Application-Level Interoperability for Blockchain Networks

Mohammad Madine, Khaled Salah, Raja Jayaraman, Yousof Al-Hammadi, Junaid Arshad, Ibrar Yaqoob

***Abstract*—Blockchain technology has the potential to revolutionize industries by offering decentralized, transparent, data provenance, auditable, reliable, and trustworthy features. However, cross-chain interoperability is one of the crucial challenges preventing widespread adoption of blockchain applications. Cross-chain interoperability represents the ability for one blockchain network to interact and share data with another blockchain network. Contemporary cross-chain interoperability solutions are centralized and require re-engineering of the core blockchain stack to enable inter-communication and data sharing among heterogeneous blockchain networks. In this paper, we propose an application-based cross-chain interoperability solution that allows blockchain networks of any architecture type and industrial focus to inter-communicate, share data, and make requests. Our solution utilizes the decentralized applications as a distributed translation layer that is capable of communicating and understanding multiple blockchain networks, thereby delegating requests and parameters among them. The architecture uses incentivized verifier nodes that maintain the integrity of shared data facilitating them to be readable by the entities of their network. We define and describe the roles and requirements of major entities of inter-operating blockchain networks in the context of healthcare. We present a detailed explanation of the sequence of interactions needed to share an Electronic Medical Record (EMR) document from one blockchain network to another along with the required algorithms. We implement the proposed solution with Ethereum-based smart contracts for two hospitals and also present cost and security analysis for the cross-chain interoperability solution. We make our smart contracts code and testing scripts publicly available.**

***Index Terms*—Blockchain; Interoperability; Cross-chain Interoperability; Decentralized Application; Ethereum; Healthcare**

I. Introduction

Blockchain is a prominent Distributed Ledger Technology (DLT) that is inherently decentralized, transparent, and auditable, thereby enabling more trustworthy and reliable services. The adoption of blockchain in various fields can bring disintermediation, delay reductions, fraud and error reductions, and complete provenance of decisions and events. However, as diverse applications can have varying requirements, organizations and developers need to choose the optimal blockchain architecture, such as public, private, or consortium. As highlighted by a recent study [1], blockchain has been adopted across diverse domains including finance and insurance, accommodation and food services, and healthcare and social assistance. Interestingly, the study found indications for potential interest in interoperability, as 9% of sectors are cross-industry, 88% have shared use of the blockchain network between different entities, and 73% plan to increase their collaboration with new partnerships.

Although blockchain introduces many benefits as highlighted above, contemporary implementation of blockchain-based systems can potentially create silos within organizations. Therefore, a major challenge impeding the widespread adoption of this technology is the lack of interoperability among different blockchains. To this end, cross-blockchain interoperability is envisaged to allow different blockchain networks to interact with each other and future blockchains without having to embed a pre-defined intercommunication layer in each network. As blockchain is adopted by an increasing number of organizations and enterprises, there is a race to research and develop appropriate standards for cross-blockchain interoperability [2]. Specifically, the drivers to the development of mechanisms for blockchain interoperability are primarily rooted in scalability and integration. Homogeneous blockchain solutions that serve the same purpose, each for a specific region, can benefit from interoperability to scale-out to new stakeholders. Similarly, heterogeneous blockchain solutions can take advantage of a potential interoperability standard through integration procedure.

An important factor in the lack of cross-chain interoperability can be attributed to two fundamental aspects in modern blockchains i.e. 1) smart contracts, and 2) fragmentation. Blockchain uses smart contracts to automate interactions between stakeholders of a network. Since smart contracts are stored on the immutable ledger on the chain, they cannot be upgraded, and therefore even if a group of blockchain systems is manually integrated, they cannot support future solutions without a total re-write of the smart contract. Furthermore, as a result of the rapid development within blockchain technology, there is a fragmentation in the types of technologies blockchain systems use, such as the architecture type, the development framework, and the consensus algorithm. Developing standards to facilitate interoperability among existing variations of blockchain, in addition to future possible ones, is a great challenge.

Mohammad Madine, Khaled Salah, Yousof Al-Hammadi, and Ibrar Yaqoob are with the Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi 127788, United Arab Emirates.

Raja Jayaraman is with the Department of Industrial and Systems Engineering, Khalifa University of Science and Technology, Abu Dhabi 127788, United Arab Emirates.

Junaid Arshad is with the School of Computing and Digital Technology, Birmingham City University, Birmingham B4 7XG, U.K.

Corresponding author: ibraryaqoob@ieee.org

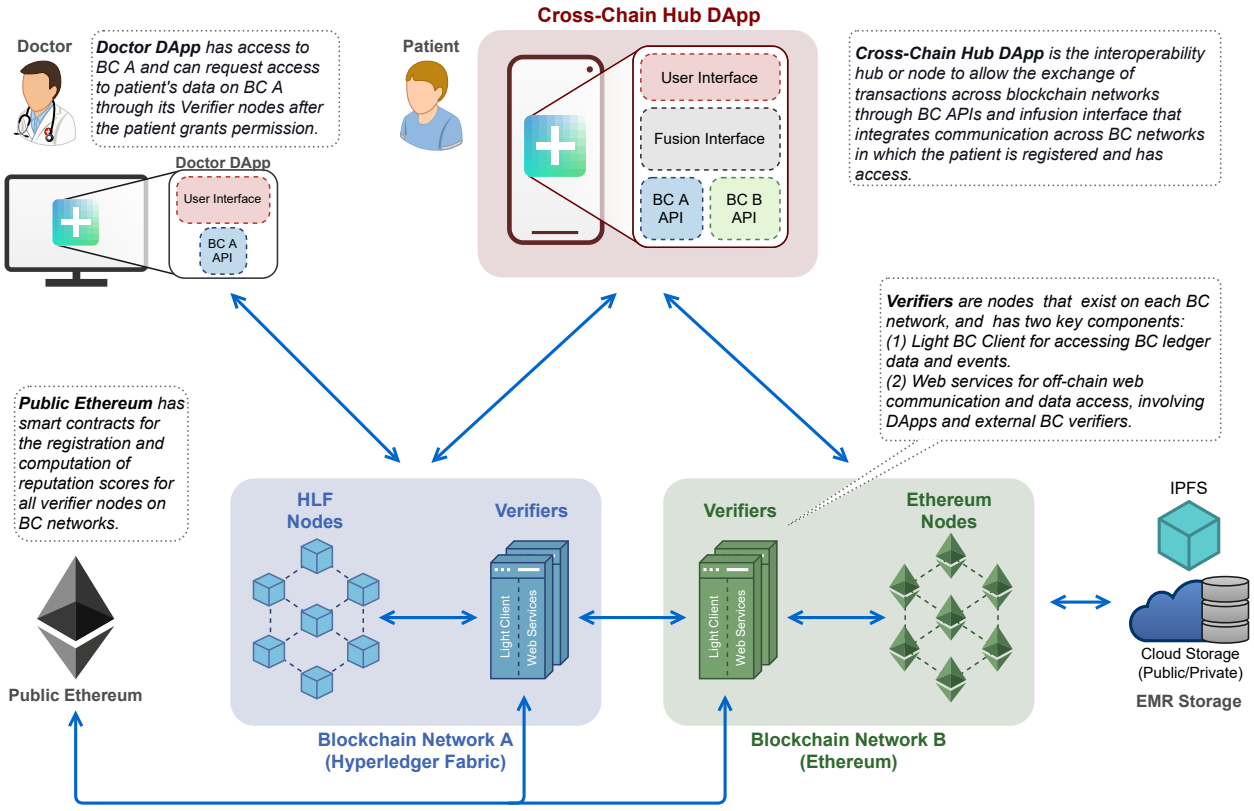

Figure 1: An overview of (a) current centralized solutions, and (b) our proposed decentralized solution

A. *Blockchain interoperability goals*

A basic blockchain interoperability solution is envisaged to allow heterogeneous blockchain-based systems to interact and share data; however, for the solution to be practical, it must achieve the following goals.

