Decentralized Interledger Gateway Architectures in Authorization Scenarios with Multiple Ledgers

Vasilios A. Siris, Michalis Tsenos, Dimitrios Dimopoulos, Nikos Fotiou, George C. Polyzos

Mobile Multimedia Laboratory, Department of Informatics
School of Information Sciences & Technology
Athens University of Economics and Business, Greece

***Abstract*—Combining multiple Distributed Ledger Technologies (DLTs) that include public and private/permissioned ledgers can allow different tradeoffs in terms of performance, cost, privacy, and transparency. However, multiple DLTs must be interconnected in a way that securely binds transactions on different ledgers and allows reliable and trusted transfer of information across the ledgers. We present decentralized interledger gateway architectures for IoT authorization scenarios that include the interconnection of two ledgers: an authorization ledger and a payment ledger. The proposed architectures differ in their complexity, transaction cost, and ability to handle transactions involving multiple ledgers.**

I. Introduction

Different types of blockchains or Distributed Ledger Technologies (DLTs) have different transaction cost, performance (transaction delay and throughput), trust, and privacy properties. Specifically, public blockchains such as Bitcoin and Ethereum are permissionless (anyone can participate as a blockchain node) and provide wide-scale decentralized trust and transparency. However, the public nature of these blockchains comes at the expense of high computation costs, hence high transaction fees, and high transaction delays. On the other hand, permissioned DLTs such as Hyperledger Fabric operate with a restricted set of peers, hence provide a lower degree of decentralized trust compared to public blockchains, but incur a smaller cost and delay. Moreover, permissioned DLTs can support different levels of write and read access, hence can be adapted to different privacy requirements. Combining multiple DLTs can allow different tradeoffs in terms of cost, performance, privacy, and transparency. However, multiple DLTs must be interconnected in a way that securely binds transactions on different ledgers and allows reliable and trusted transfer of information across the ledgers.

One use case where two types of DLTs can be combined is the payment and authorization for accessing IoT resources. One DLT can be the payment ledger where payments for resource access are performed. The second DLT can be an authorization ledger where authorization rules and decisions are recorded. The interledger functionality would be responsible for linking transactions on the two ledgers in a cryptographically secure way, and for transferring information from one ledger to another in a reliable and trustworthy manner [1]. The above use case that combines two ledgers, a payment ledger and an authorization ledger, is general and can be applied to any scenario where a payment is exchanged for accessing any type of service. The goal of the paper is to present decentralized architectures for an interledger gateway (ILG) system that ensures that the interledger operations are performed in a reliable and trusted manner. The contributions of the paper are the following:

- We present three decentralized interledger gateway architectures that have different features and involve different coordination of the interledger gateways.
- We present a preliminary evaluation that illustrates the reliability and performance of the proposed solutions.

Unlike the current paper, our previous work in [1] considered only a single interledger gateway system.

The remainder of the paper is structured as follows: In Section II we present the architecture involving a single interledger gateway. In Section III we present three decentralized interledger gateway architectures and in Section IV we present a preliminary evaluation of the proposed architectures. Finally, in Section V we present related work and in Section VI we conclude the paper and present ongoing work.

II. Single interledger gateway architecture

In a single interledger gateway architecture, Figure 1, a single interledger gateway (ILG) is responsible for performing the interledger functions. The simplest function involves copying data from one ledger to the other. If the data that is copied involves secrets to hash-locks on the two ledgers, then transactions on the two ledgers can be cryptographically linked in a dependence relation.

A hash-lock is a cryptographic lock that can be unlocked by revealing a secret s whose hash $H(s)$ is equal to the lock's value h. Unlocking a hash-lock can be one of the conditions for performing a transaction or for executing a smart contract function. Hash-locks can be used on two or more blockchains that support the same hash function, to link a transaction on one chain to one on the other chain: if the two transactions have hash-locks with the same value, then unlocking one would reveal the secret that unlocks the other; hence, the two transactions are cryptographically linked through a dependence relation. Time-locks are locks on a

This research was supported by the EU funded Horizon 2020 project SOFIE (Secure Open Federation for Internet Everywhere), under agreement No. 779984.

