Decentralized Health Data Distribution: a DLT-based Architecture for Data Protection

Decentralized Health Data Distribution: A DLT-based Architecture for Data Protection

Gioele Bigini[*], Mirko Zichichi[†], Emanuele Lattanzi[*], Stefano Ferretti[*], Gabriele D'Angelo[‡]
[*]Department of Pure and Applied Sciences, University of Urbino, Italy
[†]Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
[‡]Department of Computer Science and Engineering, University of Bologna, Italy
g.bigini@campus.uniurb.it, mirko.zichichi@upm.es, {emanuele.lattanzi,stefano.ferretti}@uniurb.it, g.dangelo@unibo.it

***Abstract*—The management, protection and sharing of sensitive data such as those associated with the health domain are crucial in enabling personal care and contributing to worldwide medical advancements. Distributed Ledger Technologies (DLTs) allow for data protection compliant solutions in untrusted contexts that guarantee data immutability, protection and transparency when needed. This paper proposes an architecture based on DLTs, Smart Contracts and Distributed File Storage (DFS), enabling user data sovereignty, confidentiality and secure access control. A use case on health data is presented, where we apply a combination of DLT, DFS and an access control mechanism to allow users to distribute their data. Finally, we show an experimental evaluation of the overall architecture to demonstrate the feasibility of implementing practical DLT-based healthcare solutions. The results are collected through independent tests, available opensource, that verify the system's response time in each of its functions and as the load increases. The results are promising and show that the system is feasible and can scale as the load increases.**

***Index Terms*—Distributed Ledger Technology, Smart Contracts, Health Data, Distributed Storage**

I. Introduction

Digital technologies are continuously transforming society. Personal devices are foundational to this transformation, where individuals are the primary sources of information generation. Storing data in inaccessible and disconnected data lakes makes them inaccessible to the public for innovation [1]. Following this, the interest in data ownership arises first and foremost from the lack of transparency in how data is collected, stored and used by different services and companies. In this regard, a low effort has been spent on easing the data management for an individual to understand and manage the risks associated with exploiting his private data.

Healthcare could enormously benefit from the ability to share information, transitioning from centralized to decentralized system architectures. Individuals can make a substantial contribution through personal devices to science, specifically in personalized medicine [2], [3]. Unfortunately, there are strong barriers represented by privacy and security for the health sector: sharing information without the individual's explicit consent constitutes a substantial violation of an individual's rights. Regulations such as the European Union's General Data Protection Regulation (GDPR) [4] help promote a pro-individual view. Specifically, these regulations impose many accountability measures on actors responsible for processing personal data and assign several rights to individuals. However, these do not always address the lack of transparency in managing personal information and the technical ability to make personal data portable, i.e. data interoperability [4]. We argue that a vision toward including the individual in the personal data flow could be reached by developing a user-centred framework for managing personal data. One such is that the individual owns control over data assets and companies comply with the legislation. It can be achieved by decoupling file storage, access control mechanisms, and application logic to guarantee data protection and security. This would pave the way for individuals' privacy needs and for a significant impact on the capabilities the healthcare field could aim for, as well as a unique common data lake and market [5], that capitalizes on the data interoperability for the social good [6]–[8].

The objective of this work is to provide a decentralized access mechanism that conveys a practical way for individuals to store, protect, and share their personal data, i.e. health data generated through mobile personal devices. A cryptographic method is in charge of encrypting each piece of information stored in a Decentralized File Storage (DFS). A Distributed Ledger Technology (DLT) stores universal, immutable resource identifiers to the data and provides smart contracts to ensure the data integrity verifiability and manage Access Control Lists (ACLs) associated with each piece of data. The paper's contribution is the following: we propose an architecture based on the use of DLT, smart contracts and DFS. The latter allows for the decentralized distribution of health data yet guarantees data sovereignty to users, confidentiality and secure access control. The access control is implemented using distributed authorization, thus envisioning a decentralized context. Moreover, we present a health-data-related use case and experimental evaluation of the overall architecture. The use case demonstrates, through a data-sharing scenario, how the proposed system can be exploited. On the other hand, the system evaluation shows that the system we proposed is viable for such a use case through the results in terms of performance and latency.

The remainder paper is organised as follows. Section II

This work has received funding from Regione Marche with DDPF n. 1189 and from the EU's Horizon 2020 research and innovation programme under the MSCA ITN grant agreement No 814177 LAST-JD-RIoE.

presents the background, while Section III describes related work. Section IV specifies the system architecture. In Section IV we describe a health data sharing use case. Performance is evaluated in terms of measured latency in Section VI before conclusions, in Section VII.

II. Background

In this section, we describe the technologies that will be used for building up the proposed software architecture.

A. Distributed Ledger Technology (DLT)

The resistance to manipulation makes DLTs a very promising technology for developing new types of applications where immutability and transparency represent a requirement. Examples of these applications can be found in general-purpose blockchains that implements smart contracts, e.g. through Ethereum [8], [9] or Hyperledger Fabric [10].