- The solution architecture must support blockchain systems of various types, such as public or private architectures, and different platforms (Ethereum, Hyperledger Fabric, etc.).
- The solution must not cause a major disruption to the blockchain network, such as by requiring repeated forking or smart contract modification for every new intercommunication link with a network.
- No manual interference by the end-users should be required.
- The dependence on off-chain infrastructures and systems shall be kept to a minimum, and in the case of usage of any off-chain approach, the off-chain entities cannot be naively trusted with their responses.
- The solution must not has an adverse impact on the performance or the security of the blockchain networks.

B. *Approach and Contributions*

This paper aims to design and develop an architecture for cross-chain interoperability that can be adopted in a wide range of applications and use cases. Our proposed solution allows data sharing and interaction across different blockchain systems with the ability to provide interoperability support for additional systems in a seamless manner. In particular, we propose application-level interoperability for blockchain networks, taking advantage of the adaptability and upgradability of Decentralized Applications (DApp) to develop a practical and standardized solution for cross-chain communication. Our proposed solution is summarized in Fig. 1.

To assess the effectiveness of the proposed solution, we use a blockchain-based healthcare system as a use case. The healthcare industry has greatly benefited from blockchain adoption and can potentially benefit even further through scaling out and utilizing data sharing opportunities achieved by cross-chain interoperability. In this respect, [3] showcases the significant impact of blockchain in healthcare, such as through Electronic Health Records (EHR) to be securely stored and accessed in a distributed and decentralized manner. Moreover, the authors also highlight how the lack of interoperability can be a barrier to blockchain adoption in the healthcare sector.

Blockchain systems for managing EHR data and giving patients control over their data typically have a private architecture. The systems are regulated by a leading entity, such as the hospital or the department of health, which is responsible for deploying the system smart contracts and verifying the identity of the stakeholders, such as the patients and the doctors.

Major contributions of this paper can be summarized as follows:

- We propose a blockchain interoperability solution based

on decentralized applications, which facilitates cross-blockchain communication, data sharing, and interaction with no end-user intervention.
- We design and develop a standardized system architecture for interoperable blockchain networks that is fully-automated, secure, trustworthy, and platform-independent.
- We adapt the interoperability standard to a healthcare use case and incorporate the proposed solution with a detailed blockchain-based patient-centered EHR management system.
- We implement functional smart contracts of hospital-regulated healthcare systems, deploy the smart contracts, and perform extensive experiments and tests to verify the correctness of our algorithms. The smart contract code along with testing scripts is publicly made available[1].
- We analyze the cost and performance of our proposed solution, and assess the security of blockchain networks adopting our interoperability standard.

The remainder of the paper is organized as follows. Section II presents related work. Section III describes the individual system components and the overall architecture and data flow. In Section IV, we present algorithms and functions that we later implement for deep solution analysis. In Section V, we evaluate the proposed approach in terms of correctness, cost, and security. We summarize our findings in Section VI.

II. Related Work

One of the earliest and most prominent works on blockchain interoperability is [4]. Vitalik Buterin is the founder of the blockchain Ethereum framework and defined three strategies for approaching interoperability in this paper. The first is called a notary scheme, in which a trustworthy set of entities allow atomic interaction and information sharing across multiple blockchains, acting as intermediaries. The second strategy is referred to as relay, which requests one of the blockchains to be responsible for verifying the claims and information of another blockchain. Finally, the third proposed strategy is hash-locking, which inter-locks multiple operations on different blockchains using the original message of a hash, thus ensuring all interactions refer to the same initial request by the end-user in a verifiable manner. Although the suggested strategies can provide interoperability in certain use cases, practically they fall short of being a standard that stimulates scalability and maintains security of the network. For example, in the notary scheme strategy, the intermediaries must be blindly trusted by all blockchains; whereas, in the relay strategy the blockchain responsible for verifying information and transactions can be seen as a point of centralization.

Over the few years since the emergence of Ethereum and Hyperledger, further research works have been dedicated to devising an interoperability standard that satisfies a wide range of networks. The study conducted in [5] describes the need for a generic inter-blockchain communication protocol that can exchange arbitrary data, such as tokens and smart contract interactions, between blockchains in a decentralized and trustless manner. On the other hand, the researchers in [6] claimed that interoperability is impossible under the classical definition of blockchain, and thus require other mechanisms, such as game theory to prevent an interface from misbehaving.

Hashed Time-Locked Contracts (HTLC) are smart contracts that provide cross-blockchain atomic transactions by ensuring that two locked transactions are either executed or canceled after a predefined timeout [7]. Further, [8] proposes a scalable and secure HTLC that uses multi-hop locks. Disadvantages of the HTLC approach include having to provide long timeout periods during which an adversary can decide whether or not to execute their transaction based on an updated state of the networks, and allowing the high cost and complexity of the design as it requires each blockchain network to understand the hash, attributes, and smart contracts of the other network [9]. In [10], the authors introduced Interledger Protocol (ILP) as a combination of two of the strategies laid out by [4]. The ILP adopts a notary scheme and hash-locking strategies to keep the intermediary entities trustless and incentivized in the payment system. Although the ILP can be expanded beyond payment-focused applications, it requires understanding between the parties of the networks and demands high costs for cryptographic operations [11], [12].

Cosmos and Polkadot are unique solutions that bring interoperability to the blockchain. The solutions share the same fundamental concept of creating an interconnected network of blockchains that can understand each other [13], [14]. As it can be expected from the approach, blockchains are required to be built specifically on top of the Cosmos or Polkadot network, limiting the interoperability feature to a unique number of projects, and demanding additional skill sets from the enterprise developers.

Among the blockchain interoperability efforts, researchers have developed properties and requirements that are sought-after in the healthcare industry. The study conducted in [15] suggests that DApps must have evolvability and minimal integration complexity to address blockchain-based healthcare interoperability, and concludes that an interoperable solution must have a flexible design, use minimal resource duplication, and scalability. Additionally, the authors in [16] defined healthcare interoperability as exchanging information across multiple systems at three levels:

- Foundational, in which receiving party does not have to interpret data.
- Structural, in which data uses predefined formats.
- Semantic, in which there are syntax and semantic data exchange and interpretation.

Within blockchain-based healthcare applications, [17] proposes an interoperable OpenPharma blockchain solution, which uses public Ethereum blockchain as an interoperability layer between existing vendor solutions. The blockchain is used to store encrypted patient information URLs and share them with doctors who obtain the patient's encrypted authentication ID. The solution proposed is not suitable for interoperability across multiple patient-centered solutions. Patients must share an inexhaustible encrypted authentication ID with their doctors, which if stolen, can be used by any other member.

