CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains

SUBMISSION DATE / POSTED DATE

03-01-2023 / 09-01-2023

CITATION

Augusto, André; Belchior, Rafael; Vasconcelos, André; Kocsis, Imre; László, Gönczy (2023): CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.21809430.v1

DOI

10.36227/techrxiv.21809430.v1

CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains

André Augusto* Rafael Belchior*† Imre Kocsis‡ László Gönczy‡ André Vasconcelos*
*INESC-ID and Instituto Superior Técnico †Blockdaemon Ltd ‡Dept. of Measurement and Inf. Systems, BME

***Abstract*—The last few years have seen a steep increase in blockchain interoperability research. Most solutions connect public blockchains; hence, the main cross-chain use case is token transfer. By-design platform transparency, tamper-resistance, and auditability make blockchains an infrastructure candidate for Central Bank Digital Currencies (CBDCs), but bridging CBDCs is an important missing piece in general. In this paper, we leverage an asset transfer protocol, ODAP/SATP, to define an extendable and dependable blockchain interoperability middleware that can bridge CBDC from Hyperledger Fabric to EVM-based permissioned blockchains. The key interoperation enabler in the solution is a shared asset definition enforced by both sides of the bridge, accompanied by a mapping between Fabric Identities and Ethereum addresses for Identity management. We implement our design for the CBDC use case utilizing Hyperledger Cactus. Through a preliminary performance evaluation, we show that the underlying ledgers heavily influence the latency of the solution, not the bridging components.**

***Index Terms*—CBDC, Interoperability, Hyperledger Besu, Hyperledger Fabric, ODAP/SATP**

I. Introduction

In general, *interoperability* across different distributed ledgers, maintained on different blockchain networks, refers to the ability to pass messages securely across smart contracts which manage the content of the distributed ledgers. Moving "units of value" in a predetermined way from one Distributed Ledger Technology (DLT) to another to support asset transfer and exchange scenarios is one aspect of blockchain and DLT interoperability.

Enabling the usually two-way movement of cryptoassets between blockchains – "*bridging*" – recently emerged as a key integration requirement and interoperability pattern in the unpermissioned, open-access blockchain world. *Consortial* distributed ledgers – ones with consensus participation tied to membership in a "consortium" of organizations and controls on network access – are beginning to follow. Important emerging use cases include temporarily moving money and money-like instruments, such as a *Central Bank Digital Currency* (CBDC)[15], from an authoritative, high-performance consortial distributed ledger to others dedicated to specific industrial and enterprise collaborations. In their bridged-out form, these instruments can serve as legal, non-volatile, and fungible payment and settlement vehicles for smart contracts – without impacting the performance of the authoritative asset ledger and preserving the collective business confidentiality of the collaborating parties.

Even for the unpermissioned blockchain world, cross-chain interoperability is still an emerging area (see, e.g., the survey [12]). There are still significant gaps in the necessary proper formal treatment of approaches, including the characterization of interoperability behavior ([9] is an important step forward) and an overall lack of standards.

Additionally, cross-consortial ledger bridging can have a fundamentally different trust model and requirement set than bridges for the unpermissioned world, especially for such assets as CBDCs, which will have heavily regulated operational models.

A. Interoperability for an ecosystem of CBDC applications

"A CBDC is a digital payment instrument, denominated in the national unit of account, that is a direct liability of the central bank"[15], complementing cash and traditional reserve or settlement accounts. Although there are only a few CBDCs already in production, with limited rollout or in minor economies, in recent years, most central banks have performed extensive research and experiments on the topic in preparation for issuing a CBDC in the future.

Whether the authoritative ledger of future CBDCs will be decentralized or centralized is still a subject of debate; experiments and prototypes exist for both. By-design tamper-resistance, auditability, and fault-tolerance are strong supporting arguments for permissioned distributed ledger-based implementations, even despite performance assurance, privacy, and operating consortium diversity challenges.

Blockchain-based applications implementing services for a given business domain (like logistics, retail, insurance, etc.) need a legally recognized vehicle for payment and settlement – and in many cases, same-chain CBDC will be the best option, when it becomes available [8].

Openly accessible documentation on the CBDC experiments and prototypes of central banks, such as work streams in the Digital Euro experiments [16], [7], strongly suggests that the core CBDC ledger will not provide wide-scale support for smart contracts. Instead, the application of interoperability solutions – classic payment initiation triggers, bridging, and payment channels – can be expected. For CBDC-using decentralized application ecosystems, arguably, bridging is the optimal solution, as it seamlessly enables performing the financial operations encoded in smart contracts.

In a wider context, it is also to note that the Multi-CBDC project (mCBDC [5]) of the Bank of International Settlements (BIS) demonstrated the usage of bridging mechanism in the context of wholesale (available only to financial institutions) CBDC and cross-border transactions.

B. Bridging CBDCs

In the cryptoasset world, bridging is predominantly based on *two-way pegs*, where the assets temporarily "moved" to another ledger are actually kept in cryptographically secured custody on the source chain until they are brought back. This approach is readily applicable for CBDC bridging between permissioned distributed ledgers, albeit with caveats regarding the threat model and necessary security guarantees. We summarize the key differences from permissionless approaches as follows:

- The party or consortium of parties performing the bridging can easily be sufficiently trustable and is, or are, incentivized by non-crypto means (for CBDC, they are expected to be regulated and auditable entities).
- Trust between the source and target ledgers can be either in place as a starting premise, or is much easier to achieve; e.g., by majority or threshold signature schemes over the known consensus-participating parties of the ledger which serves as a transaction source.
- As source and target ledgers typically employ deterministic transaction finality, questions of forks, chain splits, and chain dominance do not emerge as an issue.
- On the other hand, the atomicity, consistency, integrity, and durability of the asset transfer itself – as defined by [11] – gain primary importance.

