Cluster Computing (2022) 25:4515–4542
https://doi.org/10.1007/s10586-022-03691-3

Data governance through a multi-DLT architecture in view of the GDPR

Mirko Zichichi[1,2] · Stefano Ferretti[3] · Gabriele D'Angelo[2] · Víctor Rodríguez-Doncel[1]

Received: 15 April 2021 / Revised: 5 July 2022 / Accepted: 13 July 2022 / Published online: 10 August 2022

Abstract
The centralization of control over the processing of personal data threatens the privacy of individuals due to the lack of transparency and the obstruction of easy access to their data. Individuals need the tools to effectively exercise their rights, enshrined in regulations such as the European Union General Data Protection Regulation (GDPR). Having direct control over the flow of their personal data would not only favor their privacy but also a "data altruism", as supported by the new European proposal for a Data Governance Act. In this work, we propose a multi-layered architecture for the management of personal information based on the use of distributed ledger technologies (DLTs). After an in-depth analysis of the tensions between the GDPR and DLTs, we propose the following components: (1) a personal data storage based on a (possibly decentralized) file storage (DFS) to guarantee data sovereignty to individuals, confidentiality and data portability; (2) a DLT-based authorization system to control access to data through two distributed mechanisms, i.e. secret sharing (SS) and threshold proxy re-encryption (TPRE); (3) an audit system based on a second DLT. Furthermore, we provide a prototype implementation built upon an Ethereum private blockchain, InterPlanetary File System (IPFS) and Sia and we evaluate its performance in terms of response time.

Keywords Distributed Ledger Technology · GDPR · Smart Contracts · Personal Data · Decentralized File Storage · Data Governance

1 Introduction

The control, direct or indirect, that individuals currently exercise over their personal data is conditioned by the centralized platform-based personal information management techniques, which are then concentrated in a few internet service providers (ISPs) for the purpose of exploring, filtering and obtaining data of interest [1]. The lack of control by individuals over access to their data is of growing concern and, as a result, several regulations have been enacted with the aim of addressing this need. The General Data Protection Regulation (GDPR) [2] is a principal example, designed for European citizens to help promote a view in favor of the interests of individuals, instead of large corporations. It has been followed by other regulations around the world, such as the California Consumer Privacy Act [3] in the USA. The GDPR conveys data control by imposing a number of accountability measures on the responsible actors and by assigning a set of rights to individuals, i.e. as "natural persons should have control of their own personal data" (Recital 7). Dedicated technologies can help either companies to comply with GDPR (and similar) and individuals to exercise their rights, with particular regard to address two main issues: the lack of transparency in the management of personal information and the inability to access and make interoperable personal data.

✉ Stefano Ferretti
stefano.ferretti@uniurb.it

Mirko Zichichi
mirko.zichichi@upm.es

Gabriele D'Angelo
g.dangelo@unibo.it

Víctor Rodríguez-Doncel
vrodriguez@fi.upm.es

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain

[2] Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

[3] Department of Pure and Applied Sciences, University of Urbino "Carlo Bo", Urbino, Italy

The first step toward this aim might rely on the use of a new user-centred model for managing personal data, where storage is decoupled from the application logic, i.e. a personal information management system (PIMS) [4, 5]. Through this, individuals can decide at a granular level what to do with their data and companies can prove their compliance with regulations. This vision is not only beneficial for the privacy needs of the individual, but also for building a single data market [6] that capitalizes on the data interoperability in data spaces for the social good [7–9]. To this end, the European Union Council has recently approved the "Data Governance Act" [10] that aims to promote the availability of data for use, increasing trust in data intermediaries and strengthening data sharing mechanisms across the European Union (both personal and non-personal data). "*Data intermediation services could include bilateral or multilateral sharing of data or the creation of platforms or databases enabling the sharing or joint use of data, as well as the establishment of specific infrastructure for the interconnection of data subjects and data holders with data users*" [10]. This vision for encouraging a new market in neutral data intermediation services is in line with the work we intend to present in this paper.