[1] https://github.com/anon18012021/blockchain-interoperability

Moreover, the architecture relies on centralized services, such as SAAVHA and Amazon Web Services - Key Management Service (AWS-KMS) for membership authentication and encryption [18].

To the best of our knowledge, none of the aforementioned cross-chain interoperability solutions are capable in providing interaction and data sharing across framework-independent and domain-neutral blockchain networks with no user intervention and point of centralization. The earliest category of research, such as HTLC and multi-hop locks, mainly introduced a solution for guaranteeing the integrity of transactions, but do not provide a wide range of capabilities and use cases. On the other hand, more recent and developed research can offer secure and versatile solutions, but at the cost of high complexity and discouraging requirements for enterprises and high-scale projects.

III. DApp-Based Cross-Blockchain Interoperability Solution

In this section, we present our proposed solution for cross-chain interoperability. To achieve a real-world context, we explain our system from the perspective of healthcare and patient medical records management. The section includes a description of the major elements of the system, the overall architecture, along with detailed examples of the sequence of interactions in the system.

A. Blockchain

The characteristics of a blockchain network depend on its specific platform, which can vary depending on the requirements of the application domain. Historically, blockchain-based healthcare applications have been developed as a fully decentralized public Ethereum blockchain with pseudonym patients, or as a hospital-regulated private HLF or Quorum blockchain with identifiable patients [19]–[21].

1) Ethereum and Smart Contracts: Along with the core blockchain ledger, Ethereum provides a programmable interface to the ledger through the use of smart contracts. Ethereum operates on the Ether cryptocurrency, which can be used for asset transfer through tokens and paying public Ethereum Virtual Machine (EVM) nodes for the execution of programs in smart contracts [19]. The complexity of Ethereum smart contracts is measured in gas units valued based on wei units, where 1wei = e-18Ether.

2) Hyperledger Fabric: Hyperledger Fabric is a project initiated by the Linux Foundation as a permissioned, programmable, and confidential blockchain framework. HLF, unlike Bitcoin and Ethereum, does not operate on a cryptocurrency, nor does it depend on Proof Of Work (PoW) for consensus. A typical HLF network has 3 components:

Certificate authority, to register identities, and issue and revoke enrollment certificates.

Peer, an endorser or committer, to update or query ledgers.

Orderer, to provide the order of transactions, create blocks, and process transactions before committers.

The consensus in HLF can be decided by the network regulator or reached with an agreement protocol, such as Practical Byzantine Fault Tolerance (PBFT) where two or more parties agree on the result [22].

B. Verifier Nodes

Verifiers are nodes registered by the blockchain network that act as lightweight clients for accessing blockchain ledger data and events. Additionally, verifier nodes run a web service for off-chain web communication and data access. Moreover, depending on the design of the EMR storage, the verifier may need to process the patient medical records to translate them from the patient to the doctor. For instance, a patient-centered blockchain system may require Proxy Re-encryption (PRE) processing of the EMR to convert the records from being encrypted by the public key of the patient to being encrypted by the public key of the doctor [23].

C. Reputation Systems

Considering that verifier nodes are not necessarily trusted computing devices, their actions must be regulated to incentivize appropriate behavior and discourage malicious behavior. For secure interactions and message transfer across the blockchains, our architecture requires internal and external reputation systems.

1) Internal Reputation System: Blockchain networks that use public verifier nodes must implement an internal reputation system as means to encourage correct and quick results by the verifiers. Depending on the use case, the reputation score of a verifier can be either determined computationally by the smart contract or given as a rating by the entity requesting the service. An automatic score is possible if the proof of a truthful and quality result can be determined computationally, such as requiring the original plaintext message of a cryptographic hash stored in the blockchain. If automatic scoring is not possible, the reputation system can rely on the entities that the verifier interacts with to assign a rating for the interaction.

2) Public Ethereum Reputation System: To keep track of the scores of all verifiers across various blockchain networks, a public Ethereum-based smart contract stores the details of registration of each verifier, in addition to its average reputation score and count of ratings it has received.

D. EMR Storage

Blockchain frameworks are incapable of storing large documents. Therefore, blockchain-based solutions typically resolve this weakness by storing such data off-chain and including a link to this data on-chain. Fully decentralized blockchain solutions optimally utilize decentralized storage systems, such as InterPlanetary File System (IPFS). IPFS stores files in a peer-to-peer network of public nodes, and provides content-based addressing based on the file's SHA-256 message digest, making it easy to establish connections between the blockchain and the storage [24]. Other solutions can adopt private storage systems, such as a private cloud-based database to store EMR data.

To achieve strong cross-chain interoperability levels, the blockchain networks should use a widely adopted and familiar

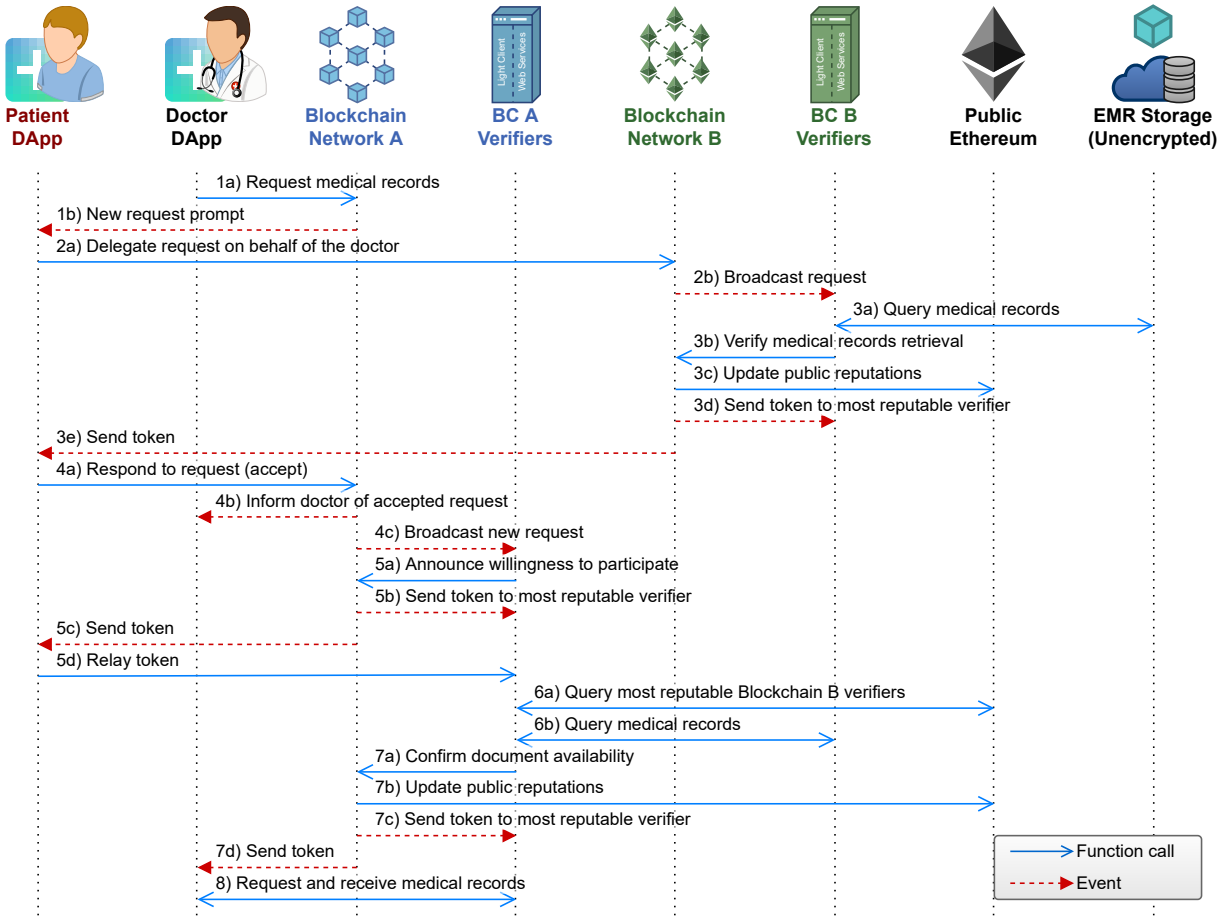

Figure 2: Sequence diagram of cross-chain interoperability for unencrypted EMR data sharing