Integration technologies are emerging to host solutions tailored to this setting, which, in comparison to cryptocurrency bridging solutions, can translate the different trust models to lower latency, significantly lower protocol complexity, and the absence of either *further* trust requirements or the introduction of cryptoeconomic incentives. Important examples include Hyperledger Cactus [26] and Weaver [2]. Work is also underway – within the scope of the Internet Engineering Task Force (IETF) - to define standard protocols for asset transfers with ACID guarantees between trusted ledgers in ODAP/SATP[20], which defines an asset transfer protocol between *gateways* attached to trusted ledgers. There is also an ACID property-preserving prototype implementation for Cactus, with the source and target gateways deployed in a Cactus network. However, as of this writing, no mature, general-purpose bridging solution is available for fungible asset transfers between permissioned ledgers.

C. Contribution

In this paper, as a novel contribution, we present the design and prototype implementation of a fungible asset bridge between Hyperledger Fabric and Hyperledger Besu (an Ethereum client implementation) operated by a trusted (regulated) party.

Our Fabric/EVM pairing is a representative case; many decentralized CBDC experiments apply Fabric and other dedicated "enterprise" blockchain technologies for CBDC implementation, while Ethereum technology is widely used for experiments as well as production systems in most application domains.

We build on the ODAP/SATP implementation by creating the Fabric and EVM (Ethereum Virtual Machine) side smart contract facilities for fungible asset bridging from Fabric to the EVM via two-way pegging, preserving the fault tolerance provided by the underlying protocol.

In our design, a single bridging entity operates both the source and target gateways. This eliminates the need for the still evolving lookup and discovery aspect of ODAP/SATP, necessary for gateway peering; and establishes a single, potentially regulated party who can be tasked with executing additional policies on "bridging out" and "bridging back" assets.

Executed policies can range from no-touch observation, such as creating audit trails, to compliance enforcement – e.g., halting the bridge-back operation on activities in the sidechain which are suspicious from a regulatory compliance point of view (the target ledger is trusted, not its users). Policies can be deployed in Cactus, which is purpose-built to host such "business logic", and are expected to be especially important in the CBDC context.

As transparent decentralized transaction validator capabilities are being worked on for Cactus, we expect our design to almost automatically gain support for permissioned distributed bridging in the near future – by the virtue of platform developments.

The rest of this paper is structured as follows. Section II presents the necessary technical background. Section III provides an overview of existing relevant blockchain interoperability approaches. Our solution design is introduced in Section IV. Sections V and VI present the implementation and a preliminary evaluation of the solution. We close out the paper with concluding remarks.

II. Background

We provide an overview of the technical background the paper builds on. We briefly introduce three related Hyperledger projects: Fabric, Besu, and Cactus; and describe ODAP/SAT, an asset transfer protocol between trusted blockchain networks.

A. Hyperledger Fabric

Hyperledger Fabric [3] is an open-source project under the Hyperledger umbrella that enables the creation of permissioned blockchains. Permissioned blockchains are usually used by organizations (or consortiums) that demand data sharing in such a way that all nodes are known and identified, contrary to widely used public blockchains, like Bitcoin [27]. Additionally, Fabric enables the deployment of smart contracts, called chaincode, using a variety of programming languages including JavaScript, Java, and Go. Fabric end-to-end transaction latency and throughput are engineerable system properties and can go as low as hundreds of milliseconds and as high as thousands of transactions per second.

B. Hyperledger Besu

Hyperledger Besu (or just Besu) is an open-source Ethereum client that is distinguished by its ability to create public as well

as private networks. Its use is directed to the enterprise environment, supporting a wide range of consensus mechanisms (e.g., PoW, PoA, and IBFT).

C. Hyperledger Cactus

Hyperledger Cactus [26] is a project in the Hyperledger ecosystem. Cactus is a blockchain integration framework that takes further steps when it comes to interconnecting enterprise-grade blockchain networks. It accomplishes that by offering a pluggable architecture that makes possible the execution of operations on as many networks as needed, through the usage of Business Logic Plugins (BLP) and Ledger Connectors. BLPs capture the necessary business logic for a certain application or protocol, whereas ledger connectors expose APIs facilitating the interaction with specific ledgers. For instance, the Fabric Ledger Connector provides an API so that any BLP can interact with an underlying Hyperledger Fabric network.

Recently, Cactus and Weaver [2], a Hyperledger Lab explained in Section III, have decided to merge their projects to form Hyperledger Cacti. Nonetheless, we refer to the project as Cactus given that the merge was not performed yet.

D. ODAP/SATP

The Secure Asset Transfer Protocol [20] (the naming is transitioning from Open Digital Asset Protocol) is an asset transfer protocol between two networks, based on relays [12], relying on trusted gateways to execute the protocol.

This gateway-based architecture can be compared with the concept of *Autonomous Systems* when the Internet was born [19]. At the time, the solution proposed to scale up and interconnect these networks was to implement *border gateway routers*, providing an entry point to each network. We can think of blockchains as the networks and gateways as the routers; these gateways run the Secure Asset Transfer Protocol, acting as the egress/ingress for data. A major advantage of this architecture is that the underlying ledgers do not need any kind of modification, thus making it ledger-agnostic.

In ODAP/SATP, client applications are responsible for communication with their local gateway in order to initiate a gateway-to-gateway asset transfer. In essence, the protocol is divided into three main phases/flows[1]:

1) *Transfer Initiation Flow*: gateways come to an agreement regarding the asset being transferred and exchange information on the legal frameworks under which they operate;
2) *Lock-Evidence Verification flow*: the asset being transferred is locked (no more transactions targeting the asset are approved) in the source chain and the proof is sent to the sidechain gateway;
3) *Commitment Establishment Flow*: both gateways commit the changes in their local ledgers, which corresponds to the deletion/burning of the asset in the source chain and the creation/minting of its representation in the sidechain.