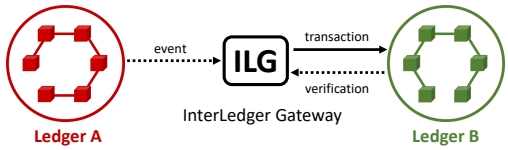

Fig. 1. In a single interledger gateway architecture an event notification from ledger A triggers the ILG to submit a transaction to ledger B. The ILG is a single point of failure, hence this architecture has low reliability.

blockchain that can be unlocked only after an interval has elapsed. The time interval can be measured in absolute time or in the number of blocks mined after a specific block. Contracts that include both hash and time-locks are referred to as hashed time-lock contracts (HTLCs) [2]. HTLCs can be implemented in blockchains with simple scripting capabilities, such as the Bitcoin blockchain, without requiring the advanced functionality of smart contracts.

The ILG, Figure 1, is responsible for i) listening to events generated on ledger A and ii) submitting transactions to ledger B. An event on ledger A can be generated when the secret to a hash-lock is submitted on this ledger. When the event on ledger A is generated, the gateway can obtain the secret and submit it to ledger B. Submission of the secret to ledger B is performed as a transaction using the account that the ILG has on ledger B. The verification that the gateway receives from ledger B can be a confirmation that the transaction was added to the transaction pool or a confirmation that the transaction was included in a mined block. In the later case, the verification can be received by having the ILG listen to an event on ledger B that is generated when the block containing the submitted transaction is mined.

Figure 2 shows the sequence of events in the payment and authorization for IoT resource access use case. An event on the authorization ledger is generated when the secret to the hash-lock is submitted on this ledger by an Authorization Server (AS). When the event on the authorization ledger is generated, the gateway can obtain the secret and submit it to the payment ledger, which triggers the transfer of the deposit made by the client to the IoT resource owner's account.

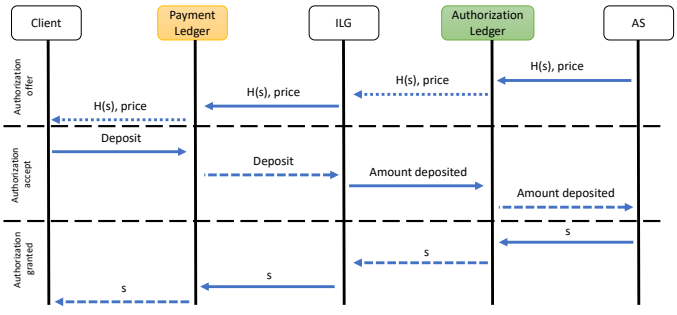

Fig. 2. Sequence diagram in the payment and authorization for IoT resource access use case. The ILG is responsible for listening to events generated on one ledger and submitting transactions to the other ledger. This includes the event triggered by the submission of the hash-lock secret to the authorization ledger and the submission of the secret to the payment ledger.

III. Decentralized Interledger Gateway Architectures

In this section we present three decentralized interledger gateway architectures. The first involves multiple ILGs that are operated by the same organization and the ILGs use the same account for submitting transactions to a ledger. The second architecture involves multiple ILGs that are operated by different organizations and the ILGs use different accounts for submitting transactions to a ledger. Finally, the third architecture utilizes a Hyperledger Fabric permissioned ledger for coordinating the interledger functions.

A. Multiple interledger gateways operated by the same organization

In this model, multiple ILGs that are operated by the same organization, Figure 3, perform the interledger functions that include the following:

- Listen to events on ledger A indicating that the secret for a hash-lock is recorded on ledger A. These events will trigger the interledger operations.
- All ILGs submit transactions to ledger B using the same account.
- All ILGs listen to an event on ledger B to confirm that the transaction has been successfully performed.

Every transaction in Ethereum has a nonce, which corresponds to the number of transactions that are sent from a given account. Each time a transaction is submitted, the nonce value increases by one. Furthermore, transactions from the same account must be ordered according to their nonce values and nonce values cannot be skipped. Because the same account is used by all ILGs for submitting transactions to ledger B, if ledger B is an Ethereum blockchain, synchronization of the nonce values is necessary to satisfy the above rules. Specifically, the ILGs need to use the same nonce in transactions that correspond to the same event (e.g. recording of the same secret) on ledger A. Synchronization of the nonce values can be achieved in a centralized manner by the organization managing the ILGs. Because the ILGs use the same account, even though N transactions are submitted to ledger B, only one will be included in the mined block; hence, the transaction fee will be incurred once. Based on the above, this solution provides decentralization of the interledger operations, but does not provide decentralization at the organization level since the ILGs are managed by a single organization. The architecture presented in the Section III-B provides decentralization of both the interledger functions and the management of the ILGs.