standard for file formats. For example, a popular standard for healthcare and EMR-specific file format is Health Level Seven's Fast Healthcare Interoperability Resources (HL7 - FHIR), which provides a detailed description of how electronic healthcare data should be formatted like [25]. Such standardization makes EMR documents easily readable by any entity that gets access to the files.

E. Cross-Chain Hub DApp (CCHDA)

The primary interoperability hub within our solution is the DApp, which allows the exchange of transactions across blockchain networks through each network's default Application Programming Interfaces (API), in addition to a Fusion Interface (FI) layer that integrates all communication across the different blockchain networks. The blockchain APIs enable communication between the DApp and the blockchain networks, individually. The FI component processes cross-chain transactions, acting as a translation layer between the two blockchain networks.

F. Blockchain Entity Definitions

In the context of hospital-regulated EMR management blockchains, a simplified system design must involve two blockchain networks with their verifier nodes, a patient and a doctor, and the public Ethereum reputation system. The entities are defined as follows:

- **Hospital**: A regulatory entity that controls all patients, doctors, and verifiers registered to it. A hospital is responsible for deploying its smart contracts, which receive EMR data from patients and allows doctors to request the data. In our example, two hospital entities exist, such that **Hospital A** registers a patient and a doctor, and **Hospital B** only registers the patient with their data. All hospitals must register a set of verifier nodes that perform the communication verification tasks and other optional ones as explained earlier in the section. Moreover, the hospitals are envisaged to be responsible for assessing the performance of the verifiers and sharing the public reputation scores with the public Ethereum reputation system.
- **Patient**: A client entity that is registered in hospital blockchain networks. A patient is responsible for allowing or denying the doctor from accessing their data and making self-requests of the data. The patient may have additional responsibilities depending on the design of the healthcare network. For example, a patient-centered solution may require the patient to upload their data along with a token key on IPFS, and submitting the IPFS hash to the blockchain.

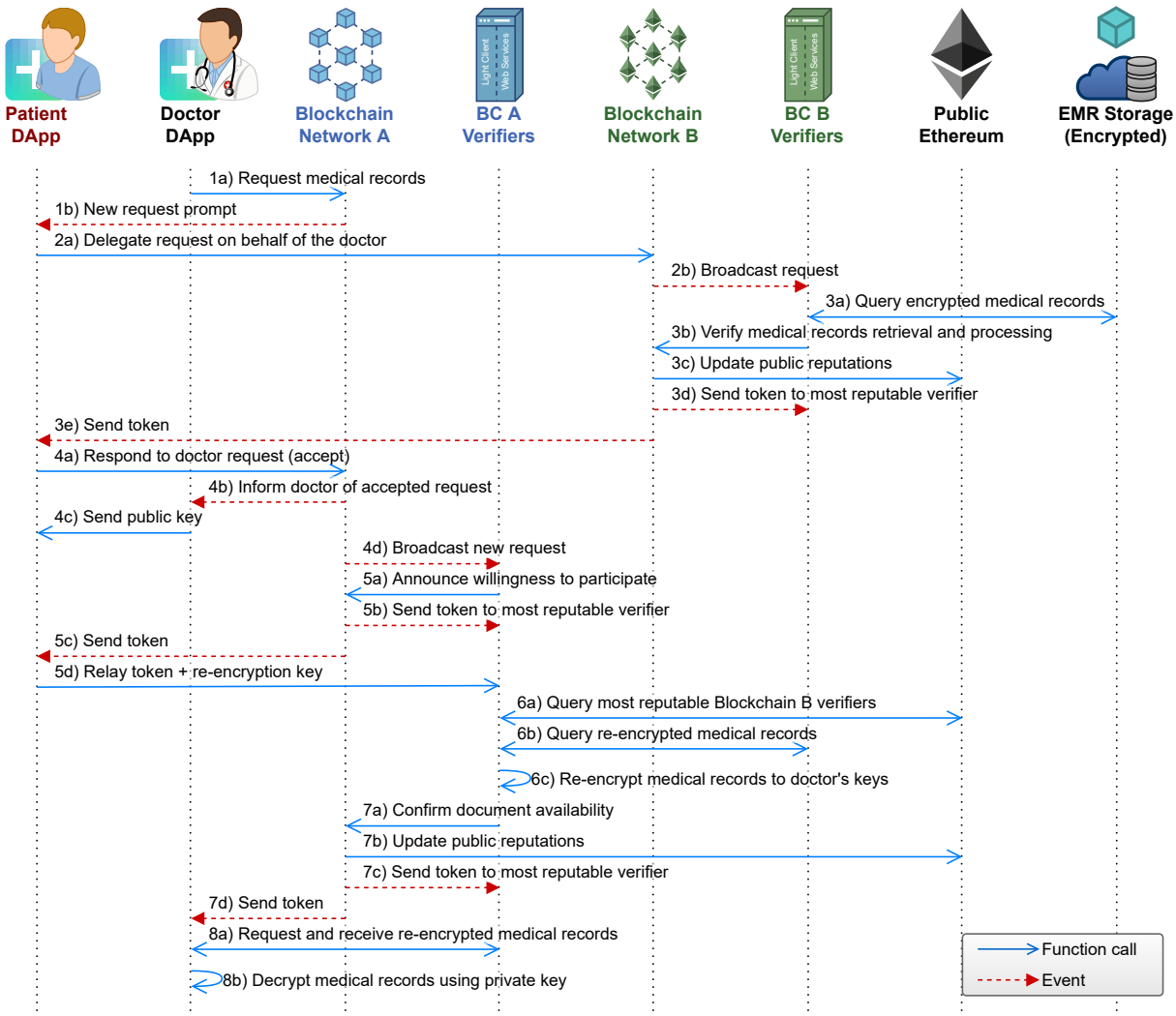

Figure 3: Sequence diagram of cross-chain interoperability for encrypted EMR data sharing

- **Doctor**: A client entity that is registered in hospital blockchain networks. A doctor is responsible for requesting EMR data from patients registered in the same network. Similar to the patient, the doctor may also have additional responsibilities, such as sharing their public key with a verifier for PRE.

All entities must be registered in their blockchain network and have identification addresses, such as Ethereum Address, and public-private key pairs.