A smart contract is a code deployed in a DLT environment or the source code from which such code was compiled [11]. This code is executed deterministically by different participants in the DLT, who receive the same inputs and then perform a computation that leads to the same outputs. When a smart contract is deployed on the DLT and the issuer is confident that the code embodies the intended and proper behaviour (e.g., by reviewing the code), then transactions originating from that contract do not require the presence of a third party to have value. This principle is based on the assumption that most DLT nodes are honest (i.e. the opposite of an attacking node) and follow the same protocol.

B. Decentralized File Storage (DFS)

A Decentralized File Storage (DFS) offers an alternative way to store files to the traditional client-server models, i.e. where a domain name is provided and is then translated to an IP address [12], [13]. A DFS comprises a network of peer nodes that have their storage and follow the same protocol for content storing and retrieval. In Content-Based Addressing, contents are directly queried through the network rather than establishing a connection with a server. In order to know which DFS node in the network owns the requested contents, it is possible to rely on a distributed hash table in charge of mapping the contents, i.e. files and directories, to the addresses of the peers owning such data. DFS follows this approach and offers higher data availability and resilience using data replication.

C. Decentralized Access Control Mechanisms

The objective of Access Control Systems (ACS) is to regulate access to system resources by enforcing permissions based on a set of system policies to determine who can access information. Centralized ACS rely on a single authority to access the data and, therefore, carry the risk of a single point of failure and the loss of privacy [14]. DLTs solve the single point of failure by providing the means to implement decentralized ACS. Different approaches are : (1) Discretionary ACS, which enables the management of data stored outside of the DLT through the access control policy stored on the ledger [15]; (2) Mandatory ACS, which constrains the ability of a subject to access data through smart contracts [16] ; (3) Role-based ACS, allows to achieve authentication based on user roles [17]; (4) Attribute-based ACS, grants or denies user requests based on user's attributes, object and environment conditions [18].

D. Cryptographic Schemes

In this work, we refer to the cryptographic schemes described in the following.

1) Proxy Re-Encryption (PRE): The Proxy Re-Encryption offers a scalable protocol where it is not necessary to know the recipient of data in advance [19]. It is useful when communication between an arbitrary number of data owners and consumers is dynamic. PRE is a type of public-key encryption, where an untrusted proxy entity transforms a ciphertext c, encrypted with a public key pk_1, into a ciphertext decryptable with a private key sk_2, without learning anything about the underlying plaintext. This is possible using a re-encryption key rk_{1-2} generated by the data owner who has the key pair (pk_1, sk_1) and that divulges (to the proxy) the authorisation of access to the plaintext to a data consumer holding the keypair (pk_2, sk_2).

2) Threshold Scheme: A (t, n)-threshold scheme can be employed to share a secret among a set of n participants, allowing the secret to being reconstructed using any subset of t (with $t \leq n$) or more shares, but no subset of less than t. In a network where more than one server keeps secret shares, a mutual consensus can be reached when t nodes provide the shares to a secret recipient, enabling the secret to be known. This can be used to provide data protection to a user who is sharing a secret since none of the servers can obtain the whole secret without the help of other $t - 1$ servers.

III. Related Work

With the emergence of the first proposals to use DLT-based systems beyond finance, i.e. digital currencies, some researches have already found a relationship between DLT and personal data sharing [15]. The general approach, in this context, is to store access control policies on DLTs securely so that the applicant can be made aware of his or her permissions to access his or her personal data stored outside the DLT [8], [9], [13], [15].

Yan et al. [9] present a Personal Data Store (PDS) that enables the collection, storing and fine-grained access to their data using a (t, n)-threshold scheme. Their solution of sharing personal information in pieces was innovative but expensive and not GDPR compliant, i.e. the system stores personal data directly on the DLT [20]. Another possible approach is to program access control policies as smart contracts in order to manage control automatically in compliance with GDPR [10]. Koscina et al. [21] enable healthcare data exchange through a distributed architecture, with the focus on consent given through smart contracts. In their system users keep a digital copy of their medical data in a personal data account that can be hosted on any cloud-based data management service.

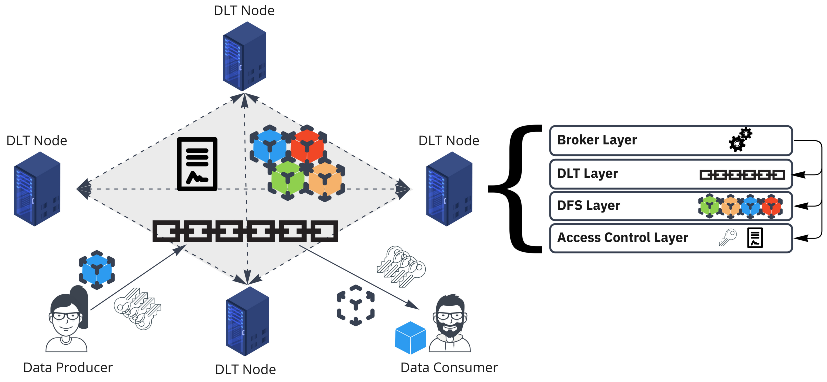

Fig. 1. Architecture of the Decentralized Health Data Sharing.

