Multi-Party Cross-Chain Asset Transfers

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

***Abstract*—Existing interoperability mechanisms usually encompass asset exchanges, asset transfers, and general data transfers. However, most of the solutions based on these mechanisms work only for pairs of permissionless blockchains, falling short in use cases that require more complex business relationships. Furthermore, contrary to existing legacy systems, there is little standardization for cross-domain communication, which multiple players in industry and academia are exploring. We present the *Multi-Party Secure Asset Transfer Protocol* (MP-SATP), a resilient multi-party asset transfer protocol built on top of the *Secure Asset Transfer Protocol* (SATP), which is being developed by the *Internet Engineering Task Force* (IETF). Furthermore, we enhance SATP's crash recovery mechanism to improve the reliability and performance of our solution. Using MP-SATP, we explain how to perform N-to-N resilient asset transfers in permissioned environments by decoupling them into multiple 1-to-1 asset transfers. Our results show that the latency of the protocol is driven by the latency of the slowest 1-to-1 session and that the use of backup gateways avoids the overhead caused by rollbacks.**

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

I. Introduction

In recent years, we have seen a shift in attention to permissioned (or private) blockchains, where companies spread across multiple industries have been adopting the technology [1]. Finance, healthcare, copyrights, and supply chain are some examples [2], [3].

In blockchain interoperability protocols, especially those focused on asset transfers, we identify and address two gaps in the literature. First, the majority of interoperability solutions focus on cross-chain communication between, at most, two parties. Although some cross-chain communication protocols focus on 1-to-1 transfers, others that involve multi-party interactions are limited to asset exchanges through atomic swap protocols [4]–[6]. Second, we observe a lack of research on interoperability among permissioned networks. The prevailing solutions are predominantly designed for permissionless networks, assuming that the parties involved can access each other's internal state. Permissioned blockchains provide lower decentralization but use consensus mechanisms that offer instant finality. The importance of such protocols is underlined by use cases like Delivery vs Payment [7] or Central Bank Digital Currencies [8].

In response to the identified gaps in the literature, we present the *Multi-Party Secure Asset Transfer Protocol* (MP-SATP). MP-SATP is specifically designed for permissioned networks and facilitates the transfer of multiple assets among N parties. Building on the SATP protocol [9], ongoing work within the IETF SATP working group [10], MP-SATP operates as a gateway-based protocol, i.e., it is executed by gateways. The *gateway-based architecture* is being actively explored by prominent players in academia and industry [11]–[13], by standardization organizations such as IETF [10] and ISO [14], and by financial corporations [15]. In this architecture, a blockchain has one or more *gateways* that allow the direct transfer of digital assets to another blockchain that has a compatible gateway.

One notable benefit of MP-SATP is its emphasis on cross-chain standardization, as it paves the way toward establishing immediate and consistent interoperation among diverse blockchains. 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 improves the gateway crash recovery procedure. This additional feature ensures a more robust and reliable system in the face of gateway failures, increasing the overall resiliency of MP-SATP.

We present the structure of this work. Section II presents the Secure Asset Transfer Protocol. MP-SATP is presented in Section III, and the primary-backup mode of SATP in Section IV. Section V presents the implementation and performance evaluation. Lastly, we present the Related Work and conclude in Sections VI and VII.

II. SATP

This section briefly introduces aspects of the SATP protocol [9] that are relevant for understanding MP-SATP.

Clients instantiate gateway-to-gateway interactions to perform asset transfers. We represent a cross-chain transfer of an asset a between party $\mathcal{A}$ and party $\mathcal{B}$ as $\mathcal{A} \xrightarrow{a} \mathcal{B}$.Considering a source gateway $\mathcal{G}_S$, and a recipient gateway $\mathcal{G}_R$, an SATP session can be represented as $\mathcal{G}_S \overset{satp}{\rightarrow} \mathcal{G}_R$. The SATP protocol has four phases:

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 (i.e., escrowed by the gateway), 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.

Hermes [16] proposed a crash recovery mechanism for SATP with two sub-protocols: 1) *self-healing*: the crashed gateway recovers and re-establishes communication with the counter-party gateway; 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.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 [9]. A rollback is equivalent to issuing transactions with a contrary effect to the ones already issued.

