactions with the contrary effect of the ones already issued (e.g. if the asset is locked in the source blockchain, a transaction unlocking the asset must be issued).

When MP-SATP reaches the *rollback phase*, at least one SATP session is not ready to commit. This might be due to a revert on the lock transaction or a dispute between gateways while performing SATP. In this case, the coordinator is in charge of delivering 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 the asset must be issued guaranteeing 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. Moreover, every message is encrypted and signed with gateways keys to ensure confidentiality and integrity, respectively.

MP-SATP Message Format

1) **Version**: MP-SATP protocol version;
2) **Message Type**: each message has a specified format (e.g., urn:ietf:mp-satp:msgtype:mp-satp-prepare);
3) **SessionID**: unique identifier (UUIDv2) representing an MP-SATP session;
4) **MP-SATP Phase**: MP-SATP phase (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 [9] 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 – jeopardizing the guarantee of 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 one. 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 counter party's gateway to resume the execution of the protocol*.

A. Data Replication

The first step to ensure that a gateway can resume the execution of the protocol is to be up-to-date with the latest operations – i.e., it has access to the logs of other SATP executions. There are multiple options concerning the log storage infrastructure: 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, one must ensure data is not 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 in another (possibly public) data structure for integrity checks and, eventually, for accountability.

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 $\mathcal{G}_1''$ a 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 gateway $\mathcal{G}$'s X.509 digital certificate.

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 $\delta_{rollback}$ expires, 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 the $\mathcal{G}_1$ and resume the execution of the protocol. The solution proposed is based on three validations conducted by $\mathcal{G}_3$ concerning the certificates of the gateways:

1) Validate $\mathcal{G}_1'$ certificate validity by running a certification path validation algorithm [14], which includes validating all the intermediate certificates up to a trusted root.
2) To ensure $\mathcal{G}_1'$'s ability and permission to replace $\mathcal{G}_1$, $\mathcal{G}_3$ needs to verify if the parent certificate of both $\mathcal{G}_1$ and $\mathcal{G}_1'$ certificates is the same. In other words, if both certificates were issued by the same institution, which proves they belong to the same legal entity.
3) Verify if $\mathcal{G}_1'$'s certificate hash belongs to the list specified in $\mathcal{G}_1$'s certificate extensions [14] 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

In this section, we present a simple 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"* [24]. Replacement bills or notes issued by central banks can be substituted and must be signed by the promisor [28]. 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 [3], [8]. Furthermore, the concept of promissory notes in the interoperability of the blockchain-based on gateways was already proposed by [8], [9].

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 [8] 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 in 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 P_2 also sells the necessary amount of goods to W_2. We can therefore represent this problem as two independent asset transfers that need to be performed atomically. The problem can be formulated as a set of transfers

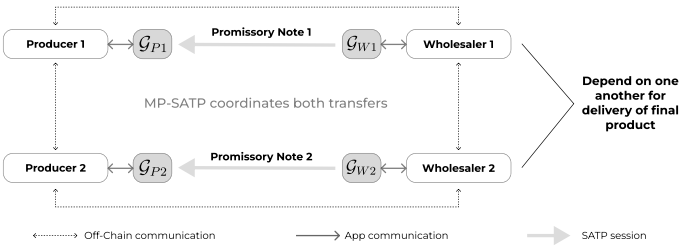

Fig. 4. Producer 1, Producer 2, Wholesaler 1, and Wholesaler 2 engaging in a multiparty asset transfer using MP-SATP.

– $W_1 \xrightarrow{PN_1} P_1$, and $W_2 \xrightarrow{PN_2} P_2$ – that must be atomic, either are both successful, or both failed.

If we consider $\mathcal{G}_{P1}$ 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 the multi-party asset transfers. The problem can, therefore, be formulated as $\mathcal{G}_{W1} \xrightarrow{satp} \mathcal{G}_{P1}$ and $\mathcal{G}_{W2} \xrightarrow{satp} \mathcal{G}_{P2}$ that must be atomic.

MP-SATP makes possible the execution of multi-party cross-jurisdiction asset transfers.