Other research efforts introduce the usage of mobile devices, i.e. the smartphones [2] and reputation systems to encrypt and share data [22]. In other cases, the focus is on the effective compliance of these systems with the healthcare sector [23], or the possibility of providing an identity to the participants anticipating a health digital identity system. Other solutions introduce an economic incentive for those who disseminate their health data [24] since these effectively contribute to a piece of greater knowledge for personalised medicine.

To the best of our knowledge at the time of writing, no related works provide a solution for the IoMT environment involving the usage of Hyperledger Fabric, smart contract and threshold schemes based on PRE for the encryption keys management while providing an experimental evaluation. We can only refer to this work [25], which provides a similar architecture but imply Ethereum without providing any performance evaluation.

IV. Decentralized Health Data Sharing Architecture

In this section, we describe our proposed system architecture. Emphasis is given to the specific case of health-related personal data, making up the main architectural drivers of our design. The system is composed of different technologies that we describe with the aid of Figure 1:

- **Broker Layer** - the APIs which a User Interface can exploit to interact with the other system components.
- **Distributed Ledger Technology** - the underlying network of nodes that maintain the ledger that validate (through the untamperability property) data exchanged by the peers involved.
- **Decentralized File Storage Layer** - the component that deals with the actual storage of encrypted personal data.
- **Access Control Layer** - a set of technologies and schemas that enable the access policies declaration (through smart contracts) and the actual data access (through keys distribution).

This architecture was built with a set of principles, functional and non-functional requirements in mind: (i) *Data Validation*: the integrity of data generated by (or on behalf) of users must be guaranteed and verified. To this end, the system takes full advantage of the untamperability property of DLTs. (ii) *Traceability*: not only the integrity of personal data, but also their life cycles must be guaranteed and verified. Also in this case, the system takes advantage of DLTs and their smart contract features. (iii) *Privacy-by-Design*: while we need to make it difficult to change or delete data from the ledger, at the same time, if we intend to comply with regulations (i.e. GDPR), the system still requires the modification or deletion of data under certain circumstances, e.g. GDPR's "right to be forgotten". This is one of the main breaking points between the DLTs and the GDPR [20] that led us to the use of off-chain DFS, i.e. data not stored directly on-chain. (iv) *Data Protection*: cryptography plays a key role in the authenticity and integrity of the data and its treatment among all the agents in the data processing chain. For this reason, we refer to advanced cryptographic techniques [26] to verify the authenticity of data and to implement users' preferences in maintaining their privacy, i.e. authorized access.

In the following, we are going to devise each system component with a dedicated sub-section.

A. Broker Layer

The Broker Layer is responsible for the interaction with the external environment. It manages the interaction with the access control layer and the decentralized file storage. From a logical point of view, the functioning of the layer is depicted in Figure 1. This layer handles the interactions with the system from the outside. Whenever an interaction attempt is made, the broker dialogues with the corresponding underlying DLT, DFS and access control layers. This ensures that the system is modular in the sense that new layers can be easily added, but also ensures that the only method of access to critical modules is through a broker.

B. Decentralized File Storage Layer

As said earlier, DLTs are designed to make it difficult to be immutable. Therefore, one approach to meet the Privacy-by-Design requirements is to implement off-chain storage of personal information [20].

The DFS network stores and shares data, files and directories in the form of objects that are identified by a content identifier (CID). This CID is the result of the application of a hash function to a piece of data and it is also used to retrieve it in the network. Once a piece of data is published in the off-chain storage, i.e. the DFS, the returned CID can be employed to retrieve it and will enable the verification of its integrity. Thus, for instance, when the piece of data is firstly uploaded into the system, it becomes a DFS object, and then then it is asynchronously referenced through its CID into a DLT. It would constitute the principle of hash pointing. If any other node in the network tries to share the same exact object, the CID will always be the same.

Thus, in our system, health data are stored *encrypted* in a DFS and then referenced in a DLT. Data protection is maintained due to the fact that all data is encrypted at the User Interface/Core level. This solution has the additional benefit of improving performances and providing higher availability for data reads and writes without introducing central trusted parties [13].

C. Distributed Ledger Technology Layer

At the core of the architecture, the DLT layer provides a network of peer nodes holding a ledger that ensures immutability and transparency with respect to the records to be stored in the smart contract. This makes it possible to store the entire history of transactions between the various peers and consequently of the requests made to the access mechanism. It is important to emphasise that this is a permissioned network, in which it is possible to establish consensus policies. The choice is based on the fact that this network is also accepted under the GDPR, as opposed to a public DLTs, which is transparent outside the participants and would allow external actors to view information [20]. Generally speaking, as soon as a request is received, the broker is able to dialogue with the peer that takes charge of the readings and writings on the ledger through the smart contract. Every time an operation is carried out on the smart contract, it is reflected to the network peers that keep the distributed ledger integrity.