G. Sequence of Interactions

We studied the sequence of interactions for cross-chain interoperability in the context of healthcare from the perspective of both encrypted and plaintext (unencrypted) EMR data.

1) Unencrypted EMR Data: This use case matches a scenario where a doctor requests the patient EMR data through a patient-centered Blockchain Network A (BN_A), and the patient EMR data is managed by a private Blockchain Network B (BN_B) which stores the data unencrypted on private cloud storage. The sequence, also depicted in Fig. 2, assumes all entities are registered in both networks and the patient has an EMR document in BN_B.

- The doctor generates a token and makes a request to BN_A containing the hashed token key. The network returns a request identifier as an event for future reference, which the doctor sends to the patient in an off-chain manner, along with the token key.
- The patient makes a self-request of the EMR document on behalf of the doctor to BN_B. The patient specifies the range of acceptable verifiers. The network returns the patient's request identifier as an event for future reference. The network also broadcasts the patient's request identifier to its internal verifiers informing them of a new request ready for fetching.
- Verifiers of BN_B fetch the EMR document from private cloud storage and calculate the document's hash and submit it to BN_B. The network verifies the correctness of the hash, which is stored privately on-chain and evaluates the performance rating of the verifiers based on their correctness and speed. The rating is used to update the network's internal reputation system and the public

Ethereum reputation system. The network then sends tokens to the most reputable verifier and the patient.
- The patient responds to the doctor's original request by accepting it. Since the doctor's request does not directly specify the patient identifier, the patient must include the doctor's original request token key in the response, which prevents other patients from responding to the doctor's request. BN_A then informs the doctor of the response, and broadcasts the doctor's request identifier to its internal verifiers, informing them of a new request ready for fetching and verifying.
- Verifiers of BN_A announce their willingness to participate in fetching and verifying the EMR document from BN_B verifiers. After an adequate number of verifiers respond, BN_A sends tokens to the most reputable verifier and the patient. The patient at this stage communicates in an off-chain manner with the verifier, to pass on the token received from BN_B to BN_A verifier.
- Chosen verifier of BN_A queries the public Ethereum reputation system to ensure that the chosen verifier of BN_B is reputable. If so, the two verifiers establish a connection to send the EMR document from BN_B to BN_A.
- Upon completion of EMR document transfer, the verifier of BN_A confirms to its network the availability of the document alongside a proof in the form of the document's hash. The network updates the reputation of the verifier in its internal records and the public Ethereum reputation system. The network then sends tokens to the verifier and the doctor for the final transfer of the EMR document.
- The doctor and the verifier use their tokens to establish a connection to transfer the EMR document from the verifier to the doctor.

2) Encrypted EMR Data: This use case aligns with the scenario where a doctor requests patient EMR data through a patient-centered BN_A, and the patient EMR data is managed by a public patient-centered BN_B which stores encrypted data on public IPFS storage. The sequence assumes all entities are registered and the patient has submitted an EMR bundle to BN_B. The EMR bundle contains the EMR document and an acquisition key, the hash of which the patient submits to the network.

- The doctor generates a token and makes a request to BN_A containing the hashed token key. The network returns a request identifier as an event for future reference, which the doctor sends to the patient in an off-chain manner along with the token key.
- The patient makes a self-request of the EMR document on behalf of the doctor to BN_B. The patient specifies the range of acceptable verifiers. The network returns the patient's request identifier as an event for future reference. The network also broadcasts the patient's request identifier to its internal verifiers informing them of a new request ready for fetching.
- Verifiers of BN_B fetch the EMR document from public IPFS storage and extract the document's acquisition key and submits it back to BN_B. The network verifies the correctness of the key and evaluates the performance rating of the verifiers based on their correctness and speed. The rating is used to update the network's internal reputation system and the public Ethereum reputation system. The network then sends tokens to the most reputable verifier and the patient.
- The patient responds to the doctor's original request by accepting it. Since the doctor's request does not directly specify the patient identifier, the patient must include the doctor's original request token key in the response, which prevents other patients from responding to the doctor's request. BN_A then informs the doctor of the response, and broadcasts the doctor's request identifier to its internal verifiers, informing them of a new request ready for fetching and verifying. The doctor, upon receiving the response, sends their public key to the patient in an off-chain manner.
- Verifiers of BN_A announce their willingness to participate in fetching and verifying the EMR document from BN_B verifiers. After an adequate number of verifiers respond, BN_A sends tokens to the most reputable verifier and the patient. The patient at this stage communicates in an off-chain manner with the verifier, to pass on the token received from BN_B to BN_A verifier. The patient uses their private key and the doctor's public key to create a re-encryption key, which is also sent to the BN_A verifier.
- Chosen verifier of BN_A queries the public Ethereum reputation system to ensure that the chosen verifier of BN_B is reputable. If so, the two verifiers establish a connection to send the EMR document from BN_B to BN_A. Upon completion of EMR document transfer, the verifier uses the re-encryption key to convert the state of EMR document encryption from the patient to the doctor.
- The verifier of BN_A confirms to its network the availability of the document alongside a proof in the form of the document's acquisition key. The network updates the reputation of the verifier in its internal records and the public Ethereum reputation system. The network then sends tokens to the verifier and the doctor for the final transfer of the EMR document.
- The doctor and the verifier use their tokens to establish a connection to transfer the EMR document from the verifier to the doctor. Upon receiving the document, the doctor decrypts the file using their private key.

IV. Implementation

Our implementation of a cross-chain interoperability system is based on the EMR document sharing across two hospitals. The implementation covers two smart contracts, one for each hospital. Each smart contract defines the characteristics of the main entities, which are the patient, doctor, and verifier. Additionally, the smart contracts control how the patient EMR documents, doctor requests, and verifier responses are processed.

We designed algorithms for submitting an EMR document to the internal blockchain, requesting the document from an external blockchain, and delegating the request by recreating

Algorithm 1: Submitting an EMR document, performed by the patient at Blockchain Network B

```
1 Function submitEMR(EMR document IPFS address da, Hash of EMR document token dth):
2     p ← Transaction initiator
3     Require that pEA is a patient
4     d ← New EMR document(patient ← p, docTokenHash ← dth)
5     Add d to list of documents at index da
```

Algorithm 2: Requesting an EMR from an external blockchain, performed by the doctor at Blockchain Network A

```
1 Function requestExternalEMR(Hash of request token rth, Min verifiers minV, Max verifiers maxV):
2     d ← Transaction initiator
3     Require that d is a doctor
4     minV ← minV < MIN_VER ? MIN_VER : minV
5     maxV ← maxV > MAX_VER ? MAX_VER : maxV
6     r ← New request(requester ← d, reqTokenHash ← rth, minVer ← minV, maxVer
        ← maxV, requestTime ← Current time, granted ← false, verified ← false)
7     Add r to list of requests at index rc
8     rc ← rc + 1
```

Algorithm 3: Self-requesting an EMR document from the internal blockchain, performed by the patient at Blockchain Network B

```
1 Function selfRequestEMR(EMR document IPFS address da, Min verifiers minV, Max verifiers maxV):
2     p ← Transaction initiator
3     d ← EMR document at index da
4     Require that p = d.patient
5     minV ← minV < MIN_VER ? MIN_VER : minV
6     maxV ← maxV > MAX_VER ? MAX_VER : maxV
7     r ← New request(requester ← p, document ← da, minVer ← minV, maxVer ← maxV,
        requestTime ← Current time, granted ← true, verified ← false)
8     Add r to list of requests at index rc
9     rc ← rc + 1
10    Emit an event to notify verifiers of the request
```

the request. Algorithm 1 describes the `submitEMR` function, which allows a patient to enter the address of the IPFS bundle as an identifier of the EMR document, accompanied with the hash of the token placed inside the bundle.