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 a 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 nearly 1.5 million lines of code, 250 stars, and 192 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. We assume client communication is performed off-chain and only communicates to the respective gateways the final graph of asset transfers. 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

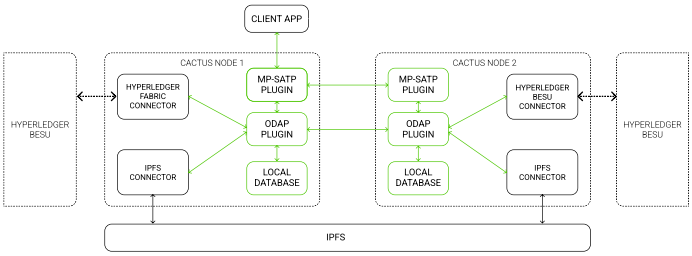

Fig. 5. MP-SATP and SATP Implementation architecture in Hyperledger Cacti. We represent in green our contributions. We leverage the Hyperledger Besu connector and the IPFS connector available in Cacti.

protocol; 2) an IPFS connector to an IPFS network, which is used as decentralized log storage to guarantee availability and integrity of the logs (only hashes of the logs are published to ensure confidentiality and integrity); 3) ledger connectors that make possible the interaction with the underlying blockchains in the form of transactions. Each gateway is connected to a different blockchain connector, that reaches a different blockchain.

We developed both the SATP and 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 specified in Section III, and its algorithm is depicted in Algorithm 1.

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 – Cacti Fabric All-In-One and Cacti Besu All-In-One – available in Docker Hub. Every result presented in this section is the average of 100 independent runs.

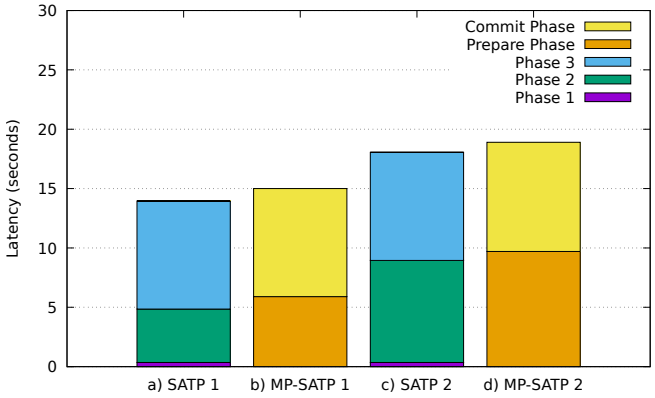

Fig. 6. (a) latency of running a single SATP session between Besu networks; (b) latency of running MP-SATP transferring 5 assets between Besu networks; (c) latency of running a single SATP session between a Fabric and a Besu network; (d) latency of running MP-SATP transferring 5 assets: 4 between Besu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

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/gateways. Figure 6 *(a)* depicts the latency of one SATP asset transfer between two different Besu networks; Figure 6 *(b)* depicts the latency of one MP-SATP session composed of 5 SATP asset transfers between different Besu networks. The MP-SATP session has a slight overhead compared to the single 1-to-1 session, that 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. This indicates that the 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. Given that transactions take longer to be confirmed in a Fabric network (as observed in Figure 6 *(c)*), the latency in this MP-SATP session is also expected to increase. In Figure 6 *(d)* we note that the overall latency is similar to a single Fabric to Besu asset transfer, even though there are still 4 transfers between Besu networks.

These findings lead us to the predicted conclusion that the latency of such a protocol is strongly related with 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

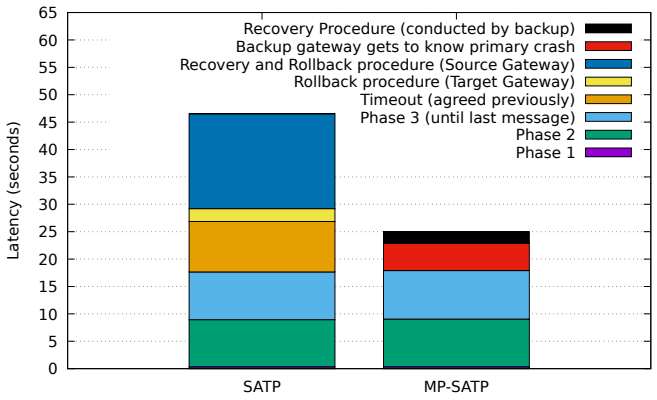

Fig. 7. Latency of an SATP session having a source gateway crash. When our proposal is not implemented both gateways rollback – i.e., every action is reverted. When using a backup gateway, it gets to know the crash of the original one and resumes the execution of the protocol – i.e., the asset is successfully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

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 ith cross-chain 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. In the worst-case scenario, there is a crash at the end of SATP's last phase. Firstly, we ran SATP without our solution, hence, we simulate the crash of a source gateway and let the target gateway timeout (set to 5 seconds), triggering the rollback procedure. 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 a source gateway, however, this time we ensure there is a backup gateway that resumes the execution of the protocol within $\delta_{rollback}$. This one only needs to run the recovery procedure and continue the protocol execution. 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.