[1]https://github.com/CxSci/IETF-SATP/blob/main/Figures/gateway-model-flows-v10PNG.png, accessed on October 15, 2022

For the time being, gateways are assumed to be trusted. In order to relax this trust assumption, progress has been made paving the way for trustless gateways, leveraging the concept of blockchain views [10], [1], [28]. In this case, gateways exchange verifiable proofs to attest to the success of the operations performed in each ledger. On the other hand, even though the implementation represents a non-trivial solution, one can think of securing gateways at the hardware level, through the deployment of such infrastructure in Trusted Execution Environments (TEE), such as Intel SGX [25] (a solution based on TEE to enable interoperability is [24]).

SATP is currently implemented in Hyperledger Cactus in the form of a Business Logic Plugin, along with its crash recovery mechanism [18] which provides recovery and rollback procedures in the presence of crashes. These procedures specify the steps necessary for a crashed gateway to resume the execution of the protocol, and in the worst case scenario rollback the execution – given that a DLT is an append-only data structure, rolling back the protocol corresponds to issuing transactions with the contrary effect of the ones already issued. For accountability, auditability, and integrity purposes, the implementation uses a decentralized log storage infrastructure (an IPFS network [14]), where gateways publish the proofs necessary for a successful asset transfer.

III. Related Work

In this section we present some solutions focused on interoperability between blockchains considering our CBDC bridging use case. We lay out the most representative solutions, hence, this list is not meant to be exhaustive.

An extensive survey on blockchain interoperability [12] classifies interoperability solutions into three categories: 1) *Public Connectors* provide interoperability between public blockchains – e.g., sidechains, notary schemes, and HTLCs; 2) *Blockchains of Blockchains* pave the way for the *"creation of application-specific blockchains that interoperate with each other"* [12] on top of existing infrastructure – e.g., Polkadot [32] and Cosmos [22]; 3) *Hybrid Connectors* encompasses the solutions that are not suitable to any of the previous two classes, mainly the ones directed to both public and private environments.

There are multiple trustless and privacy securing bridging solutions such as Falazi et al. [17], A. Xiong et al. [34], Horizon [23], Stone D. [31], and Bridging Sapling [29]. These solutions, integrated into the *Public Connectors* category, are either focused on permissionless blockchains supporting cryptocurrencies or do not have working implementations of the protocols. We also discard solutions that require both ledgers to have access to each other (e.g. SPV-like solutions), or that allow any user to become a bridging validator given the permissioned nature of our use-case (e.g., a group of validators running a consensus mechanism in order to accept/reject a cross-chain transaction, where anyone can run a node [34]).

Under the Hybrid Connectors [12] category, we find the *Trusted Relay* solutions where a trusted relay redirects packets

TABLE I
COMPARISON BETWEEN SOLUTIONS THAT SUPPORT PERMISSIONED NETWORKS. TRANSFERS AND EXCHANGES ARE BETWEEN FABRIC AND EVM BASED BLOCKCHAINS

	Provides Infrastructure	Asset Transfer	Asset Exchange	No changes to ledgers
Interledger	✓	?	✓	✓
Weaver	✓	✗	✓	✓
Cactus	✓	✓	✓	✓
YUI	✓	✓	✓	✗

in-between blockchains. Currently, there is only one implementation of a bridge between Fabric and EVM-based blockchains created by Datachain and NTT Data in cooperation. The bridge is created using the YUI Hyperledger Labs project[2], that leverages the Inter-Blockchain Communication protocol (IBC). The authors leverage an SPV-like architecture trusting on a relayer to forward the block headers of each blockchain as packets. Since both chains need IBC support, the necessity of making changes to the underlying ledgers, to support IBC, constitutes a downside of the solution. Additionally, Interledger [33] introduces a relay architecture similar to Cactus's that provides the underlying infrastructure for interoperability; however, the protocol to realize asset transfers was not found.

Weaver [2] is a Hyperledger Labs project that proposes a generalized protocol for data transfer within permissioned networks using trusted relay services. Relay services are responsible for representing a blockchain and running a protocol to transfer data between them. ODAP/SATP [20] is a similar solution where the relays are called gateways, providing a standardized communication protocol between gateways. While Weaver is focused on both the infrastructure and the communication protocol, ODAP/SATP is focused only on the standardization of the latter. Weaver only supports transfers of assets between Fabric and Corda [21], thus, for now, not suitable for our use case. Hence, we opt for using the ODAP/SATP implementation in Cactus as the underlying protocol and infrastructure for our solution.

Table I presents a comparison between these solutions. Interledger and Hyperledger Cactus are positioned as the best solutions that adhere to our requirements. Their architecture is very similar, however, we could not find the underlying protocol used for the execution of asset transfers. Also, given that Cactus presents a more mature project and includes an implementation of a future standard for communication between networks, we choose Cactus as the base for our solution.

IV. SOLUTION DESIGN

We propose a bridging approach between Fabric and Besu using Cactus and ODAP/SATP. In this section, we present our bridging model and describe the components of the architecture implementing the approach (see Figure 1).

[2]https://github.com/hyperledger-labs/yui-docs, accessed on October 15, 2022

A. Bridging Model

Any cross-chain bridge must have the means to translate data between blockchains. Technologically heterogeneous networks pose special challenges, such as nontrivial harmonization of protocol finalities due to different architectures and consensus mechanisms, or harmonizing the syntactic and semantic differences between communication interfaces and APIs. Bridging Fabric and Besu is certainly a heterogeneous setup: Besu is an Ethereum client (though we assume deterministic finality consensus), while Fabric uses a special execute-order-validate deterministic finality consensus approach [3] and a transaction and identity model radically different from Ethereum.