At a later time, a doctor can call the `requestExternalEMR` function defined in Algorithm 2 to request the EMR document from the patient, where the document is located in a different blockchain network. The function receives from the doctor a hashed token and the range of verifiers. The hashed token ensures that the request cannot be responded to unless the patient has a valid token, whereas the range of verifiers helps the doctor in controlling the confidence in the verification process and its cost. A range of a high number of verifiers would provide more confident responses, but for more expensive fees. As for the `selfRequestEMR` function explained in Algorithm 3, it performs almost the same actions as the `requestExternalEMR` function, with the difference being storing the IPFS address of the EMR document in the request and directly considering the request as granted and waiting to be verified.

When the EMR management functions are executed, the verifiers will have to process the requests at three steps as described in the set of algorithms executed by the verifier nodes only. The `verifyInternalRequest` detailed in Algorithm 4 is executed by various verifier nodes of Blockchain Network B, responding to the self-request of the patient made through `selfRequestEMR` function. The verifiers embed an identifier for the request and the original token key located inside the EMR document as part of their transaction. The function ensures that the response is acceptable by checking the validity of the token, and evaluates the performance of the verifier based on the latency and the correctness. In the algorithm, the ratings are mapped from 0 to 1, however, considering the high cost of floating-point operations, the scores were mapped from 0 to $2^16 - 1 = 65,535$. Once a sufficient number of verification responses are received, which is evaluated based on the requester's minimum and maximum number of verifiers, the function updates the reputations of the verifiers and sends a token to the one with the highest final reputation score. The verifier's reputation is updated such that the new score is added to the old reputation average, and the reputation count is incremented by one. Updating the reputation average is computed dynamically, to minimize the stored data and processing time.

Algorithm 5 shows the `requestParticipation` function, which receives participation requests from verifiers to fetch and verify the EMR document from Blockchain Network B's verifiers. The function simply looks for the verifier of the highest reputation and sends to it and the patient tokens for them to establish a secure connection. In this function, the comparison of the reputations is not computed based on the average reputation alone, but it also combines the reputation count as given by Laplace's rule of succession, which assumes two more random ratings were given to the verifier. The two random ratings, assuming one is positive and the other is negative, result in an extra score of 1, whereas the reputation count is increased by 2. Finally, the `documentAvailable` function in Algorithm 6 is called by the verifier once it is ready to send to the doctor the re-encrypted EMR document.

V. Discussions

In this section, we review our proposed architecture and designed algorithms, by verifying the functional correctness, analyzing the execution costs, and assessing the overall security.

A. Correctness Verification

To evaluate our proposed system, we implemented each of the hospital blockchain network as a separate Ethereum

Algorithm 4: Verifying a request, performed by verifiers at Blockchain Network B

Function `verifyInternalRequest(`*Request identifier* `ri`*, EMR document token* `dt`**):**
- `r` ← Request at index `ri`
- Require that `r` is granted by patient, internal, and not already verified
- `l` ← Current time − `r.requestTime`, where `l` is the response latency
- **if** `r` can be verified **then**
  - `vt` ← `keccak256(dt)` == `r.docTokenHash`, where `vt` indicates the validity of the token
  - `vr` ← `vt` × Map `l` from 0 to 1, where `vr` indicates the verifier's rating
  - Add transaction initiator and `vr` to `r`'s lists of verifiers `r.lv` and ratings `r.lr`
- **if** Sufficient verifications received **then**
  - `bvs` ← 0, where `bvs` indicates the best verifier score
  - **for** $i \leftarrow 0$ to length of `r.lv` **do**
    - `vs` ← $\texttt{r.lv}_\texttt{i} \times (\texttt{sa}_\texttt{i}+1)^2$, where `vs` indicates the verifier's current score and `sa` indicates the verifier's average score
    - **if** `vs` > `bvs` **then**
      - `bvs` ← `vs`
      - `bva` ← Address of the verifier
    - `sa` ← $(\texttt{sc} \times \texttt{sa} + \texttt{vr})/(\texttt{sc}+1)$, where `sc` indicates the verifier's score count
    - Send token to the verifier of `bva` and `r.requester`

Algorithm 5: Requesting to participate in the external verification process, performed by verifiers at Blockchain Network A

Function `requestParticipation(`*Request identifier* `ri`**):**
- `r` ← Request at index `ri`
- Require that `r` is granted by patient, internal, and not already verified
- `l` ← Current time − `r.requestTime`, where `l` is the response latency
- **if** `r` can be verified **then**
  - **if** Length of `r.lv` = 0 **then**
    - `r.lv` ← Transaction initiator
  - **else**
    - `bvs` ← $(\texttt{sc} \times \texttt{sa}+1)/(\texttt{sc}+2)$, where `sa` and `sc` are for the best verifier
    - `nvs` ← $(\texttt{sc} \times \texttt{sa}+1)/(\texttt{sc}+2)$, where `sa` and `sc` are for the new verifier
    - **if** `nvs` > `bvs` **then**
      - $\texttt{r.lv}_0$ ← Transaction initiator
  - `r.participations` ← `r.participations` + 1
- **if** Sufficient participations were received **then**
  - Send token to the verifier of `bvs` and the patient

Algorithm 6: Confirming document availability, performed by verifiers at Blockchain Network A

Function `documentAvailable(`*Request identifier* `ri`**):**
- `r` ← Request at index `ri`
- Require that `r` is granted by patient, internal, and not already verified
- Require that `r`'s external verifier = transaction initiator
- Send token to the verifier and the doctor

network. The smart contracts for the two hospitals have been developed using Solidity language and Truffle framework for managing the compilation and deployment. The smart contracts, which we denote as $\texttt{HSC}_\texttt{A}$ for hospital A and $\texttt{HSC}_\texttt{B}$ for hospital B were compiled using the standard Ethereum Solidity compiler version 0.7.0 with code optimization at 200 runs and deployed to local Ethereum test networks (testnets) created by Ganache from the Truffle Suite. The testnets for $\texttt{HSC}_\texttt{A}$ and $\texttt{HSC}_\texttt{B}$ contain 8 and 7 unique Ethereum addresses respectively. Each testnet has its hospital administration and verifiers, whereas both share the patient, and only $\texttt{HSC}_\texttt{A}$ has a doctor. The testing of the code was automated using Truffle's JavaScript-based tests, allowing various use cases to be simulated with capabilities, such as time advancement and token generation. The tests were scripted such that they match the sequence diagram Fig. 3 depicts.