III. MP-SATP

This section presents 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. Notation

Here, we define the notation used to model our protocol. Some concepts are presented based on the example in Figure 1. 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 an account in blockchain $\mathcal{B}_i$, thus, we consider blockchain $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$. For simplicity, without loss of generality, we 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. We represent a backup gateway for $\mathcal{G}_i$ as $\mathcal{G}_i'$ (not represented in Figure 1).

B. System Model

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 1, $\mathcal{E}_1 = \{(\mathcal{C}_1, \mathcal{C}_2, a_1), (\mathcal{C}_3, \mathcal{C}_4, a_2)\}$.

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 $\mathcal{E}_2 = \{(\mathcal{C}_1, \mathcal{C}_2, \mathcal{G}_1, \mathcal{G}_2, a_1), (\mathcal{C}_3, \mathcal{C}_4, \mathcal{G}_3, \mathcal{G}_4, a_2)\}$.

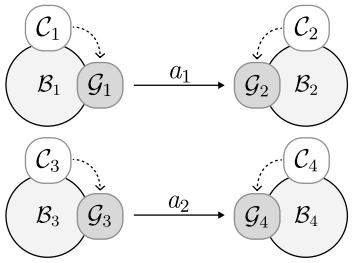

Fig. 1. Example of multi-party asset transfers between clients ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$) through the respective gateways. Asset a_1 is initially owned by $\mathcal{C}_1$ and a_2 by $\mathcal{C}_3$

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

C. MP-SATP Session Context

We stated that clients authorize their gateways to act on their behalf for that specific asset transfer. This is done by leveraging the concept of a session context *ctx*. A session context 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. Then each $\mathcal{C}_i$ sends this context to the corresponding $\mathcal{G}_i$ along with the graph $\mathcal{D}_2$.

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

D. Protocol

The execution of MP-SATP is represented in Figure 2. Any client can initiate the protocol within its local gateway. In our example, we assume that $\mathcal{C}_4$ initiates MP-SATP by sending a request to $\mathcal{G}_4$. This particular gateway is designated as the *coordinator*. To initiate the transfer, the coordinator sends a *mp-satp-init* message containing the relevant context (ctx) to all gateways involved in the asset transfer.

The protocol is divided into two phases, following a two-phase commit pattern: 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*. 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.

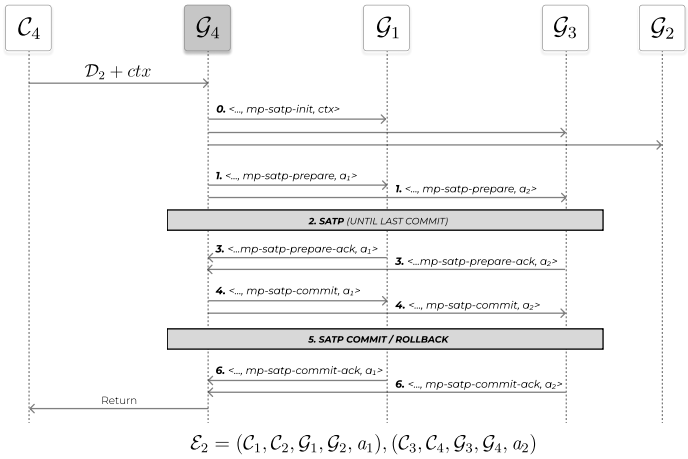

Fig. 2. MP-SATP session initiated by a client $\mathcal{C}_i$. In this example, we consider $\mathcal{G}_1 \overset{satp}{\rightarrow} \mathcal{G}_2$ and $\mathcal{G}_3 \overset{satp}{\rightarrow} \mathcal{G}_4$, where $\mathcal{G}_4$ was elected as the coordinator.

It includes the data necessary for each gateway to start its SATP 1-to-1 session with the corresponding counterparty gateway. Gateways run SATP only until the end of the Lock-Evidence Verification Flow, until every a_i in $\mathcal{E}_2$ is locked in the corresponding source blockchains. All source gateways acknowledge the coordinator.

2) ***Commit Phase****:* Committing to an 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, and a final *mp-satp-commit-ack* message is sent to the coordinator.