In this work, we stem from this vision and propose a decentralized approach, based on distributed ledger technologies (DLTs) and decentralized file storage (DFS), to manage data sharing and access. The resulting system is compliant with the GDPR requirements, protects users' personal data and thus promotes data sharing, as intended by the Data Governance Act. Mutual trust is obtained through the use of smart contracts, data custody, systems for access authorization and traceability of personal data. The main benefit of such a decentralized architecture is that, by bringing together regulations and technologies, it provides individuals with the ability to record their data in some interoperable personal data spaces (PDS) [6], guarantees data sovereignty and enables users to control what personal data they want to share [11–14]. PIMS and PDS can be built through the use of decentralized services (e.g. DFS), but even transparent (e.g. DLT-based) services offered by ISPs or 'data altruism organisations', i.e. legal entities operating on a not-for-profit basis and carrying out activities related to 'data altruism' [10].

The use of DLTs and DFS is of paramount importance in our system architecture. In fact, DLTs provide the technological guarantees for trusted data management and sharing, as they can offer a fully auditable decentralized access control policy management and evaluation [15]. In the view of the GDPR, this makes it possible to check whether the involved actors comply with the regulation or not. As concerns DFS, its combined use with DLT allows overcoming the typical scalability and privacy issues of the latter, while maintaining the benefits of decentralization [16]. In practice, DFS are leveraged for storing the actual data outside the DLT, i.e. by means of "off-chain" storage, and tracing all the data references in the DLT (i.e. "on-chain").

The original contributions and novelties of our work are described in the following:

- First, we provide an interdisciplinary analysis of technical and non-technical drivers for the design of a PIMS. In particular, in the background, related work and architecture description, we refer to the GDPR and work/analyses related to this.
- Second, we provide the description of a novel PIMS, based on a multi-DLT compliant design. This system is composed of different components, that are all discussed in this work. We propose a PDS component based on the use of DFS, that is an evolution of the architecture we proposed in [14]. Then we propose a component for the secure control of access to personal data as an evolution of a system that we already discussed in [13]. These two components are aggregated through a novel multi-DLT system, where a permissioned DLT provides the authorization mechanism and a permissionless DLT provides auditability.
- Third, we provide a prototype implementation of the described system and we evaluate its performance by means of an experimental evaluation. More specifically, the implementation is based on an Ethereum private blockchain [17] and a client application for communicating with two DFS, i.e. InterPlanetary File System (IPFS) [18] and Sia [19].

The remainder of this paper is organized as follows. In Sect. 2 the background concepts behind the proposed architecture and related works are presented. Section 3 has the purpose of providing an overview of the system we propose. In Sect. 4 we specify system's architecture and components, then we discuss its GDPR compliance and the security and privacy analysis. In Sect. 5 performance is evaluated and conclusions are presented in Sect. 6.

2 Background and related work

In this section, we describe the technologies that will be used for building up the proposed software architecture, and we provide an introduction to the GDPR norms and terms, which is needed to better understand some of the specific design choices we made.

2.1 Distributed ledger technologies

Distributed ledger technologies (DLTs) consist of a set of protocols and components that guarantee untampered data

availability thanks to the immutable persistence of data in the distributed ledger. DLTs, born with the advent of the Bitcoin blockchain [20], replicates the ledger among nodes of a peer-to-peer (P2P) network. This append-only ledger is expanded through transactions that are disseminated throughout the network, and that are independently verified by each node in order to ensure their consistency. This protocol allows the exchange of data, currency or assets without the need to rely on a human intermediary. In particular, DLTs enable: (1) transparency, i.e. the guarantee for the auditability of transactions and data accesses [15, 20]; (2) security, i.e. the shifting of the trust, that is normally placed to intermediaries, such as ISPs, towards a distributed consensus mechanism [21–23]; (3) immutability, i.e. the verifiability of the data stored in the ledger [20, 24]; (4) decentralization, i.e. the ability of direct user-to-user interactions and agreements, without intermediaries [22].