D. Access Control Layer

Smart contracts are part of the proposed architecture where access control logic to share data is performed. Through these, access to the data can be purchased or can be allowed by the owner. The use of data, then, is authorized only to users indicated by the policies in a smart contract owned by the data subject. Hence, due to the presence of smart contracts, no direct interactions are needed between the data owner and users interested in his data. In practice, each piece of data stored in the DFS is referenced in a specific smart contract through the CID of the data or of the directory. One simple policy would be for the smart contract to maintain an Access Control List (ACL) that represents the rights to access one or more data. In the rest of the paper, we will focus on the application of such kind of policy. Once a user is eligible to access certain data, i.e. he is in the related ACL on a smart contract, then he/she will also be eligible to obtain the key used for encrypting the data.

1) Cryptographic Scheme: We provide a general overview of the cryptographic scheme without going into the details of the implementation in order to convey a clear understanding of the whole access control layer. We refer to a hybrid cryptographic scheme making use of both asymmetric and symmetric keys. The general principle is that each piece of health data is encrypted using a symmetric "content" key k and then this key is encrypted using an asymmetric keypair (pk_{KEM}, sk_{KEM}). This consists of a Key Encapsulation Mechanism (KEM) [27] in which the key is encapsulated and the capsule is distributed.

2) Key Distribution Component: The presence of an off-chain key distribution component is necessary: i) to free the owner of the data from the burden of managing the distribution of keys, which can be very costly in the case of fine-grained access; ii) to complement the public execution operations of smart contracts in the DLT, since it is not possible to independently release content keys or decrypt messages.

In our proposal, the DLT nodes are in charge of enforcing the access rights that are specified in the smart contracts ACLs. We take advantage of the high degree of trust that a DLT offers for the data written in the ledger, and therefore focus on the trust given to the entities that have to read this data and follow the correct policy. Indeed, DLT nodes rely on the ACLs to make so that the entitled data consumer can obtain the content key, and thus access the piece of data. In order to provide complete data protection to the data subject, only the entitled recipient of health data must obtain the key and not DLT nodes. For this reason, we make use of a (t, n)-threshold scheme to share content keys among the network, and in particular, shares of the content key's capsule. When a data consumer with keypair (pk_c,sk_c) is entitled to access some data in a smart contract ACL, he requests the release of the associated capsule to some DLT nodes through a message signed with pk_c. Upon consumer request, the DLT nodes check if this one is entitled to through interaction with the smart contract. If this is the case, i.e. the data consumer is on the ACL, each DLT node starts the operation for releasing the part of the capsule that was shared with him previously by the data owner. Once the data consumer gets all the shares of the capsule, their reconstruction provides the key k needed to decrypt the desired data stored in the DFS.

We refer to a Threshold Proxy Re-Encryption (TPRE) scheme for the data capsule distribution. The capsule, initially obtained from the pk_{KEM} by the data owner, can be re-encrypted by each contacted DLT node using a re-encryption key $pk_{O \to C}$ generated by the owner. The re-encrypted capsule, then, can be decrypted using sk_c by the consumer to obtain the k_{DEM} needed to decrypt the data. TPRE offers more guarantees rather than a simple PRE scheme that usually involves only one semi-trusted proxy node. One proxy node only can collude with the consumer to attack the data owner's private key. TPRE, instead, uses a (t, n)-threshold scheme to produce "re-encryption shares" in such a way that these can only be combined client-side by the data consumer and not by any $t - 1$ subset of proxies.

V. Health Data Use Case

Nowadays most information is collected through digital channels and applications. Due to the regulatory frameworks, health data generally resides in centralized locations. Mobile personal devices generally perform data pre-processing onboard, with the goal of obscuring personal information, and favouring pseudonymization [26]. In most cases, data is not directly accessible and, when it is, then takes the form of open datasets, where anonymization techniques are applied to completely remove any link to the individuals who generated it and potentially reducing the overall information.

In this work, we focus on sensitive health data produced by a mobile devices in which the collection process is known.

A. Traditional vs. DLT-based Healthcare

Traditional healthcare infrastructures are mostly self-managed or assigned to a third party. These traditional systems generally impact data accessibility, as the provider is not willing to share them for security reasons. The infrastructure to protect sensitive data is built-in trusted infrastructure using techniques such as encryption for fine-grained access control. Moreover, the amount of information from users' mobile devices is usually not collected, and even if it is, it is progressively more difficult due to the scalability of data storage and growing concerns about privacy, security, and infrastructure costs [28].

DLT-based systems work with a different philosophy behind, including individual health data. The system we propose falls in this category. The proposed decentralized health data sharing architecture offers the possibility of transacting data between institutions guaranteeing provenance and immutability. Our architecture aims to put a focus on decentralized authentication and authorisation of individuals' health data. The DFS is responsible for facilitating the data sharing and providing secure information storage, while smart contracts are used to reach consensus, enabling secure access, processing, and sharing of medical data among diverse e-health entities.