3) ***Rollback Phase*** *(optional):* 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.

IV. Enhancing SATP Crash Recovery

We also propose an enhancement to SATP's crash recovery mechanism, directly impacting the guarantees of our solution. We remove the previous 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. To ensure that a gateway can resume the execution of an SATP session, it is up-to-date with the latest logs. We leverage a primary-secondary scheme in which the primary gateway replicates log entries to all the backup gateways.

A. Protocol Description

To demonstrate the solution, we assume only one SATP session, given by $\mathcal{G}_1 \overset{\text{satp}}{\longrightarrow} \mathcal{G}_3$. The proposed enhancement is based on X.509 certificates, so we consider that every gateway has a valid X.509 certificate that was issued to its owner, the entity legally responsible for the gateway. Moreover, in the *extensions* field of the certificate, there is a list that contains 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 II, if $\mathcal{G}_1$ does not send any message for δ_t units of time, $\mathcal{G}_1'$ 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 [17], 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. Implementation & Performance Evaluation

We implement MP-SATP as a plugin of the Hyperledger Cacti interoperability framework [18]. Cacti has more than 2.5 million lines of code, 300 stars, and 254 forks in GitHub. 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. Every result presented in this section is the average of 100 independent runs.

A. MP-SATP Evaluation

We evaluate 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 blockchains. Figure 3 *(a)* depicts the latency of one 1-to-1 SATP session between two different Besu networks; Figure 3 *(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.

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 3 *(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 compared to Figure 3 *(a)*). To confirm our previous hypothesis, we run MP-SATP where 4 transfers

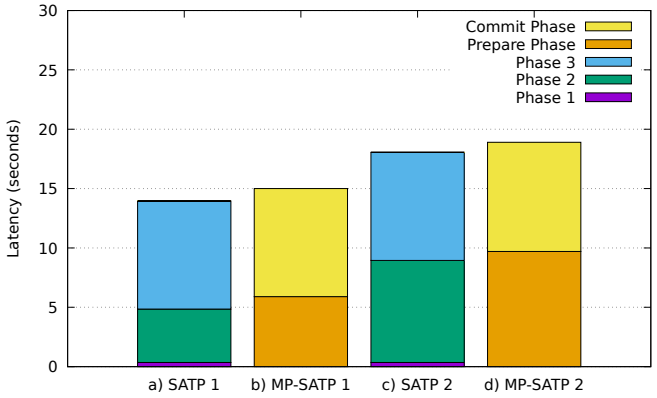

Fig. 3. (a) latency of running a single SATP session between Besu networks; (b) latency of running MP-SATP transferring 5 assets between Besu networks; (c) latency of running a single SATP session between a Fabric and a Besu network; (d) latency of running MP-SATP transferring 5 assets: 4 between Besu networks and 1 between a Fabric and a Besu network.

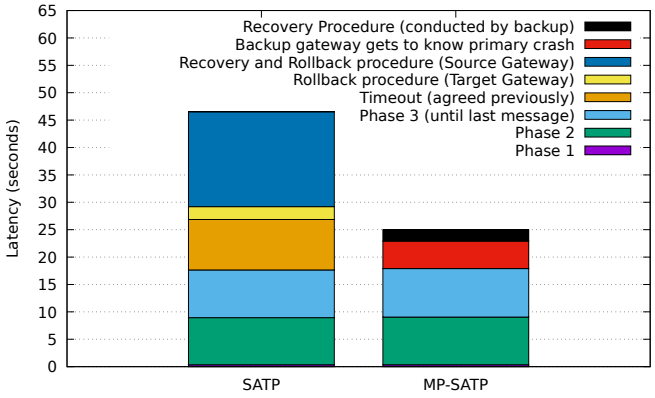

Fig. 4. Latency of an SATP session having a source gateway crash. When our proposal is not implemented, both gateways rollback – i.e., every action is reverted. When using backup gateways, one resumes the execution of the protocol upon crash of the original one – i.e., the result is the asset being successfully transferred to the target chain.

are still between Besu networks, and one is between a Fabric and a Besu network. Figure 3 *(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$^{\text{th}}$ cross-chain asset transfer in $\mathcal{E}_2$.