There are several DLT implementations, all with their pros and cons; however, all of them are built on a network of peer nodes that maintain the distributed ledger. Firstly, implementations can be subdivided into "public" and "private" DLTs. The former type consists of a DLT where anyone can have full access and read the data stored in the ledger, while in the latter the ledger data is private. A hybrid, probably less common, model is the "semi-private" DLT, which can be used in scenarios where a private part of the ledger remains internal and shared among known participants, while a public part can still be used by anyone [25]. Secondly, we can distinguish between two main DLT categories: "permissionless", i.e. where anyone can participate in the consensus mechanism, and "permissioned", i.e. where one or more authorities act as a gate for the participation of new nodes to the consensus mechanism. Bitcoin [20] and Ethereum [22] are examples of public permissionless DLTs, while Hyperledger Fabric is an example of a (semi-)private permissioned DLT [23]. A permissioned solution is often a very convenient approach since it makes it easier to compose a software architecture. Nonetheless, a consortium of trusted entities is usually required in order to employ a permissioned DLT. These entities can also act as certificate authorities that release public and private keys to access the ledger [26]. Obviously, such a solution requires trusting such a consortium [27], while, in contrast, a permissionless approach is more suitable to enable trustless services. Furthermore, DLTs can also be distinguished from their ability to support smart contracts.

2.1.1 Smart contracts

An immutable set of instructions whose execution is calculated deterministically by all (or several, depending on the protocol) peers in the DLT network is embraced by the definition of smart contract. Each node executing the instructions receives the same inputs and produces the same outputs, thanks to a shared protocol. Hence, these properties allow the issuer of a smart contract not to require the presence of a trusted human third party validator to check the terms of an agreement (which is why the term contract is used). However, since it consists of executable code, the issuer must also be sure that the behaviour implemented is correct (e.g. through code verification). In Ethereum [22], the smart contract is a set of instructions and a state, where the latter is modified by means of transactions that enclose data and references to the former. The state evolution is completely traced in the ledger. These contracts can be considered trustless based on the assumption that the majority of participant nodes are honest and follow the Ethereum protocol. In particular, this protocol allows computing (*quasi*-)Turing-complete programs, i.e. smart contracts, capable of processing any type of calculation where steps are bounded. A price, measured in a unit called "gas", is associated with each smart contract execution, and a gas limit is imposed to avoid infinite computation [22].

2.2 Decentralized file storage (DFS)

In order to overcome the typical DLTs' scalability and cloud services' privacy issues, decentralized file storages (DFS) are a potential solution for storing files while maintaining the benefits of decentralization. They offer higher data availability and resilience thanks to data replication. DFSs are crucial for DLTs, as they can be leveraged to store data outside the DLT, i.e. off-chain, when the consensus mechanism discourages on-chain storage.

2.2.1 IPFS

The InterPlanetary File System (IPFS) [18] is a DFS and a protocol that provides a distributed file system over a P2P network. The purpose of IPFS is to provide a resilient and single-point-of-failure-resistant storage system for sharing data, which does not depend on mutual trust between network peers. In the network, files are represented by IPFS objects and are identified by a CID (content identifier), i.e. the digest produced when a hash function is applied to a file. This hash digest, or CID, is also used to retrieve the referenced IPFS object.

An important remark to make is that peers in the IPFS network have no incentive to maintain objects when asked to replicate them. A peer maintains a replica of an object until it needs to free up space in its local storage (a process called "unpinning").

2.2.2 Incentivized file storage and Sia

In order to maintain greater reliability and ensure that the file can be correctly retrieved, an incentive mechanism can be placed on top of a DFS. Filecoin [28] is an incentive layer on top of IPFS where participants are rewarded (with Filecoin tokens) for serving and hosting content on their storage. The protocol matches client requests with storage node offers through the use of a blockchain and dedicated smart contracts. Besides Filecoin, other solutions exist that provide incentives to persistently store data. Such an example is Sia [19]. It consists of a DFS that also leverages smart contracts, i.e. file contracts, to arrange an agreement between storage providers and clients.