- **Deploying networks and uploading EMR documents**: Each of the administrations of the two hospitals deploy their blockchain networks and register their entities. $\texttt{HSC}_\texttt{A}$ registers a patient, a doctor, and 5 verifiers, while $\texttt{HSC}_\texttt{B}$ registers a patient and 5 verifiers. Then the patient prepares a random token to be placed along with the EMR document and uploads it to IPFS, the address of the uploaded bundle is its hash. The bundle hash and the hash of the token are then submitted to $\texttt{HSC}_\texttt{B}$ by calling `submitEMR`.
- **Requesting EMR document**: The doctor generates a random request token and specifies $[1, 4]$ as the range of number of verifiers, which are used in calling `requestExternalEMR` from $\texttt{HSC}_\texttt{A}$. The network processes the request to validate the doctor parameters, resulting in constructing the desired range of verifiers $[2, 4]$ because the minimum number of acceptable verifiers is 2. Upon successful processing of the request, the network returns the request identifier to the doctor as a broadcasted event. Fig. 4 depicts the transaction inputs and outputs from our testing.
- **Delegating request**: The doctor informs the patient about the desired data so that the patient can identify which EMR document to look for. Additionally, the doctor sends the request token to the patient for future use. The patient then calls self-request function `selfRequestEMR` at $\texttt{HSC}_\texttt{B}$, specifying the IPFS address and the desired range of verifiers, which in our tests is $[2, 4]$. The network processes the request and immediately broadcasts the IPFS address to the $\texttt{HSC}_\texttt{B}$ verifiers.
- **Verifying request**: The verifiers communicate with the IPFS nodes and obtain the file bundle. The bundle contains the encrypted EMR document and the

original possession token identifier, which is passed by the verifiers as a parameter while executing `verifyInternalRequest` at HSC_B. In our tests, the first and third verifiers respond with invalid tokens, with the remaining two responding correctly. The function must additionally be executed by the fifth verifier, which acts as a time-based trigger. At this call, the function executes the `sufficient verifications received` path, resulting in a transaction as shown in Fig. 5.

- **Responding to request**: The patient accepts the doctor's request by calling `respondRequest` at HSC_A. In the function call, the patient must pass the doctor's request token as proof that it is the intended entity for the documents to be requested from. By accepting the request, the network informs its verifiers to start participating in the external verification process. Additionally, the network also informs the doctor of the acceptance, allowing the public key to be sent to the patient. The patient uses the public key of the doctor, along with their private key to generate a re-encryption key from the patient to the doctor.
- **Announcing willingness to participate**: The verifiers of HSC_A execute `requestParticipation` to get evaluated based on their reputations. Once a sufficient number of verifiers respond to the request, the network determines the verifier holding the highest reputation score. The network broadcasts a communication token to the verifier of the best reputation score and the patient. Later, the chosen HSC_A verifier connects with the patient and receives the communication token with HSC_B verifier, in addition to the re-encryption key.
- **Verifying external interactions**: After verifying the reputation of HSC_B's verifier based on its score in the public Ethereum reputation system, the two chosen verifiers establish a connection to transfer the EMR document. Then, the verifier of HSC_A re-encrypts the EMR document to become readable by the doctor.
- **Confirming document availability**: The chosen verifier of HSC_A informs the network that the verification process is complete. The network then sends a token to the verifier and the doctor.
- **Receiving and decrypting EMR document**: The doctor establishes a connection with the verifier node to transfer the EMR document. Upon receiving the document, the doctor decrypts the file, revealing the requested data.

Figure 4: The details of a doctor-initiated transaction to request an external EMR document

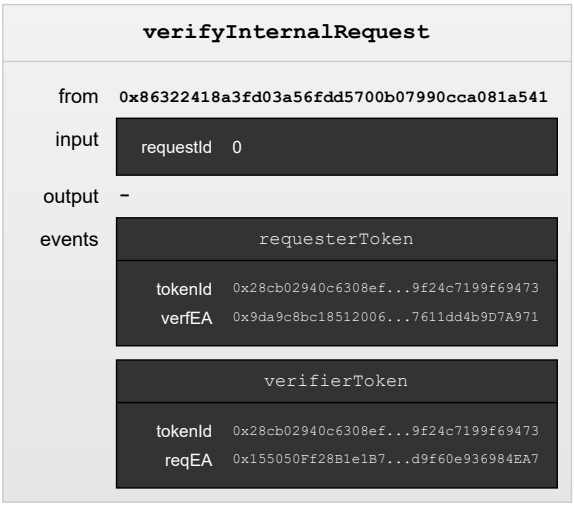

Figure 5: The details of a verified internal EMR document self-request

B. Cost Analysis

Executing our cross-chain interoperability solution on local testnets enables the ability to analyze the cost of executing the smart contract functions on both networks individually. Our cost analysis is based on the scenario described in the correctness verification, at which there are 2 hospital networks, the first of which has a patient and a doctor, and the other one has only the patient with their EMR data. Throughout the evaluation, all smart contract transactions were recorded including their transaction costs. The transaction cost is measured in gas units, and it depends on the computation complexity of the function, alongside its transmission and deployment costs.

Table I and Table II present the cost of the prominent functions of HSC_A and HSC_B, respectively. The estimated costs in USD were based on the average gas price of 80wei and the average Ether price of $1200, both recorded as of January 17, 2021. The functionality achieved by HSC_A includes registering entities, requesting documents, responding to requests, participating in the external verification process, and claiming that the document is available for the doctor. The function achieved by HSC_B also includes registering entities as well as submitting documents, initiating self-requests, and verifying the internal request.

A typical trend in both networks is the positive correlation between the number of modified state variables and the cost. Besides the smart contract deployment, which is a one-time action, the functions that cost the most are the ones performed by the verifier nodes, especially when the final evaluation of all nodes is executed. The remaining functions, such as registering

Table I: Function caller, and gas and currency costs of $\texttt{HSC}_\texttt{A}$ functions.

Function Caller	Function Name	Transaction Cost [Gas]	Cost [USD]
Admin A	$\texttt{HSC}_\texttt{A}$	1,801,931	172.99
Admin A	`registerPatient`	43,442	4.17
Admin A	`registerDoctor`	43,473	4.17
Admin A	`registerVerifier`	43,465	4.17
Doctor	`requestExternalEMR`	166,183	15.95
Patient	`respondRequest`	49,475	4.75
Verifier	`requestParticipation`	68,231	6.55
Verifier	`requestParticipation + evaluation`	81,152	7.79
Verifier	`documentAvailable`	31,177	2.99

Table II: Function caller, and gas and currency costs of $\texttt{HSC}_\texttt{B}$ functions.

Function Caller	Function Name	Transaction Cost [Gas]	Cost [USD]
Admin B	$\texttt{HSC}_\texttt{B}$	1,804,518	173.23
Admin B	`registerPatient`	43,442	4.17
Admin B	`registerVerifier`	43,460	4.17
Patient	`submitEMR`	64,423	6.18
Patient	`selfRequestEMR`	204,920	19.67
Verifier	`verifyInternalRequest`	99,629	9.56
Verifier	`verifyInternalRequest + evaluation`	194,788	18.70

entities and responding to requests, have costs ranging from $3 to $7, which is reasonable considering the tenfold increase in Ethereum transaction costs over the last year.

C. Security Assessment

Considering that our proposed solution is generic and adaptable to diverse blockchain systems, the security goals achieved are dependant on the security measures adopted by the underlying blockchain systems. For instance, a misconfigured and insecure blockchain system that adopts our solution would remain insecure. Therefore, as part of the analysis of our proposed solution, we investigated the security elements to determine if the support for cross-chain interoperability in the CCHDA and blockchain network would expose the systems to any security threat. In doing so, we assume that the individual blockchain is secure.