As justified in Section III, we base the architecture of the solution on Hyperledger Cactus. Cactus was envisioned to enable interoperability through a set of nodes (Cactus Nodes) that can together form a consortium and validate cross-chain transactions by running a consensus protocol [26]. It can be thought of as a relay solution that works in a trusted environment working towards a trustless setting in the future. Although the consortium feature is not yet fully available in the project, it is planned for the near future, therefore, for now, we leverage a single Cactus Node.

Each Cactus Node can be composed of multiple API Servers, hence our solution comprises two (Figure 1), each targeted to a different side of the bridge. We make use of both the Fabric and the Besu ledger connectors[3][4] as a means to access the ledgers; IPFS connectors allowing communication with an IPFS network that, as explained before, acts as decentralized log storage for integrity, accountability, and auditability purposes; finally, the ODAP/SATP business logic plugin[5]. The ODAP/SATP plugin acts as our gateway and exposes an API that is accessible to the end users to trigger bridging operations – *bridging out* or *bridging back* CBDC.

This bridging model can be easily extended to other ledgers with minor efforts, using ODAP / SATP, given that most existing blockchains support the deployment of smart contracts.

B. CBDC and Common Asset definitions

The integration of both Fabric and Besu can be achieved through either the definition of an asset that can be interpreted by both parties or through the development of a translation algorithm designed specifically for the required bridge. Because of its flexibility and simplicity, we opt for the first alternative; therefore, we build 1) a specification for the CBDC in which tokens are represented in each ledger through chaincode/smart contracts; and 2) a common asset definition called Asset Reference that represents a certain amount of CBDC to be bridged. This can be thought of as the piece that enables interoperability between both chains

[3]https://github.com/hyperledger/cactus/tree/main/packages/cactus-plugin-ledger-connector-fabric, accessed on October 15, 2022

[4]https://github.com/hyperledger/cactus/tree/main/packages/cactus-plugin-ledger-connector-besu, accessed on October 15, 2022

[5]https://github.com/hyperledger/cactus/tree/main/packages/cactus-plugin-odap-hermes, accessed on October 15, 2022

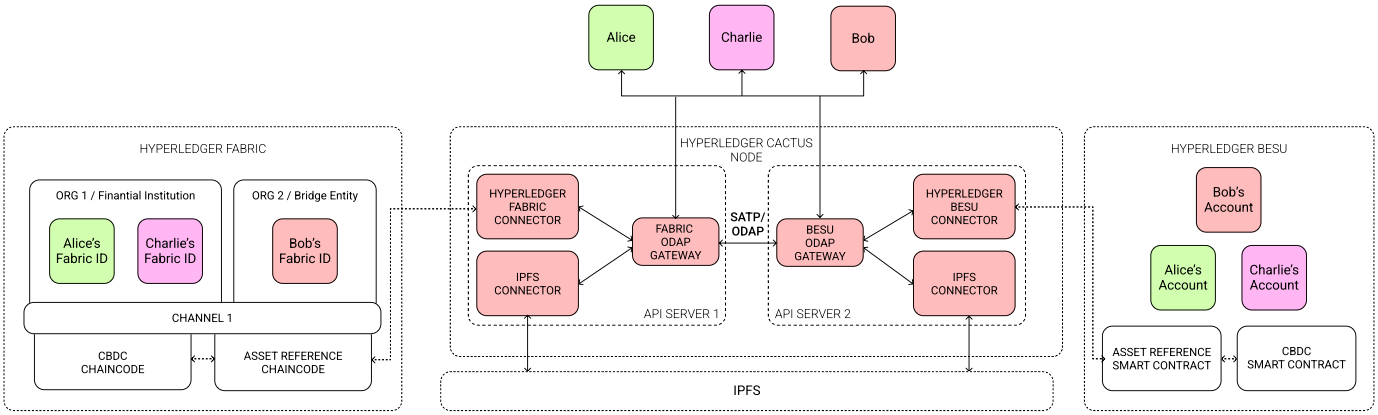

Fig. 1. Architecture of the solution – A bridge between Fabric and Besu using ODAP's implementation in Cactus

making the bridging operation possible. Thus, ODAP/SATP will not interact directly with the CBDC, but instead, with asset references.

The reasons for the creation of this layer are threefold. Firstly, the bridge is not bound to a specific CBDC implementation, thus serving as an interface; it protects against double-spending given that it offers a locking mechanism, hence, a client can not initiate two bridging operations on the same asset at the same time. Finally, it allows the storage of information about the CDBC locked in the source chain in another data structure.

In each ledger, the deployed smart contracts adhere to a specified mapping between the ledger-specific asset definition and this common asset definition of the bridging solution.

The Asset Reference and the CBDC smart contracts interoperate bidirectionally (explored in Section V-B) – when *bridging out* and *bridging back* CBDC. Hence, given these interactions, the Asset Reference smart contracts' access control is defined by only accepting requests from the bridge and the CBDC smart contract.

Each asset reference must be unique within the Asset Reference chaincode and smart contract respectively. The *identifier* is used exclusively by the clients to initiate bridging operations on the CBDC escrowed by them. The recipient of the Asset Reference (the sidechain when *bridging out*, and the source chain when *bridging back*), is responsible for translating the Asset Reference object into function calls, not needing the identifier. Given that a user can initiate two simultaneous bridge requests on the same Asset Reference, there must be a boolean indicating whether the asset reference is being bridged or not. If *isLocked* is true, simultaneous requests must be rejected. The *amount* field indicates the amount of CBDC represented by this Asset Reference. It matches the amount of CBDC put in custody by the final user. This final user is the *owner* of the Asset Reference. This is at the same time the sender and the recipient of the CBDC (explained in Section IV-C). The *tokenID* field is the ID of the token that is represented by this asset. This is due to the flexibility of the solution, whether multiple CBDC definitions can be supported (e.g. a digital euro or a digital USD).