B. Internet of Medical Things Scenario

With the advent of big data, every data collected could be used for enhancing new medical studies and personalised medicine. Patient's health data is generally composed of two main parts: general personal information and medical health records. Examples of personal information include age, gender, and weight, while medical health records depend on the topic, i.e. medications, treatments.

In this work, we consider a platform collecting postural stability data where the patient is storing sensitive personal data along with the results of the measurements he performs [29]. The detail of these data can be found in Table I. We consider a scenario where a system user, namely Alice, collects her data through her mobile device. We refer to her as the data owner. Another user of the system is her physiotherapist Bob, i.e., the data consumer. The idea is to share Alice's health data collected with Bob that can use to provide a better medical evaluation for Alice. Thus, Alice will first send her encrypted data to the DFS and contact the ACL to record all the necessary information for third party authorization. She distributes the keys to the DLT network to authorize her doctor. The DLT will retain the authorizations over time and serve as evidence of the transaction. Once the process is successful, Bob is authorized and has the opportunity to look for Alice's health data. He will leverage the ACL in the opposite way of Alice, by recovering all the distributed parts from the DLT and decrypting the original health data.

TABLE I
Example of Health Data

Health Data	
Sensitive Data	Measurement Data
Age	Stabilogram
Gender	Time Domain Features
Weight	Frequency Domain Features
Postural Problems	Structural Features

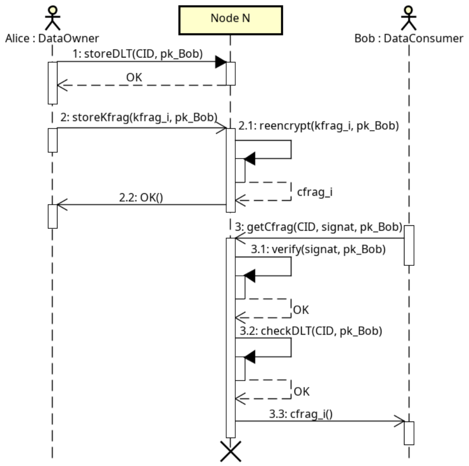
Alice : DataOwner
Node N
Bob : DataConsumer
1: storeDLT(CID, pk_Bob)
OK
2: storeKfrag(kfrag_i, pk_Bob)
2.1: reencrypt(kfrag_i, pk_Bob)
cfrag_i
2.2: OK()
3: getCfrag(CID, signat, pk_Bob)
3.1: verify(signat, pk_Bob)
OK
3.2: checkDLT(CID, pk_Bob)
OK
3.3: cfrag_i()

Fig. 2. UML Sequence Diagram showing the main operations carried out during the testing by the simulated actors.

VI. Performance Evaluation

This section describes the evaluation of the implemented architecture along with the executed tests. Each layer described in Section IV is described in the following:

- The DLT Layer deployed is based on the Hyperledger Fabric Framework [10]. It is a permissioned ledger where all the participant's identities are known and authenticated. The smart contract is implemented by exploiting the Fabric's Chaincode.
- The Access Control Layer includes a smart contract, a TPRE scheme and a key distribution mechanism. Chaincode allows storage and retrieval of relevant information to the access mechanism in a shared and immutable way. Furthermore, the TPRE scheme and keys distribution are built using the Rust language and are based on the Umbral protocol [30].

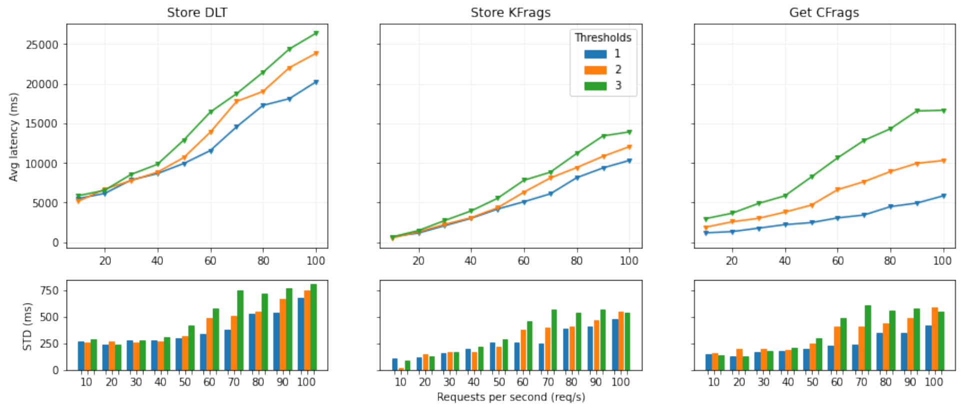

Fig. 3. Average latency per operation.

- The DFS layer is based on the IPFS technology, which allows storing and accessing data on the IPFS network in a persistent but not permanent condition, which means that data stored on IPFS can eventually be deleted.
- The Broker Layer module exposes an API through which the users can interact with the system.