1) Management of ILGs: The management of ILGs is performed by a single organization and includes managing the nonce value that the ILGs include in the transactions they submit to ledger B using the same account. The organization managing the ILGs must be trusted in order to achieve reliable interledger gateway operation. Note, however, that in the authorization scenario discussed in this paper the interledger operation involves copying a secret from ledger A to ledger B. Copying the correct secret is ensured since ledger B will

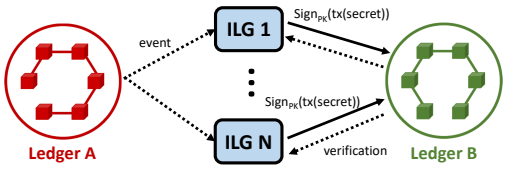

Fig. 3. Decentralized interledger gateway architecture when the ILGs are operated by the same organization. When they receive an event notification from ledger A, all ILGs submit a transaction to ledger B using the same account (which corresponds to a public key PK).

check that the submitted secret's hash is equal to the hash of the hash-lock in the smart contract on ledger B.

Instead of using the same account on ledger B, each ILG can use a different account. With this approach, synchronization of the nonce values if ledger B is an Ethereum network is not necessary. Moreover, only one (or a few) ILGs can be selected in a centralized manner to submit transactions to ledger B, making this scheme more efficient from this perspective.

2) Reliability: Reliability is achieved by having all ILGs submit a transaction to ledger B, once they receive an event notification from ledger A. Moreover, after they submit a transaction to ledger B, all ILGs listen to an event on ledger B that confirms that the transaction with the hash-lock's secret is included in a mined block of ledger B. In this way, ILGs can verify that the transaction is successfully submitted to ledger B. If the ILGs do not receive an event verifying the submission of the transaction until some timeout, they resubmit the transaction to ledger B.

B. Multiple interledger gateways operated by different organizations

The decentralized interledger gateway architecture when the ILGs are operated by different organizations is shown in Figure 4. In this architecture, the interledger functions that need to be supported include the following:

- Listen to events on ledger A indicating that the secret for a hash-lock is recorded on ledger A. These events will trigger the interledger operations.
- Selection of one (or more) ILG(s) that will submit transaction(s) to ledger B. These transactions contain the secret copied from ledger A.
- The selected ILG (or ILGs) submits the transaction to ledger B.
- The ILGs verify that the transaction was successfully submitted to ledger B. If the submission is not verified after some timeout, a new ILG (or ILGs) is selected to submit the transaction.

In this model, N ILGs listen for an event on ledger A indicating that the hash-lock's secret has been submitted, Figure 4. The ILGs are operated by different organizations and use different accounts to submit transactions to ledger B. Because the ILGs use different accounts, synchronization of the nonce values if ledger B is an Ethereum network is not necessary.

Once the event notification is received by the ILGs, one of them (ILG k in Figure 4) is selected to submit a transaction to ledger B containing the secret recorded on ledger A. The selection can be made by taking the modulo N of the last or

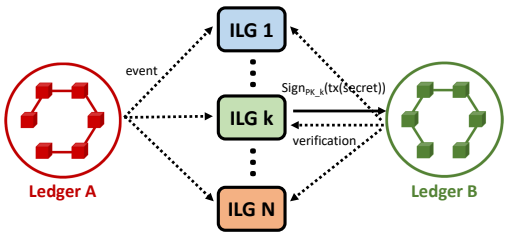

Fig. 4. Decentralized interledger gateway architecture where the ILGs are operated by different organizations. One of the N ILGs (ILG k in the figure) is selected to submit the transaction to ledger B. The selection is based on the hash of the last or the next block mined on ledger B. ILGs use different accounts to submit transactions to ledger B.

the next block on ledger B. Specifically, each of the N ILG is assigned a unique identifier from 0 to N-1 and the result of the ledger B's block hash modulo N operation determines the ILG that is responsible for submitting the transaction to ledger B. The hash of ledger B's block must be used because ledger B must also verify that the correct ILG submitted the transaction. In Section III-B1 we discuss alternatives for managing the bindings of ILGs to unique identifiers.