C. Asset Transfer Model

Let us denote userA's Fabric Identity (FI) by Id_A, and its Ethereum address as $Addr_A$. We represent the *bridging out* of X CBDC from user A FI in Fabric to user A's address in Besu as $Id_A \xrightarrow{\text{X CBDC}} Addr_A$. The bridge back of X CBDC from the user A's address in Besu to the user A's FI in Fabric is represented as $Addr_A \xrightarrow{\text{X CBDC}} Id_A$.

Our ultimate goal is to perform a successful transfer of CBDC from a Hyperledger Fabric to a Hyperledger Besu network, however, we haven't defined yet what are the requirements that must be met for such an asset transfer to be deemed successful. We summarize them as follows:

1) the *bridging out* of X CBDC represented as $Id_A \xrightarrow{\text{X CBDC}} Addr_B$, is only valid if user A in the source ledger corresponds to User B on the target ledger;
2) the *bridging back* of X CBDC represented as $Addr_A \xrightarrow{\text{X CBDC}} Id_B$ is only valid if user A in the source ledger corresponds to User B on the target ledger;
3) the *bridging out* of X CBDC represented as $Id_A \xrightarrow{\text{X CBDC}} Addr_A$, is only valid if X CBDC were locked in Fabric and cannot be double spent in the origin chain (*two-way pegging* mechanism);

Note that in ODAP/SATP, the asset reference in the source chain is deleted before the creation of its representation in the target chain; however, the amount of CBDC it represents is not, it is maintained in locked/escrowed in the original one. In fact, it remains in secured custody until a user bridges back the CBDC to the source chain again – it might not be the same user because it can change hands in the sidechain. The amount of CBDC that is put in custody is constantly tracked, and it is possible (for the authorized members of the consortium – e.g., the Central Bank) to know at any time how much CBDC was bridged out at any moment. This plays a fundamental part in future auditability procedures.

D. Failure Model

Since our solution is built on top of the existing ODAP/SATP, it inherits its properties. Therefore, our CBDC

bridging solution guarantees atomicity, consistency, integrity, durability, and termination properties [11]. In addition, it also provides the logging infrastructure for future auditability procedures [18]. The protocol is not concerned with Byzantine (arbitrary) behavior from gateways; it supports crash faults and there is a crash recovery mechanism that comprises both a self-healing mode – where one gateway recovers by itself – or a primary-backup mode – one gateway is replaced by an authorized backup gateway. In the worst-case scenario, the rollback procedure is triggered if there is no communication beyond a defined timeout. An earlier analysis of the recovery and rollback procedure showed that the rollback is very costly and should be avoided as much as possible – it represents 38% of the total latency, compared to 0.5% when running the recovery procedure alone [6]. Hence, a conclusion from this work is that *"if we guarantee a backup gateway for each gateway running ODAP, the recovery procedure is always triggered to the detriment of the rollback."* [6]. This gives us some confidence in the performance of the protocol when the necessary infrastructure is provided, e.g., multiple gateways can serve as backups to one another.

E. Security Model

ODAP/SATP assumes correct behavior from all gateways involved in the bridging operation. Nonetheless, we protect the solution from any possible misbehavior from the final users when triggering bridging operations.

Section IV-C presents the Asset Transfer model which encompasses a set of requirements that must be followed to consider a bridging operation successful. In particular, the bridge rejects any request made to bridge CBDC from/to different users. Moreover, the CBDC and the Asset Reference smart contracts have access control policies that only expose a set of predetermined functions to the exterior based on their role. In the source chain, the CBDC chaincode is the one that can be directly accessed by the end users and only exposes the *Escrow* functionality. In turn, the Asset Reference chaincode can only be accessed by the bridge entity or by the CBDC smart contract (e.g., when tokens are escrowed/put in custody, which triggers the creation of an Asset Reference representing the same amount of CBDC). In the target chain, the CBDC smart contract is the only one that can be called by the end users – to escrow funds in order to bridge them back. Other operations exposed by the CBDC smart contract must only be exposed to the Asset Reference smart contract, which in turn can only be called by the bridging entity (e.g., deleting an Asset Reference triggers a call to the *burn* function of the CBDC smart contract).

V. Implementation

We implement the presented solution in Hyperledger Cactus as an application built on top of the existing ODAP/SATP business logic plugin, comprising around 4k lines of code, plus 2k for tests. Due to the double-blind review, we upload the source code[6], which will be merged into the main branch of the Hyperledger Cactus project in the near future.

A. Identity Management

In order to guarantee requirements 1 and 2 enumerated in IV-C, the bridge must be able to establish a mapping between Fabric Identities and Ethereum addresses.

We explore the possibility of generating Ethereum addresses based on the keys of the existing X.509 certificates used by Fabric; the elliptic curves used in Besu and Fabric-generated certificates turned out to be incompatible. Fabric only supports prime256v1, secp384r1, or secp521r1 curves[7], whereas Ethereum employs the secp256k1 curve[4], just like Bitcoin [27]. Therefore, we decide to create a registry that maps Fabric IDs to Ethereum addresses, and the other way around, in the Fabric ledger – the authoritative one. The current prototype implements one-to-one mappings, however, there is no architectural/technical obstacle to defining one-to-many mappings as well. As an example, this would enable the support for simultaneous transactions from one CBDC account to multiple Ethereum addresses participating in different business collaborations.

The bridge has the responsibility of verifying the compliance of each request with the content of the identity registry. These checks are performed in ODAP's first phase, where gateways also must reach an agreement on the asset to be transferred, the respective owner, and whether it complies with the asset transfer model detailed in Section IV-C.