While IPFS has already been considered and evaluated in several studies [29, 30], Sia does not match this level of maturity despite looking very promising.

2.3 Cryptographic access control and keys distribution

Access control is the ability to regulate access to some resources by enforcing permissions established by a set of policies, e.g. discretionary, mandatory, role-based, attribute-based [15]. In cryptographic access control [31], the policy enforcement depends on both security of the underlying cryptographic primitives and appropriate key distribution. A centralized control of data accesses, conveyed through a central access control server and keys distributor, entails the risks of single point of failures and, above all, privacy leakages [32]. On the other hand, a cryptographic access control paradigms built around secret sharing or proxy re-encryption can offer a better guarantee of privacy and security in the key distribution. This is obtained through proper decentralized key exchange mechanisms, as described in the next Sub-Sections and in Sect. 4.3.

2.3.1 Secret sharing

Secret sharing (SS) was first proposed in [33] and [34] by Shamir and Blakley. It consists of a threshold scheme (t, n) in which each participant in a set of n participants owns 1 of the n shares of a secret and any subset of $t \leq n$ participants can reconstruct it. Considering the key to decrypt data as a secret, in a network of n nodes a consensus can be reached by issuing t shares to an eligible data consumer to allow the latter to decrypt the data. Any node is not able to access the data on its own, as it would need the help of other $t - 1$ nodes.

2.3.2 Proxy re-encryption

In a dynamic distributed communication, between an arbitrary number of data owners and consumers, proxy re-encryption (PRE) represents a scalable cryptographic protocol that allows ciphering a datum without the need to know the recipient of that datum in advance. The general definition of PRE [35] consists of a public key encryption protocol that contains a re-encryption phase in which the plaintext is not revealed. Specifically, in this phase a sender (i.e., a data owner) with a key pair (pk_1, sk_1) generates a re-encryption key rk_{1-2} to be sent to a semi-trusted proxy server together with a ciphertext c_{pk_1} encrypted with the public key pk_1. Then the proxy server can run the proxy re-encryption algorithm to generate a new ciphertext c_{pk_2}, containing the plaintext, which is decryptable by a receiver (i.e., a data consumer) with the key pair (pk_2, sk_2). The proxy only uses rk_{1-2} and c_{pk_1}, therefore it has no access to the plaintext. However, it must be semi-trusted because thanks to rk_{1-2} it can re-encrypt any ciphertext with pk_1 in favour of the recipient. A specific instance of PRE is one-way proxy re-encryption, where the re-encryption function is one-way.

2.4 GDPR

The General Data Protection Regulation (GDPR) [2] became applicable in the European Union in 2018 with the aim of protecting the personal data of its citizens. It applies to the processing of all the data recognized as any information relating to an identified or identifiable natural person. The impact of the GDPR is global, because it is considered as the magna carta for regulations on the processing, storage and management of personal data and because it affects any organization that addresses the European market and its citizens [36]. The regulation relies on the interaction between three main actors:

- *Data subject* The natural person identified or identifiable by the data.
- *Data controller* The natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data. The controller plays a central role in the interactions between the various roles, being called into action by the subjects to exercise their rights and being liable in the event of violation of the rules by the data processors.
- *Data processor* The body which processes personal data on behalf of the controller. Processors have their obligations under the GDPR, although they do ultimately report to the data controller.

The regulation follows six core data processing principles, stating that personal data should be: (1) processed lawfully, fairly and in a transparent manner; (2) collected for specified, explicit and legitimate purposes, and only used for those purposes that have been stated; (3) adequate, relevant and limited to what is necessary for the stated purposes (data minimization); (4) accurate and, where necessary, kept up to date (5) kept in a form which permits identification of data subjects for no longer than is necessary, i.e. deleting the data when it is no longer necessary; (6) processed in a manner that ensures appropriate security e.g. against hacks or accidental loss or damage.