B. 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. First, we run SATP without our solution. We simulate the crash of the source gateway and let the target gateway timeout, triggering the rollback procedure. We set $\delta_{rollback} = 9sec$. When the crashed gateway recovers, it learns the rollback performed by the other gateway through the recovery procedure and rolls back. 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 $\delta_t = 5sec$ ($\delta_t < \delta_{rollback}$).

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 4. This showcases that the latency range of financial transactions processed by gateways is acceptable for modern financial infrastructure.

VI. Related Work

The interconnection of different chains through the Cross-Chain Message Passing Protocol (XCMP) [19] enables the interoperation of more than two blockchains. However, it can only interoperate blockchains in the same ecosystem, limiting communication with other solutions – not blockchain agnostic. Wang et al. [5] also leverage 2PC to conduct transactions across N blockchains. However, atomicity is not guaranteed if the coordinator crashes. Our protocol guarantees atomicity through the rollback of every state change. Multiple authors propose multi-party protocols but focused asset exchanges, which are only suitable to permissionless blockchains unless every party involved in the swap has been granted access to every network beforehand [4], [6], [20].

VII. Conclusion

To address 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 SATP. MP-SATP launches and coordinates multiple SATP sessions on multiple fungible or non-fungible assets agreed upon by the clients. We also improved the SATP 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, using gateways, one can guarantee the auditability of transfers of assets performed between gateways and compliance with legal frameworks.

Acknowledgements

This work was developed within the scope of the project nr. 51 "BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain", financed by European Funds, namely "Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial", included in the NextGenerationEU funding program. This work was also supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] X. Wang, O. T. Tawose, F. Yan, and D. Zhao, "Distributed nonblocking commit protocols for many-party cross-blockchain transactions," *arXiv preprint arXiv:2001.01174*, 2020.

[6] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2018, pp. 245–254.

[7] 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," in *Proceedings of the 20th ACM International Middleware Conference (Industrial Track)*, Dec 2019.

[8] A. Augusto, R. Belchior, I. Kocsis, L. Gönczy, A. Vasconcelos, and M. Correia, "CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023, pp. 1–9.

[9] R. Belchior, M. Correia, A. Augusto, and T. Hardjono, *SATP Gateway Crash Recovery Mechanism*, Jul. 2023, no. draft-belchior-satp-gateway-recovery-00, citation Key: belchiorSATPGatewayCrash2023a. [Online]. Available: https://datatracker.ietf.org/doc/draft-belchior-satp-gateway-recovery

[10] "Secure asset transfer protocol (satp) working group." [Online]. Available: https://datatracker.ietf.org/group/satp/about/

[11] V. Ramakrishna, "Making permissioned blockchains interoperable with weaver," Jul 2021. [Online]. Available: https://www.ibm.com/blog/making-permissioned-blockchains-interoperable-with-weaver/

[12] "DLT gateways and blockchain interoperability - DLT gateway interoperability." [Online]. Available: https://ledger.mit.edu/

[13] "Quant - bridging." [Online]. Available: https://quant.network/overledger-platform/bridging/

[14] ISO, "ISO/TC 307 - blockchain and distributed ledger technologies," Sep 2020. [Online]. Available: https://www.iso.org/committee/6266604.html

[15] S. Nazarov, P. Shukla, A. Erwin, and A. Rajput, "Bridging the governance gap: Interoperability for blockchain and legacy systems," World Economic Forum whitepaper, 2020. [Online]. Available: https://www.weforum.org/whitepapers/bridging-the-governance-gap-interoperability-for-blockchain-and-legacy-systems

[16] 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.

[17] 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

[18] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior, "Hyperledger cactus whitepaper," Hyperledger, Tech. Rep. 2, 2022. [Online]. Available: https://github.com/hyperledger/cactus/blob/main/docs/whitepaper/whitepaper.md

[19] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White paper*, vol. 21, no. 2327, p. 4662, 2016.

[20] V. Zakhary, D. Agrawal, and A. El Abbadi, "Atomic commitment across blockchains," *VLDB Endowment*, vol. 13, no. 9, p. 1319–1331, may 2020.