VII. Related Work

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

Polkadot [29] and Cosmos [22] enable the interconnection of different chains through the Cross-Chain Message Passing Protocol (XCMP) [4], and the Inter-Blockchain Communication protocol (IBC) [1], 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 the communication with the rest of the world.

Herlihy [20] proposed multi-party atomic cross-chain swaps, using HTLCs; however, it requires some assumptions. A cross-chain swap is modeled 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 [31] 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 [16] 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. [21] suggest an inter-blockchain architecture for routing management and transfer of messages between blockchains that requires a third-party blockchain. Fyn et al. [17] 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. [27] 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 [27] presents a centralized component that performs actions in multiple blockchains based on a 2PC.

Some solutions also leverage TEEs [11], [23], [30] 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 this new proposals.

REFERENCES

[1] Inter-blockchain communication protocol. Technical report. [Online].

[2] Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.*, 14(4):352–375, jan 2018.

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

[4] Cross-consensus message format (xcm) · polkadot wiki. Technical report, 2020. [Online].

[5] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic. Trustworthy blockchain oracles: Review, comparison, and open research challenges. *IEEE Access*, 8:85675–85685, 2020.

[6] A. Augusto, R. Belchior, A. Vasconcelos, I. Kocsis, and G. László. CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains. 1 2023.

[7] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia. Do you need a distributed ledger technology interoperability solution? *Distrib. Ledger Technol.*, sep 2022. Just Accepted.

[8] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono. Enabling cross-jurisdiction digital asset transfer. In *2021 IEEE International Conference on Services Computing (SCC)*, pages 431–436, 2021.

[9] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono. Hermes: Fault-tolerant middleware for blockchain interoperability. *Future Generation Computer Systems*, 129:236–251, 2022.

[10] 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.

[11] 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*, CCS '19, page 1521–1538, New York, NY, USA, 2019. Association for Computing Machinery.

[12] 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*, 8:79764–79800, 2020.

[13] 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*, ICBTA 2018, page 17–21, New York, NY, USA, 2018. Association for Computing Machinery.

[14] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008.

[15] M. Di Pierro. What is the blockchain? *Computing in Science & Engineering*, 19(5):92–95, 2017.

[16] 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)*, pages 1–8, 2022.

[17] 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)*, pages 233–244, Los Alamitos, CA, USA, jul 2020. IEEE Computer Society.

[18] T. Hardjono, A. Lipton, and A. Pentland. Toward an interoperability architecture for blockchain autonomous systems. *IEEE Transactions on Engineering Management*, 67(4):1298–1309, 2020.

[19] M. Hargreaves, T. Hardjono, and R. Belchior. Secure Asset Transfer Protocol. Internet-Draft draft-hargreaves-sat-core-01, Internet Engineering Task Force, Nov. 2022. Work in Progress.

[20] M. Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18, page 245–254, New York, NY, USA, 2018. Association for Computing Machinery.

[21] 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)*, pages 139–145, 2018.

[22] J. Kwon and E. Buchman. Cosmos whitepaper, 2019.

[23] Y. Lan, J. Gao, Y. Li, K. Wang, Y. Zhu, and Z. Chen. Trustcross: Enabling confidential interoperability across blockchains using trusted hardware. In *2021 4th International Conference on Blockchain Technology and Applications*, ICBTA 2021, page 17–23, New York, NY, USA, 2022. Association for Computing Machinery.

[24] D. M. Lobl. Promissory notes, 2013.

[25] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior. Hyperledger cactus whitepaper.

[26] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[27] X. Wang, O. T. Tawose, F. Yan, and D. Zhao. Distributed nonblocking commit protocols for many-party cross-blockchain transactions, 2020.

[28] J. S. Waterman. The promissory note as a substitute for money. *Minn. L. Rev.*, 14:313, 1929.

[29] G. Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*, 21, 2016.

[30] 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*, 17:3465–3478, 2022.

[31] V. Zakhary, D. Agrawal, and A. El Abbadi. Atomic commitment across blockchains. *Proc. VLDB Endow.*, 13(9):1319–1331, jun 2020.

[32] 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 N. Borisov and C. Diaz, editors, *Financial Cryptography and Data Security*, pages 3–36, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.