2.5 Related work

In related works, many DLT-based frameworks run in a decentralised manner and provide an autonomous and traceable way to manage data. Such frameworks can be leveraged in access control mechanisms to solve problems related to centralization and privacy leakage [32, 37] and to store, share and transmit data securely [8, 38, 39].

In the following, we will describe related works and compare them using Table 1.

2.5.1 DLT-based data sharing

Droplet [40] takes advantage of a DLT to provide data owners the ability to share their data through a secure encryption key derivation and management mechanisms, with a focus on Internet-of-Things generated data. In this study, the DLT is used to hold the keys used for data encryption and their distribution is the responsibility of the data owner. This system (and the key derivation mechanism in particular) is pluggable to alternative solutions such as the one we present in this work. Related to this kind of solution is the work of Jiang et al. [41], where the encryption operations are outsourced. Both works share the use of stealth addresses [42] to provide privacy in authorizations while maintaining traceability. In Blockstack [43], users' transactional metadata is stored in a DLT, while the data itself is stored off-chain, through a cloud service provider (e.g. Google Drive, etc.). Although this technology shares similarities with our data storage proposal, it is not particularly targeted at authorizing access to third parties.

2.5.2 Attribute-based access control

Existing literature provides many DLT-based access control system implementations that are based on attribute-based encryption (ABE) [53]. This solution provides policy expressiveness without introducing many elements into the system infrastructure. ABE, indeed, encrypts the data using a set of attributes that form a policy and only those who have a secret key that meets the policy can decrypt the data. This DLT-based access control through ABE is a specific case of general attribute-based access control (ABAC) where access is given after a policy evaluation based on subjects' attributes, e.g. using the eXtensible Access Control Markup Language (XACML) for enabling the access to a subject having attributes such as "email domain equals to abc.com" [37].

In [44], the authors designed a system using ABE-based access control and smart contracts to grant data access, with similar policies mechanism to our solution, while authors of [45] and [46] propose similar frameworks that combine DFS and blockchains to achieve fine grained ABE-based access control. However, in all three works, the secret attribute keys are issued directly by the data owner in the DLT or by a central authority. This limits data sharing both from security (key immutably stored in DLT) and GDPR (right to data deletion) perspective.

Among these schemes, ABE presents some issues such as privacy leakage from the private key generator [54] and a single point of failure [32, 37]. Moreover, it also presents issues on feasible (in terms of efficiency and security) decentralized key generation and revocation [55]. In our work, we focus on access keys distribution, rather than on policy evaluation, e.g. ABAC. In fact, with the aim to reduce the complexity of the smart contract, we leverage an access control list (ACL) instead of attributes. Thus, our work should be investigated in terms of cryptographic schemes to be employed, such as ABE, PRE and SS.

2.5.3 Personal data sharing

In the broader scope of DLT-based data sharing and access control systems, we may find a subset of personal data management solutions that: (1) require an additional effort to comply with regulations such as GDPR; (2) include architectural components designed following the logic of "Privacy by Design" [56]. The main purpose of a DLT-based system, again, is to provide transparency in the process of accessing personal data, but simultaneously enable users to control their own data [6, 7, 14, 39]. Nevertheless, few studies address GDPR compliance and even these studies do not compare the opinions that regulators have regarding specific DLT architectural parts. DeepLinQ [47] is a multi-blockchain architecture similar to our proposal. It aims to support privacy-preserving data sharing in the healthcare sector through granular access control and smart contracts. Zyskind et al. [48] provide a system in which DLTs are leveraged to hold discretionary access control policies and track user permissions to give or deny access to data. Yan et al. [49] present a PDS that allows users to collect, store and give third parties fine-

Table 1 Summary of the features and comparison of the related works with respect to our work