B. Hyperledger Fabric

In our current implementation, we leverage a single-channel Fabric network composed of two organizations: Org1 (for CBDC holders and the issuing central bank), and Org2 (the bridging entity). In the prototype, we enroll two users in Org1 (Alice and Charlie) and one user in Org2 (Bob). Additionally, we leverage a solo ordering service [30] for testing purposes (a single node that simulates consensus). All of this is available in a Fabric Test Network provided by Cactus. It allows quickly spinning up a Fabric network with the above configuration while facilitating the deployment of the chaincode. The prototype is trivial to extend to a more refined multi-organization setup; the key point is the presence of the "bridging organization".

In the Fabric network, the CBDC smart contract provides out-of-the-box support for the well-known ERC20 Token Standard[8] – a specification of fungible tokens. Since the ERC20 token standard is directed to EVM-based blockchains, on the Fabric side, we leverage and extend the implementation of an existing ERC20 token standard for Fabric, in JavaScript,

[6]https://anonymfile.com/zE3J/cactus-example-cbdc-bridging-backend.zip, accessed on October 15, 2022

[7]https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/users-guide.html, accessed on October 15, 2022

[8]https://ethereum.org/en/developers/docs/standards/tokens/erc-20, accessed on October 15, 2022

available in the fabric-samples repository[9]. It is possible to use other token standards like ERC721[10] under some restrictions, namely supporting the *Escrow* and *Refund* functionality. The *Escrow* function gives the user the ability to escrow CBDC, which, in our solution, is implemented as a transfer of ownership of CBDC from the end user to a user belonging to Org2 (e.g., Bob). On the other hand, the *Refund* function reverts the operation performed by *Escrow*, thus, it can only be called by an Org2 user that has control over escrowed funds.

Furthermore, the Asset Reference chaincode offers not only the implementation of the Asset Reference data structure mentioned before, but also functionality such as *CreateAssetReference*, *DeleteAssetReference*, *LockAssetReference*, *UnlockAssetReference*, and *Refund*.

As previously stated, the Asset Reference and the CBDC smart contracts interoperate bidirectionally. When a user escrows funds in the CBDC chaincode, a new asset reference is created that represents the amount of CBDC that was escrowed. This asset reference is then transferred from Fabric to Besu using ODAP/SATP, which is used by the bridge to mint the corresponding CBDC to userA's Ethereum address in the sidechain. On the other hand, when bridging back, the bridge initiates the refund operation which triggers the refund of CBDC in the CBDC chaincode.

Note that, at any time, it is possible to query the CBDC chaincode on the Fabric side so as to retrieve the total value locked (TVL) in the chaincode at the moment.

C. Hyperledger Besu

Similarly to Fabric, we leverage a Besu all-in-one image provided by Cactus in the form of a Besu Test Network, creating user accounts for the same users as before: Alice, Charlie, and Bob.

On the Besu side, similarly to what was mentioned in the last section, we provide out-of-the-box support for the ERC20 Token Standard for the CBDC smart contract, but it may be implemented as other token standards as well. We summarize the requirements for an asset to be bridgeable from Besu to Fabric as having both the *Mint* and *Burn* functionality. The *Mint* function allows CBDC to be minted to an address. On the other hand, the *Burn* function reverts the operation performed by the previous one. Either operation can only be executed by the bridging entity through the Asset Reference smart contract.

Similarly to the Fabric side, the Asset Reference smart contract supports the following functionality: *CreateAssetReference*, *DeleteAssetReference*, *LockAssetReference*, *UnlockAssetReference*, and *Mint*. When *bridging out* CBDC, the *mint* function is called which triggers the minting of CBDC to the final user address. On the contrary, when bridging back CBDC, the *DeleteAssetReference* function is called by the gateway which burns tokens from the final user address in the CBDC smart contract. The ownership of the Asset Reference smart contract is given to the bridge entity.

D. Bridge Components

In Cactus, each API Server (containing the gateways, and the connectors to the networks) is exposed to the exterior and is accessible by the end users. We implement two extensions, one for each ledger, to the default behavior of an ODAP/SATP gateway by developing the ledger-specific functionality necessary to run the protocol (lock, delete, or create an asset reference). Moreover, we implement the functionality related to issuing the rollback transactions.

In the first phase of the protocol, where gateways must agree on the parameters of the transfer, a set of checks are run to make sure the operation is valid regarding the rules and requirements it must enforce in order to be deemed a valid transfer. These checks range from the amount of CBDC to be bridged, to the validity of the sender/receiver pair, and the asset reference.

E. Example Scenario

We demonstrate the bridging operations on the scenario captured by Figure 2, where Alice initiates the bridge out 500 CBDC to her address on the sidechain.

Alice starts by putting the 500 CBDC in custody by calling the *Escrow* function in the CBDC chaincode. Internally, it triggers the transfer of those funds to the bridge address and calls the *CreateAssetReference* function in the Asset Reference chaincode. An Asset Reference object (Listing 1) is created representing the escrowed amount.

Listing 1. Asset Reference representing 500 CBDC

```
{
  "id": "ID1",
  "isLocked": "false",
  "amount": 500,
  "owner": "Id_Alice",
  "tokenID": "CBDCX"
}
```

Alice now has the identifier (ID1) of the asset reference that represents her 500 CBDC that can be used to trigger the *bridging out* operation.

Alice can now use the asset reference ID1 to make a *bridge out* request to the gateway on the source chain – Gateway 1. Gateway 1 communicates with Gateway 2 to initiate the execution of ODAP/SATP. ODAP/SATP executes the locking and deleting of the asset in the source chain, and the creation of a representation in the sidechain [20], thus an asset reference would be created in the latter. Given that the bridged CBDC might change hands in the sidechain, we do not want to create the asset reference to avoid generating inconsistencies (e.g., an asset reference representing 500 CBDC exists, but the user has already transferred a portion of the bridged CBDC to another user). Hence, the bridge does not create an asset reference, instead, calls the *mint* function that is responsible for parsing

[9]https://github.com/hyperledger/fabric-samples, accessed on October 15, 2022

[10]https://ethereum.org/en/developers/docs/standards/tokens/erc-721, accessed on October 15, 2022

the Asset Reference with *id1* and mint 500 CBDC to Alice's address in the CBDC smart contract.