The tests and datasets can be found in [31]. We do not go through DFS performances as this is already documented in previous works and it is not the main focus of this work, i.e. [13].

A. *Network Setup*

In order to perform the tests, we simulated the issuing of new keys and the access to these from several data consumers. Specifically, we simulated a set of data owners injecting new capsules into the system and a variable number of data consumers wanting to access these capsules. The simulation starts with a client acting both as a data consumer and owner (a personal computer with internet access) and interacting with the real DLT network. The network deploys four nodes geographically distributed: two of them in Europe, while the other two in the USA and China respectively, the idea is to represent a real case scenario. The virtual private servers used as node instances have the following specification: two cores, 4 GB of RAM, 50 GB storage and run Ubuntu 18.04 LTS. For more information, one of the four is responsible for syncing the ledger. In fact, the permissioned DLT deployed requires a node for transaction ordering because of its deterministic consensus algorithm.

B. *Testing Workflow*

The testing flow needs several pre-processing steps. First, it is necessary to configure the environments of all the simulated entities. Then, for each actor, a set of asymmetric keypairs, e.g., $(pk_B,\, sk_B)$, is created for encrypting/decrypting data and for digitally signing. A piece of data is encrypted for a data consumer and the associated capsule (see Section IV-D1) is created and distributed to the DLT nodes. This operation is independent from any data consumer request.

The foremost step is executed in parallel for each simulated data consumer. This step consists of a request composed of three primary operations shown in Fig. 2:

- **StoreDLT** - it is the operation where a data owner indicates to a DLT node to add a public key pk_B to the ACL in the smart contract for a specific CID, i.e., the owner instructs the DLT nodes to give access to the data represented by the CID to the consumer pk_B.
- **StoreKfrags** - it consists of a series of methods that perform the actual key distribution (see Section IV-D2). During the pre-processing step, a capsule is created for each piece of data shared. The data consumer uses the capsule to create a fragmented re-encryption key, following the (t, n)-threshold scheme. The re-encryption key is unique for each data consumer. The single re-encryption key fragments are unique for each DLT node. We call these key fragments "kfrags" for simplicity. Each of the n DLT nodes, thus, receives a unique *kfrag*$_i$ and can perform a re-encryption for the indicated pk_B (step 2.1 in Figure 2). The result is a fragment of another capsule that will be used by the data consumer. We call these capsule fragments "cfrags" for simplicity.
- **GetCFrag** - The data consumer requires at least t *cfrag*s to reconstruct the capsule needed for the decryption. Thus, it performs a remote procedure call using the *getCFrag* operation, to t DLT nodes. It also provides in such requests the signature of a message as a way to authenticate itself (in this step, we skipped the whole challenge-response mechanism in which the server, i.e., the DLT node, sends to the client, i.e., the data consumer, a challenge message with a nonce to sign). Each DLT node autonomously verifies the signature (step 3.1 in Figure 2) and checks if the related pk_B is present in the ACL related to the indicated CID (step 2.2). If so, it returns the unique *cfrag*$_i$ to the data consumer.

The final post-processing step involves each data consumer aggregating the $cfrag$s, obtaining the content key and decrypting the piece of data.

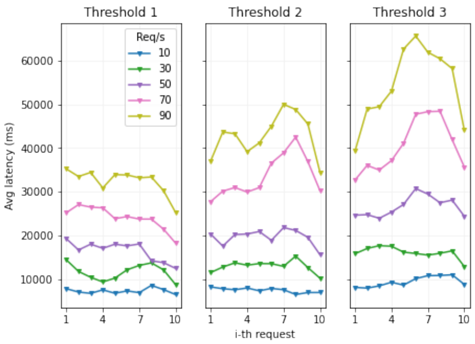

Fig. 4. Average Latency per i-th request step and requests per second.

C. Parameters and Metrics

We observed the following parameters and metrics in the testing:

- **Controlled parameters** - the number of DLT nodes n was set to 3. The number of independent tests was set to 3, and in each test, the main step described previously was repeated 10 times for each data consumer (from now on, this main step will be referred to as *request*). In this case, the time between a request and the next one was given by a Poisson Process with a mean $\lambda = 1000ms$.
- **Independent parameters** - the *threshold* t varies in the tests from 1 to 3. The number of *requests per second* depend on the data consumers, that vary from 10 to 100, with an increase of 10 each time.
- **Dependent metrics** - the *latency* for a response to a *request* is the measure we are interested in. As well as the *latency* in encryption, decryption and *kfrags* generation operations.

D. Results

We recorded the latency for each operation, including the latency of network transmissions. Only the kfrags generation and encryption/decryption latencies do not include network transmissions' latency. Moreover, no errors were recorded during the whole set of tests.