Next we discuss the implications from using the hash of the last or the next mined block of ledger B for determining the ILG that is responsible for submitting the transaction to ledger B. If the hash of the last block is used, then anyone can know which of the ILGs is responsible for submitting the transaction to ledger B that corresponds to the next event on ledger A. This allows the possibility of a DoS attack to the ILG responsible for sending the transaction *before* the event on ledger A occurs. On the other hand, if the ILG selection uses the hash of the next block that is mined, then the ILG that is responsible for submitting the transaction on ledger B is known *after* the next block on ledger B is mined. Assuming that the selected ILG submits a transaction *immediately after the next block is mined*, the window for conducting a DoS attack on this ILG can be very small. Finally, we note that the generation of new blocks is determined by miners based on the PoW in blockchains such as Ethereum and Bitcoin. As long as the fees for interledger services is much lower than the block mining fees, the approach described above is not susceptible to attacks by miners withholding new blocks.[1]

Because the N ILGs are operated by different organizations, their trustworthy operation must be ensured, i.e. the system must ensure that an ILG must not submit a transaction to ledger B if it is not entitled to do so. Two observations can be made regarding this issue: First, even if more than one ILGs submit transactions with the hash-lock secret to ledger B, the interledger operation is still executed correctly. Moreover, if ledger B is a public blockchain such as Ethereum, multiple transactions would incur an execution cost for the ILGs that submit the transactions. Second, checking that the transaction is submitted by the ILG that is entitled to do so is necessary if the ILGs receive some compensation for their services. To

[1]The requirement that the economic gains from withholding blocks is lower than the mining fees does not hold for high-stake lottery applications, hence mined block hash approaches for generating random numbers are not appropriate in such applications.

achieve this, the smart contract on ledger B must verify that the ILG submitting the transaction is the one that was entitled to do so; the smart contract can perform this verification based on the hash of the last block mined on ledger B and the ILG's ID. Moreover, once the smart contract verifies that the transaction was submitted by the correct ILG, it can proceed to automatically reimburse that ILG for its interledger services.

1) Management of ILGs: Next we discuss how the ILGs that perform the interledger transactions can be managed. In the scenario described in the introduction that includes linking payments to authorizations for accessing IoT resources, the two parties involved are the client (buyer) and the IoT resource owner (seller). The scenario requires that smart contracts containing hash-locks and time-locks are created on the two ledgers: the payment ledger and the authorization ledger. The interledger services for obtaining authorization to an IoT resource can be managed by the IoT resource owner, which is the interledger services client. This IoT resource owner can determine the ILGs that can provide interledger services. One approach to achieve this is to submit to the smart contract on ledger B the list of ILGs that can submit transactions for interledger services related to the owner's IoT resource. Alternatively, the ILGs can obtain a signed credential from the owner that allows them to submit transactions to the smart contract on ledger B. Such credentials can be based on the Verifiable Credentials Model developed by W3C's Credentials Community Group [3]. The credential can be submitted inside each transaction and the smart contract on ledger B can verify the IoT resource owner's signature to ensure that the ILG submitting the transaction has authorization from the owner. The two options for determining the set of ILGs that can submit interledger transactions differ in terms of privacy and the operations and cost for performing revocation. Namely, recording the set of legitimate ILGs in the smart contract allows all entities that have read access on the ledger, which in the case of public ledgers is anyone, to know the ILGs that provide interledger services for a particular IoT resource. Revocation, i.e. removing an ILG from the list of ILGs, would incur the cost of a transaction to update the list in the smart contract. On the other hand, using credentials has higher privacy, since in this case the smart contract does not contain the list of ILGs. However, the credential-based approach requires different actions to perform revocation. One approach is to have credential with fixed time validity, which implies that they must be periodically renewed for an ILG to continue to provide interledger services.

An additional management task is the assignment of unique numbers 0 to N-1, which is necessary for the ILG selection discussed above. The assignment needs be known, in a reliable manner, by the smart contract running on ledger B. This can be achieved by including the number assigned to each ILG in the list maintained by the smart contract or include the number in the credential that the interledger services client provides to each ILG.

2) Reliability: Next we discuss how this architecture achieves reliability. After an event is generated on ledger A, all ILGs, including the ILG that was selected to submit the transaction to ledger B, listen to an event on ledger B that verifies that the transaction with the hash-lock's secret is included in a mined block of ledger B. In this way, ILGs can verify that the transaction is successfully submitted to ledger B. If the ILGs do not receive an event verifying the submission of the transaction until some timeout, then a new ILG can be selected, using the same procedure as the one described above, which will (re)submit the transaction to ledger B.