The opposite operation, bridging back, is not shown here for the sake of preserving space. To bridge back tokens, Alice needs to perform the same operations, but inversely. Tokens are escrowed in the sidechain, creating an asset reference that will be used for the bridging entity when running ODAP/SATP. Instead of minting tokens to the final user FI, there is a *Refund* operation in the source chain. This transfers the tokens that were in custody back to Alice.

Note that one can bridge back only a portion of CBDC that was bridged out initially. Additionally, Alice can bridge back tokens that were bridged out by other users who, in the meantime, transferred CBDC to her sidechain address.

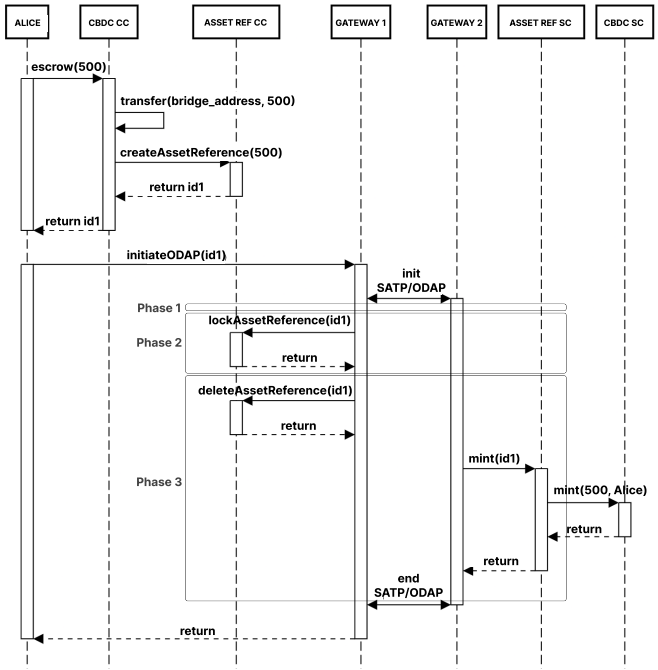

Fig. 2. CBDC bridging out sequence diagram

VI. Empirical Evaluation

We set up a pilot environment to perform an early evaluation of the proposed solution. We used a GCP Compute Engine VM, with an instance composed of 4 vCPUs, and 20 GB of memory, using an Ubuntu 18.04 image. We spun up a Fabric and Besu network using the publicly available Cactus Fabric All-In-One and Cactus Besu All-In-One Docker images. The results shown in this section are the average of 500 independent runs. We are interested in the latency introduced by the bridging solution; hence we separate the latency introduced by the steps executed on ledger platforms (i.e., lock/delete at the Fabric side and mint at the Besu side) from those of the bridging solution (steps executed by Gateways). Our goal is to perform a first validation and check whether the delay caused by the solution is within the acceptable range.

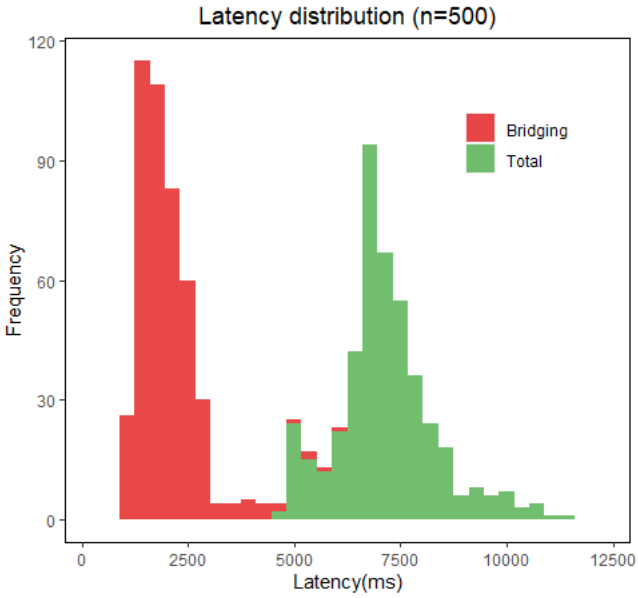

Fig. 3. Bridging protocol vs overall latency

Our preliminary results show that the overhead introduced by the bridge (compared to the steps which are performed within the blockchain networks) are in line with other recent studies (e.g., [13] using Cactus, separating intra and inter-blockchain communication) and in most cases, are below 30% of the end-to-end response time. There are of course several further improvements of the measurement configuration necessary before a well-founded benchmarking campaign: separation of components, changing consensus mechanism at the Besu side (from PoW to IBFT), and more precise workload definitions would be the most obvious ones.

VII. Conclusions

In this work, we contribute to the state of the art with an interoperability approach across permissioned chains in heavily regulated settings, while also adopting emerging standards that can alleviate some of the core interoperability problems. In specific, we leverage the ODAP/SATP protocol, under specification at IETF, to conduct cross-chain asset transfers between heterogeneous permissionless blockchains. We implement a CBDC bridging solution between Fabric and Besu, leveraging ODAP/SATP's implementation in Hyperledger Cactus. We build a common understanding on both sides of the bridge through a common asset definition that represents a certain amount of CBDC. Moreover, a mapping between Fabric Identities and Ethereum addresses is assembled to ensure client-initiated operations follow our Asset Transfer Model. The solution guarantees the *ACID* properties as well as *termination* and *auditability*. Finally, our analysis proves our initial hypothesis in which the total latency of the bridging solution is tightly coupled to the ledgers, thus the bridging components, i.e. gateways, do not incur in significant overhead.

References

[1] E. Abebe, Y. Hu, A. Irvin, D. Karunamoorthy, V. Pandit, V. Ramakrishna, and J. Yu. Verifiable observation of permissioned ledgers. *CoRR*, abs/2012.07339, 2020.