- **Privacy and confidentiality**: The proposed system does not require participating blockchain entities to share private or sensitive information. As the data translations conducted as part of the interoperability process are managed by the data owner entity, these do not present additional data privacy risks. Furthermore, confidential communication between the requesting and the translating entities relies on the off-chain communication link used by the two entities, such as private peer-to-peer messaging or a face-to-face exchange in real life. In the healthcare example as discussed earlier, cryptographic constructs are used to achieve appropriate protection for the patient's EMR.
- **Integrity and authenticity**: The proposed solution does not introduce additional third parties to translate or relay information and transactions across the interoperating blockchain networks. For example, in the case of sharing an EMR document, the patient is responsible for translating the transaction and communicating with the verifier nodes to relay information, such as the re-encryption key and the token for connecting with the external verifier. Furthermore, communication between the entities involved is regulated by the blockchain network registering such entities that utilize the ability to generate tokens ensuring communication link security and authenticity of both parties.
- **Availability and resiliency**: As blockchain is an inherently peer-to-peer system, it facilitates protection against availability-focused attacks, such as Denial of Service (DoS). However, blockchain interoperability mandates trustworthy connections between the participating blockchain networks. Another major part of our design is the CCHDA whose availability is affected by remote servers, such as Infura that connect the DApp to the blockchain networks. This in return affects the availability of cross-chain communication in case the DApp was required for a translation of the interactions. In our architecture, we ensure that the entities that manage the CCHDA have an interest in a successful cross-chain interaction that facilitates achieving protection against adverse attempts.
- **Non-repudiation and provenance**: Given the considerable off-chain interactions as part of the proposed architecture, data provenance is significant to facilitate non-repudiation especially due to the decentralized nature of the setup. Within the healthcare use case, all entities, including the patient, doctor, and verifiers, are capable of sharing invalid information. For example, a patient can submit an invalid EMR document token hash to the healthcare causing any verifier response to be incorrect, thereby lowering their scores. To overcome the possibility of such attacks, our architecture uses internal and exter-

nal reputation systems that allow entities to inspect the reputation of other entities within the same network, in addition to other public entities, such as the verifiers of an external network.

VI. Conclusion

In this paper, we have proposed an application-based cross-chain interoperability solution that is fully decentralized, with the potential to be adapted to a wide range of use cases at low implementation and deployment cost. Our proposed architecture utilizes the fusion interface layer inside the Cross-Chain Hub DApp of critical entities in the network to translate cross-chain interactions and delegate requests from one blockchain network to another. Our system adopts the concept of verifier nodes that are internal computation nodes for each network with the capability of communicating with other verifier nodes of external networks to ensure the validity of responses and shared data. To govern the off-chain interactions, our approach leads to a public reputation system for external verifiers, allowing nodes to query and learn the reputation of other nodes. To confirm the viability of our proposed architecture, we designed a set of algorithms specific to the healthcare industry and implemented the smart contracts for sharing patient-centered EMR documents across separate blockchain networks. We evaluated our solution to verify its correctness in terms of operations and analyze the cost overheads. Furthermore, we analyzed the security implications of our solution that revealed the proposed cross-chain interoperability system does not introduce additional security threats.

VII. Acknowledgement

This publication is based upon work supported by the Khalifa University of Science and Technology under Award No. CIRA-2019-001.

References

[1] M. Rauchs, A. Blandin, K. Bear, and S. B. McKeon, "2nd global enterprise blockchain benchmarking study," *Available at SSRN 3461765*, 2019.

[2] C. Lima, "Developing open and interoperable dlt\/blockchain standards [standards]," *Computer*, vol. 51, no. 11, pp. 106–111, 2018.

[3] M. Zarour, M. T. J. Ansari, M. Alenezi, A. K. Sarkar, M. Faizan, A. Agrawal, R. Kumar, and R. A. Khan, "Evaluating the impact of blockchain models for secure and trustworthy electronic healthcare records," *IEEE Access*, vol. 8, pp. 157 959–157 973, 2020.

[4] V. Buterin, "Chain interoperability," *R3 Research Paper*, 2016.

[5] S. Schulte, M. Sigwart, P. Frauenthaler, and M. Borkowski, "Towards blockchain interoperability," in *International Conference on Business Process Management*. Springer, 2019, pp. 3–10.

[6] P. Lafourcade and M. Lombard-Platet, "About blockchain interoperability," *Information Processing Letters*, p. 105976, 2020.

[7] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[8] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability." in *NDSS*, 2019.

[9] D. Robinson, "Htlcs considered harmful," in *Stanford Blockchain Conference*, 2019.

[10] S. Thomas and E. Schwartz, "A protocol for interledger payments," *URL https://interledger. org/interledger. pdf*, 2015.

[11] I. A. Qasse, M. Abu Talib, and Q. Nasir, "Inter blockchain communication: A survey," in *Proceedings of the ArabWIC 6th Annual International Conference Research Track*, 2019, pp. 1–6.

[12] G. Sagirlar, J. D. Sheehan, and E. Ragnoli, "On the design of co-operating blockchains for IoT," in *3rd International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2020, pp. 548–552.

[13] J. Kwon and E. Buchman, "Cosmos whitepaper," 2019. [Online]. Available: https://cosmos.network/cosmos-whitepaper.pdf

[14] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, 2016. [Online]. Available: https://github.com/polkadot-io/polkadotpaper/raw/master/PolkaDotPaper.pdf

[15] P. Zhang, J. White, D. C. Schmidt, and G. Lenz, "Applying software patterns to address interoperability in blockchain-based healthcare apps," *arXiv preprint arXiv:1706.03700*, 2017.

[16] P. Zhang, M. A. Walker, J. White, D. C. Schmidt, and G. Lenz, "Metrics for assessing blockchain-based healthcare decentralized apps," in *19th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, 2017, pp. 1–4.

[17] G. Carter, B. Chevellereau, H. Shahriar, and S. Sneha, "Openpharma blockchain on fhir: An interoperable solution for read-only health records exchange through blockchain and biometrics," *Blockchain in Healthcare Today*, 2020.

[18] B. Chevallereau, G. Carter, and S. Sneha, "Voice biometrics and blockchain: Secure interoperable data exchange for healthcare," *Blockchain in Healthcare Today*, 2019.

[19] C. Chinchilla, "A Next-Generation Smart Contract and Decentralized Application Platform," 2019, [Accessed on: 18 January 2021]. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper/

[20] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[21] "Quorum Whitepaper," 2018, [Accessed on: 18 January 2021]. [Online]. Available: https://github.com/jpmorganchase/quorum/blob/master/docs/Quorum\%20Whitepaper\%20v0.2.pdf

[22] 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*, 2018, pp. 1–15.

[23] M. M. Madine, A. A. Battah, I. Yaqoob, K. Salah, R. Jayaraman, Y. Al-Hammadi, S. Pesic, and S. Ellahham, "Blockchain for giving patients control over their medical records," *IEEE Access*, vol. 8, pp. 193 102–193 115, 2020.

[24] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," 2014, [Accessed on: 18 January 2021]. [Online]. Available: https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf

[25] "Hl7 - fast healthcare interoperability resources documentation," 2019, [Accessed on: 18 January 2021]. [Online]. Available: https://hl7.org/implement/standards/fhir/