Above we have assumed that one ILG is selected for submitting the transaction to ledger B. Alternatively, more than one ILGs can be selected and submit transactions from their accounts. Such an approach can yield a smaller delay for completing the interledger operations, if the probability of faulty or misbehaving ILGs is high. The tradeoff is that if more than one ILGs are selected then, in the case of the Ethereum blockchain, the total transaction cost would increase with the number of submitted transactions. One way to reduce this cost in Ethereum is for the smart contract to use Solidity's Revert call, with which duplicate transactions incur only the cost (gas) of the transaction invocation.

C. Use of Hyperledger Fabric in the interledger gateway system

Next we discuss the use of Hyperledger Fabric, which is a permissioned distributed ledger that is part of Linux Foundation's open-source Hyperledger project [4], for coordinating the interledger operations. The advantages of using a permissioned ledger for interledger operations are the following:

- A permissioned ledger can implement elaborate consensus logic and rules that can jointly consider events from different ledgers. This is necessary if the interledger operations involve more than copying a hash-lock secret between two ledgers.
- A permissioned ledger can record, in a reliable and immutable manner, transactions across different ledgers.
- Even though some of the logic can be implemented in ledger B (provided it supports smart contracts), the execution cost when the logic is implemented in a permissioned ledger will be significantly lower compared to the cost if ledger B is a public ledger.

The aforementioned advantages allow a permissioned ledger such as Hyperledger Fabric to serve as a hub for interledger services among multiple ledgers. The above advantages are achieved with a higher complexity compared to the two previous architectures, since Fabric is used for coordinating the functionality of the interledger mechanisms.

Figure 5 shows the architecture for an interledger system that is based on Hyperledger Fabric. Observe that the functionality of the interledger gateways of the first two architectures shown in Figures 3 and 4 has been separated. The architecture shown in Figure 5 has two types of interledger gateways: level 1 interledger gateways listen to events on Ledger A and level 2 interledger gateways submit transactions to ledger B. Hyperledger Fabric can implement elaborate consensus rules that can include selecting one or more level 2 ILGs that are

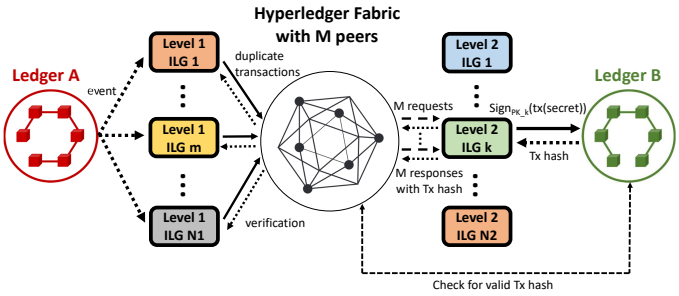

Fig. 5. Use of Hyperledger Fabric in the interledger gateway system. Compared to the architectures in Figures 3 and 4, the interledger functionality is now split: Level 1 ILGs listen to events on ledger A and level 2 ILGs submit transactions to ledger B.

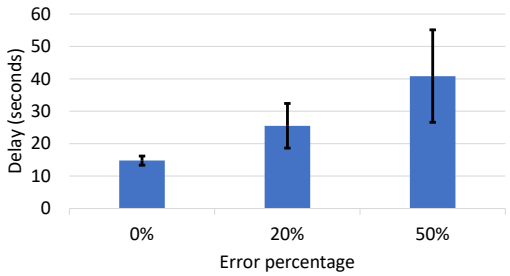

Fig. 6. Interledger delay for the architecture in Figure 4 with three ILGs.

responsible for submitting transactions to ledger B. From an implementation perspective, level 2 ILGs can run on the same nodes where Fabric peers reside. Similarly, level 1 ILGs can also reside on Fabric peer nodes.

1) Management of ILGs: In this architecture the management of ILGs can be performed as discussed in Section III-B1. A difference compared to the architecture in Section III-B is that the Fabric peers now select the level 2 ILG that will be responsible for submitting the transaction to ledger B. In this step we utilize the capability of Fabric peers that all execute the same smart contract (called chaincode in Fabric) to call an external API. Hence, in Figure 5 the selected level 2 ILG receives a call from all M Fabric peer nodes.