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

[3] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[4] A. M. Antonopoulos and G. Wood. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[5] R. Auer, P. Haene, H. Holden, et al. Multi-CBDC arrangements and the future of cross-border payments. *BIS Papers*, 2021.

[6] A. Augusto, R. Belchior, A. Vasconcelos, and T. Hardjono. Resilient Gateway-Based N-N Cross-Chain Asset Transfers. TechRxiv, Jun. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Resilient_Gateway-Based_N-N_Cross-Chain_Asset_Transfers/20016815.

[7] Banque de France. Digital euro experiment, combined feasibility – tiered model, July 2021. https://www.banque-france.fr/sites/default/files/media/2021/08/02/821220_digital_euro_en.pdf.

[8] A. Bechtel, A. Ferreira, J. Gross, and P. Sandner. The future of payments in a dlt-based european economy: A roadmap. In *The Future of Financial Systems in the Digital Age*, pages 89–116. Springer, Singapore, 2022.

[9] R. Belchior, P. Somogyvari, J. Pfannschmid, A. Vasconcelos, and M. Correia. Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions. TechRxiv, Sep. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Hephaestus_Modelling_Analysis_and_Performance_Evaluation_of_Cross-Chain_Transactions/20718058.

[10] R. Belchior, L. Torres, J. Pfannschmid, A. Vasconcelos, and M. Correia. Is My Perspective Better Than Yours? Blockchain Interoperability with Views. TechRxiv, Jun. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Is_My_Perspective_Better_Than_Yours_Blockchain_Interoperability_with_Views/20025857.

[11] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono. Hermes: Fault-tolerant middleware for blockchain interoperability. *Future Generation Computer Systems*, 2021.

[12] 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.

[13] P. Bellavista, C. Esposito, L. Foschini, C. Giannelli, N. Mazzocca, and R. Montanari. Interoperable Blockchains for Highly-Integrated Supply Chains in Collaborative Manufacturing. *Sensors*, 21(15):4955, July 2021.

[14] J. Benet. IPFS – Content Addressed, Versioned, P2P File System. arXiv, 2014. [Online]. Available: http://arxiv.org/abs/1407.3561.

[15] BIS Innovation Hub. Central bank digital currencies: foundational principles and core features, October 2020. https://www.bis.org/publ/othp33.pdf.

[16] U. Emanuele, B. Alessia, C. Daniele, C. Angela, C. Marco, F. Simone, G. Giuseppe, G. Giancarlo, M. Gabriele, T. Pietro, and V. Alessia. A digital euro: a contribution to the discussion on technical design choices. *Mercati, infrastrutture, sistemi di pagamento*, (10), 2021.

[17] G. Falazi, U. Breitenbücher, F. Daniel, A. Lamparelli, F. Leymann, and V. Yussupov. Smart contract invocation protocol (scip): A protocol for the uniform integration of heterogeneous blockchain smart contracts. In S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, editors, *Advanced Information Systems Engineering*, pages 134–149, Cham, 2020. Springer International Publishing.

[18] T. Hardjono, R. Belchior, M. Correia, and A. Augusto. Dlt gateway crash recovery mechanism (draft-belchior-gateway-recovery-04). https://datatracker.ietf.org/doc/html/draft-belchior-gateway-recovery-04, 2021. [Online].

[19] T. Hardjono, A. Lipton, and A. Pentland. Towards an interoperability architecture for blockchain autonomous systems. *IEEE Transactions on Engineering Management*, 67(4):1298–1309, 2019.

[20] M. Hargreaves, T. Hardjono, and R. Belchior. Secure Asset Transfer Protocol draft 00. Internet-Draft draft-hargreaves-sat-core-00, Internet Engineering Task Force, 2021. [Online]. Available: https://datatracker.ietf.org/doc/draft-hargreaves-sat-core/.

[21] M. Hearn and R. G. Brown. Corda: A distributed ledger. *Corda Technical White Paper*, 2016, 2016.

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

[23] R. Lan, G. Upadhyaya, S. Tse, and M. Zamani. Horizon: A gas-efficient, trustless bridge for cross-chain transactions. *CoRR*, abs/2101.06000, 2021.

[24] M. Li, J. Weng, Y. Li, Y. Wu, J. Weng, D. Li, G. Xu, and R. Deng. Ivycross: A privacy-preserving and concurrency control framework for blockchain interoperability. Cryptology ePrint Archive, Paper 2021/1244, 2021. https://eprint.iacr.org/2021/1244.

[25] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, HASP 2016. Association for Computing Machinery, 2016.

[26] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior. Hyperledger cactus whitepaper. https://github.com/hyperledger/cactus/blob/main/whitepaper/whitepaper.md, 2020. [Online].

[27] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[28] C. Pedreira, R. Belchior, M. Matos, and A. Vasconcelos. Securing Cross-Chain Asset Transfers on Permissioned Blockchains. TechRxiv, 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Trustable_Blockchain_Interoperability_Securing_Asset_Transfers_on_Permissioned_Blockchains/19651248, 2022.

[29] A. Sanchez, A. Stewart, and F. Shirazi. Bridging sapling: Private cross-chain transfers, 2022.

[30] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, and M. Guizani. Performance evaluation of hyperledger fabric. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 608–613, 2020.

[31] D. Stone. Trustless, privacy-preserving blockchain bridges. https://arxiv.org/abs/2102.04660, 2021. [Online].

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

[33] L. Wu, Y. Kortesniemi, D. Lagutin, and M. Pahlevan. The flexible interledger bridge design. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 69–72, 2021.

[34] A. Xiong, G. Liu, Q. Zhu, A. Jing, and S. W. Loke. A notary group-based cross-chain mechanism. *Digital Communications and Networks*, 2022.