1) Requests per second: Recall that a request is the execution in sequence of *StoreDLTs*, *StoreKFrags* and *GetCFrags*. Thus, Figure 3 shows the results for each operation when the request per second is increased for different values of t. In general, results show a strong dependence on the requests per second value and also on the t value, but the three operations behave differently. Moreover, we see a clear inflection point after 40 requests per second, especially for *StoreDLTs* and *GetCFrags* operations. The *GetCFrags* operation (rightmost plot in Figure 3) is the one where the difference in the three thresholds curves' spread is more evident. This is because the operation heavily depends on t, i.e. the data consumer makes a request to t nodes. In the other two operations, the effect of t is indirect because the number of nodes to which a request is made is fixed.

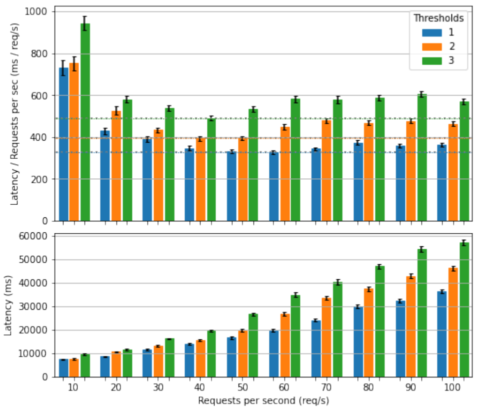

Fig. 5. Average Latency per requests per second.

2) Threshold value: Figure 4 shows the results when increasing the t value and the requests per second for each i-th request, i.e. it shows the performances for each subsequent request instead of aggregating all requests through their mean. In this case, results show how the increase of t amplifies the response delay due to the increase in the requests per second. Specifically, this temporal point of view shows that a low t value (i.e., $t = 1$) keeps the response latency almost stable, while a higher t causes an accumulation of delay in the response, which worsens the performances (i.e. with $t = 2, 3$, from the 5-th request to the 9-th one).

3) Scalability: Figure 5 shows the results for the total average latency of all operations when the requests per second increase. The plot at the top normalizes the latency for the number of requests per second made to the network, i.e. the recorded average latency is divided by the requests per second. This gives a measure of scalability, meaning that when increasing the requests per second, the normalized latency values should remain equal to the previous (or best performing) step in an ideal scenario (the dotted lines in Figure 5 show the minimum normalized latency for each t). More in general, we obtained a linear dependency on the number of requests made concurrently. The optimal-case scenario is deducted by considering latencies below 20 seconds on average, which seems can be reachable when we set $t = 2$ and 50 data consumers. In this case, in the network of 3 DLT nodes, each node handles 16.7 requests per second. In the worst case, the average latency reaches almost 60 seconds, i.e. when the configuration is set to $t = 3$ and 100 data consumers, each DLT node handles 33.3 requests per second. The best-case scenario (in terms of acceptable request-response delay) seems to happen when each node handles about 13.3 request per second, i.e. 40 data consumers, with a response latency ranging between 13 and 19 seconds, depending on the threshold.

Finally, we focus on two operations executed only once per key or payload and happen only on the data owner or data consumer node. Thus we measured these without considering network transmission. Results show a linear dependency of the *kfrags* generation on the t value that does not cross the 100ms even at higher thresholds (50) and denoting an exponential behaviour for the encryption and decryption operations, leading to a system under pressure when the payload's dimension overcome the 10kB.

VII. Conclusion

The management, protection and sharing of sensitive health data are crucial in enabling personal care and contributing to worldwide medical advancements. Health data exploitation could improve patient care and enhance the delivery of health care services. However, besides the benefits, there is widespread concern that patients' privacy and security of medical data could be compromised. In this paper, we introduced a DLT-based access mechanism based on DLT, which could be used to provide more robust security while preserving data protection. The system could ensure that patients have complete access control to their records, stored securely on DFS and only verified participants can interact with patients' sensitive data. Implementing the ACL on the smart contract helps securely share health information among all parties on the network while providing patients' data protection. The experimental evaluation of the overall architecture shows the viability of implementing practical DLT-based healthcare solutions over decentralized systems. Specifically, our configuration handles 40 requests per second with a reasonable response time in the best-case scenario.

As future work, we plan to deploy our solution in a network with a greater number of nodes to test its scalability further. In addition, we are interested in exploiting the expressiveness of smart contracts to encode more complex data access policies.

References

[1] W. Christl, K. Kopp, and P. U. Riechert, "How companies use personal data against people," *Automated Disadvantage, Personalized Persuasion, and the Societal Ramifications of the Commercial Use of Personal Information. Wien: Cracked Labs*, 2017.

[2] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for secure ehrs sharing of mobile cloud based e-health systems," *IEEE access*, vol. 7, pp. 66792–66806, 2019.

[3] G. Bigini, V. Freschi, and E. Lattanzi, "A review on blockchain for the internet of medical things: Definitions, challenges, applications, and vision," *Future Internet*, vol. 12, no. 12, p. 208, 2020.