2) Reliability: The reliability of the functionality of level 1 ILGs, namely of sending a transaction to the Fabric network when an event occurs on ledger A, is ensured by having all level 1 ILGs send a transaction to Fabric (duplicate transactions in Figure 5). Fabric can handle multiple concurrent transactions, but cannot handle concurrent transactions that affect the same key in the ledger. Hence, only one transaction from Level 1 ILGs will eventually be valid and the other will cause a Multi Version Concurrency Control (MVCC) failure. Nevertheless, only one valid transaction is necessary for the interledger functionality to operate correctly, hence the above behavior is sufficient. Note that, since this paper focuses on interledger functions that involve copying hash-lock secrets from one ledger to another, it is not necessary for the Fabric network to verify the transaction on ledger A that reveals the hash-lock secret; if the secret was not revealed on ledger A, then the corresponding transaction on ledger B would fail.

The reliability of submitting transactions to ledger B is achieved different than the way discussed in Section III-B2, since in this model the Fabric peers select the ILG that will submit the transaction to ledger B. Specifically, recall that all Fabric peers select the same level 2 ILG to submit the transaction to ledger B, hence the selected level 2 ILG receives a call from all M Fabric peer nodes. Once this level 2 ILG submits the transaction to ledger B, it returns the transaction hash (Tx hash in Figure 5) to the Fabric peer nodes. The Fabric peer nodes check ledger B to verify that a valid transaction with the transaction hash (Tx hash) has indeed appeared in a mined block. If the transaction is not verified, then the Fabric peer nodes select a new level 2 ILG to submit the transaction. The Fabric peer nodes also select a new level 2 ILG to submit the transaction to ledger B if the original level 2 ILG does not respond after some timeout.

Finally, as discussed in Section III-B2, more than one level 2 ILGs can be selected for transmitted transactions to ledger B. This can increase the reliability and reduce the delay, at the expense of a higher transaction cost on ledger B.

IV. Evaluation

The evaluation results presented in this section consider the architecture shown in Figure 4. The evaluation setup consisted of the Rinkeby public Ethereum testnet[2] that was ledger B in Figure 4. We used the Infura Ethereum node cluster[3] for submitting transactions to the Rinkeby. For ledger A in Figure 4 we used Hyperledger Fabric.

We consider the case of 3 ILGs and investigate the reliability of the interledger functionality that the architecture shown in Figure 4 can support in the presence of ILG errors. Figure 6 shows the interledger delay for different ILG error percentages. The results are the average from 20 executions. Also shown are the 95% confidence intervals. The interledger delay is the time interval from the point an event on ledger A (Hyperledger Fabric) is generated until the time that a transaction is successfully submitted to ledger B (Ethereum). Since ledger B is an Ethereum network, the interledger delay is determined by the Ethereum transaction delay, which depends on the Ethereum block mining time. Hence, for zero errors, Figure 6 shows that the delay is approximately 15 seconds, which is the average time for mining a new Ethereum block. In the presence of errors, Figure 6 shows that the delay increases by approximately 70% and 270% when the percentage of ILG errors is 20% and 50%, respectively. The transaction delay in the presence of ILG errors can be reduced by having more than one ILGs submit interledger transactions to ledger B.

The delay results shown in Figure 6 also pertain to the architecture in Figure 3, since the interledger delay is determined by the transaction delay on ledger B and the differences of the interledger operations in Figure 3 and Figure 4 do not influence this delay.

Table I shows the gas, which quantifies the amount of EVM (Ethereum Virtual Machine) resources, for the transaction on ledger B (Ethereum testnet) that the ILG submits for three

[2]https://www.rinkeby.io/
[3]https://infura.io

TABLE I
LEDGER B (ETHEREUM) TRANSACTION COST (GAS)

Transaction	Gas
Architecture in Figure 3	64 425
Architecture in Figure 4	66 968
Architecture in Figure 4 with revert	27 132

cases: i) the architecture in Figure 3, ii) the architecture in Figure 4, and iii) the architecture in Figure 4 when a revert is called in the transaction. The transactions are submitted with gas price 2.5 Gwei. The results show that the transaction cost for the architecture in Figure 4 is higher than the architecture in Figure 3. This occurs because of the additional checks in the latter architecture that is necessary to ensure that the correct ILG has submitted the transaction. Specifically, we assume that the smart contract has a table containing the address of the ILGs that can submit an interledger transaction and their corresponding identifiers. The specific ILG that should submit the transaction is selected as described in Section III-B, and is based on the modulo 4 (since the number of ILGs is 3) of the hash of the last block on ledger B. Table I shows that the transaction cost with a revert is less than 40% of the transaction cost without a revert (for the architecture in Figure 4). The revert can be used when more than one ILGs are selected to submit transactions to ledger B, in order to increase the reliability and reduce the delay of the interledger operation.

V. RELATED WORK

Next we present a brief overview of related proposals for interconnecting two (or more) blockchains. A more extended discussion can be found in [5]. Blocknet supports the decentralized exchange of cryptocurrency between blockchains [6]. The interledger services are implemented by nodes interconnected through a DHT-based peer-to-peer network. Trust among the nodes is achived through a Proof-of-Stake consensus algorithm. ARK provides interledger bridging functionality similar to Blocknet [7]. In addition to the exchange of tokens, ARK also supports the execution of service contracts, which can include the transfer of data, the creation of smart contracts, and the execution of code on different blockchain platforms. The nodes of the ARK network implement a Delegated Proof-of-Stake (DPoS) consensus algorithm.

BTC Relay[4], which was initiated by the Ethereum Foundation, is a smart contract on Ethereum that can read the Bitcoin chain and verify Bitcoin transactions. This allows using Bitcoin payments for executing Ethereum smart contracts. BTC Relay uses Bitcoin block headers to build a "mini-version" of the Bitcoin blockchain. When an application processes a Bitcoin payment, it uses a header to verify that the payment is legitimate. Relayers are those who submit block headers to BTC Relay. When any transaction is verified in the block, or the header is retrieved, relayers are rewarded with a fee.

The POA Network[5] utilizes Proof-of-Authority (PoA) as its consensus mechanism and is another attempt for developing a cross-chain bridge solution for connecting Ethereum-compatible blockchains. Aion is a proposal that has common features to the proposals discussed above. Namely, inter-chain transactions are performed by bridges, which implement a lightweight BFT-based consensus algorithm and receive inter-chain transaction fees.

Polkadot [8] proposes a scalable heterogeneous "multi-chain", upon which a large number of so-called *parachains* can be built. A relay-chain is responsible for finalizing all the transactions between parachains. Cosmos is similar in structure to Polkadot and proposes a Hub as the main blockchain that interconnects many other independent parallel blockchains, called zones [9]. The Cosmos hub and zones can implement a classical BFT consensus algorithm.

The primary focus of all the above proposals is the interconnection of different blockchains to allow the exchange of tokens. On the other hand, the work in this paper focuses on the interconnection of transactions on different blockchains to link payment and IoT resource access authorization transactions. In this direction, we investigate solutions, namely the first two architectures shown in Figures 3 and 4, which are simpler than the proposals discussed above.

VI. CONCLUSIONS AND FUTURE WORK

We have presented decentralized interledger gateway architectures for IoT authorization scenarios that include the interconnection of two ledgers. The proposed architectures differ in their complexity, transaction cost, and ability in handling transactions involving multiple ledgers. Ongoing work focuses on further evaluating the proposed architectures, and in particular on quantifying the tradeoffs between reduced delay and increased transaction cost when multiple ILGs submit interledger transactions.

[4]http://btc-relay.readthedocs.io

[5]https://poa.network

REFERENCES

[1] V. A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, and G. C. Polyzos, "Decentralized authorization in constrained IoT environments exploiting interledger mechanisms," *Computer Communications*, no. 152, pp. 243–251, 02 2020.

[2] Bitcoin Wiki, "Hashed Timelock Contracts (HTLC)," https://en.bitcoinwiki.org/wiki/Hashed_Timelock_Contracts.

[3] M. Sporny *et al.*, "Verifiable Credentials Data Model 1.0: Expressing verifiable information on the Web," Draft Community Group Report, W3C, September 05, 2019.

[4] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. of ACM EuroSys Conference*, 2018.

[5] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger Approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, 2019.

[6] A. Culwick and D. Metcalf, "The Blocknet design specification." [Online]. Available: https://www.blocknet.co/wp-content/uploads/2018/04/whitepaper.pdf

[7] ARK, "ARK Ecosystem Whitepaper. Version 2.0.0," April 1, 2019. [Online]. Available: https://ark.io/Whitepaper.pdf

[8] L. Wood, "Polkadot: Vision for a heterogenous multi-chain framework, white paper," November 2016. [Online]. Available: https://polkadot.network/PolkaDotPaper.pdf

[9] Cosmos Whitepaper. [Online]. Available: https://cosmos.network/resources/whitepaper