[4] P. De Hert, V. Papakonstantinou, G. Malgieri, L. Beslay, and I. Sanchez, "The right to data portability in the gdpr: Towards user-centric interoperability of digital services," *Computer law & security review*, vol. 34, no. 2, pp. 193–203, 2018.

[5] European Commission, "A european strategy for data," 2020.

[6] M. Furini, S. Mirri, M. Montangero, and C. Prandi, "Privacy perception when using smartphone applications," *Mobile Networks and Applications*, vol. 25, p. 1055–1061, June 2020.

[7] M. Zichichi, S. Ferretti, and G. D'Angelo, "A distributed ledger based infrastructure for smart transportation system and social good," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, 2020.

[8] M. Zichichi, S. Ferretti, and G. D'Angelo, "A framework based on distributed ledger technologies for data management and services in intelligent transportation systems," *IEEE Access*, pp. 100384–100402, 2020.

[9] Z. Yan, G. Gan, and K. Riad, "Bc-pds: protecting privacy and self-sovereignty through blockchains for openpds," in *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 138–144, IEEE, 2017.

[10] M. Davari and E. Bertino, "Access control model extensions to support data privacy protection based on gdpr," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 4017–4024, IEEE, 2019.

[11] P. De Filippi, C. Wray, and G. Sileno, "Smart contracts," *Internet Policy Review*, vol. 10, no. 2, 2021.

[12] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[13] M. Zichichi, S. Ferretti, and G. D'Angelo, "On the efficiency of decentralized file storage for personal information management systems," in *Proc. of the 2nd International Workshop on Social (Media) Sensing, co-located with 25th IEEE Symposium on Computers and Communications 2020 (ISCC2020)*, pp. 1–6, IEEE, 2020.

[14] M. Jemel and A. Serhrouchni, "Decentralized access control mechanism with temporal dimension based on blockchain," in *2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*, pp. 177–182, IEEE, 2017.

[15] G. Zyskind, O. Nathan, *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, pp. 180–184, IEEE, 2015.

[16] M. Zichichi, S. Ferretti, G. D'Angelo, and V. Rodríguez-Doncel, "Personal data access control through distributed authorization," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pp. 1–4, IEEE, 2020.

[17] J. P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc: Role-based access control using smart contract," *Ieee Access*, vol. 6, pp. 12240–12251, 2018.

[18] D. D. F. Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *IFIP international conference on distributed applications and interoperable systems*, pp. 206–220, Springer, 2017.

[19] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[20] M. Finck and F. Pallas, "They who must not be identified—distinguishing personal from non-personal data under the GDPR," *International Data Privacy Law*, vol. 10, pp. 11–36, 03 2020.

[21] M. Koscina, D. Manset, C. Negri, and O. Perez, "Enabling trust in healthcare data exchange with a federated blockchain-based architecture," in *IEEE/WIC/ACM International Conference on Web Intelligence-Companion Volume*, pp. 231–237, 2019.

[22] 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. 193102–193115, 2020.

[23] I. A. Omar, R. Jayaraman, K. Salah, M. C. E. Simsekler, I. Yaqoob, and S. Ellahham, "Ensuring protocol compliance and data transparency in clinical trials using blockchain smart contracts," *BMC Medical Research Methodology*, vol. 20, no. 1, pp. 1–17, 2020.

[24] T. M. Fernández-Caramés, I. Froiz-Míguez, O. Blanco-Novoa, and P. Fraga-Lamas, "Enabling the internet of mobile crowdsourcing health things: A mobile fog computing, blockchain and iot based continuous glucose monitoring system for diabetes mellitus research and care," *Sensors*, vol. 19, no. 15, p. 3319, 2019.

[25] M. Abouali, K. Sharma, O. Ajayi, and T. Saadawi, "Blockchain framework for secured on-demand patient health records sharing," in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0035–0040, 2021.

[26] European Union Agency for Cybersecurity, "Data Pseudonymisation: Advanced Techniques & Use Cases," tech. rep., European Union Agency for Cybersecurity, 2021.

[27] J. Herranz, D. Hofheinz, and E. Kiltz, "Kem/dem: Necessary and sufficient conditions for secure hybrid encryption," *IACR Cryptology ePrint Archive*, 2006.

[28] J. Indumathi, A. Shankar, M. R. Ghalib, J. Gitanjali, Q. Hua, Z. Wen, and X. Qi, "Block chain based internet of medical things for uninterrupted, ubiquitous, user-friendly, unflappable, unblemished, unlimited health care services (bc iomt u 6 hcs)," *IEEE Access*, vol. 8, pp. 216856–216872, 2020.

[29] G. Bigini, V. Freschi, and E. Lattanzi, "Decentralising the internet of medical things with distributed ledger technologies and off-chain storages: A proof of concept," in *EAI GOODTECHS 2021*, vol. 401, Springer Nature, 2021.

[30] D. Nunez, "Umbral: A threshold proxy re-encryption scheme," 2018.

[31] M. Zichichi and G. Bigini, "miker83z/web5-health-data-sharing-tests: decentralized health data sharing tests," June 2022.