References	Schemas	Off-chain	On-chain Anonymization	GDPR	Keys Distribution
[40]	Dual-Key Regression + ACL	FS	Yes, Stealth addresses	Not explicit, Possibly compliant	Efficient
[41]	ECC	DFS	Yes, Stealth addresses	Not explicit, Possibly compliant	*Data owner* burden
[43]	ECDSA + Symmetric Encryption	(D)FS	No, Pseudonymous	Not compliant	*Data owner* burden
[44]	ABE	FS	Yes, Attribute-Based Signature	Not explicit, Possibly compliant	Single point of failure
[45]	ABE	DFS	No, Pseudonymous	Not compliant	*Data owner* burden + Keys On-chain
[46]	ABE	DFS	Yes, Identity managed by central auth center	Not explicit, Possibly compliant	Single point of failure
[47]	RBAC	FS	Yes, Multi-DLT architecture	Not explicit, Possibly compliant	Efficient
[48]	ECDSA + Symmetric Encryption (+ Multi-party computation)	DFS	No, Pseudonymous	Not compliant	*Data owner* burden
[49]	Hierarchical SS	No	No, Pseudonymous	Not compliant	Efficient
[50]	ECDSA + ACL	FS	Yes, Private DLT	Compliant	Single point of failure
[51]	ECDSA + Symmetric Encryption	DFS	Yes, Private DLT	Compliant	Single point of failure
[52]	KEM/DEM technique + TPRE	DFS	No, Pseudonymous	Not explicit, Possibly compliant	*Data owner* burden
Ours	KEM/DEM technique SS + TPRE + ACL	(D)FS	Yes, Multi-DLT architecture	Compliant	Efficient

Possibly GDPR compliant means that the provided architectures could be made compliant with low effort, e.g. appoint data controllers for permissioned DLTs nodes. The efficiency in the keys distribution operation is intended with respect to the *data owner*

ECC Elliptic Curve Cryptography, *ECDSA* Elliptic Curve Digital Signature Algorithm, *ABE* Attribute-Based Encryption, *RBAC* Role-Based Access Control, *SS* Secret Sharing; *TPRE* Threshold Proxy Re-Encryption

grained access to their data using a SS scheme. Their solution is innovative, but expensive and not recommended for GDPR because the system stores personal data on-chain, when it is possible to do it off-chain.

2.5.4 GDPR and DLT

Other studies in the literature for GDPR compliance do not address key distribution and primarily focus on programming smart contracts for automatically managing access control policies [30, 57–59]. Truong et al. [50] provide a DLT-based GDPR-compliant personal data management solution where consent is handled through a token and data is stored and served through a Database Management System. Fewer studies provide a system architecture while discussing GDPR and DLT conflicts in depth. A model for tracing the personal data life cycle is proposed by Onik et al. [51], where smart contracts manage consent and terms for the use of off-chain stored personal data. The study of Ahmed et al. [60] focuses on the lack of GDPR compliant consent management mechanisms. They highlight the challenges of DLT under the GDPR and respond to these by presenting some opportunities for fine-grained control over personal data.

2.5.5 Self-sovereign identity

Finally, much of the current interest in the problems of privacy and data sharing aggregates into a specific definition, namely the paradigm of the Self-Sovereign Identity

(SSI). It consists of the complete control of individuals' digital identities and their personal data through decentralization [61]. The architecture we describe in this paper is based on the SSI principles, since these seem to be fully supported by the features of DLTs, in particular peer-to-peer interactions and data integrity. Other scholars, indeed, already produced some contributions leveraging public and/or private DLTs for SSI. Sovrin [62] is an open source identity network built on a layered architecture involving a public permissioned DLT that only stores identity transactions, without personal data, and where only trusted institutions, i.e. banks, universities, governments, can be DLT nodes. uPort [63] is based on Ethereum's public permissionless DLT and provides a platform for user-centric identity and communication, consisting of a mobile app, smart contracts and a set of open protocols for message flow.

3 Multi-DLT architecture for a personal information management system

In this section we provide an overview of the architecture of a multi-DLT-based PIMS, with regards to personal data storage and sharing. Resiliency to manipulation in DLT is a very promising property to develop new types of applications, especially in presence of smart contracts, as in Ethereum [8, 22, 29, 39]. However, the ability to run smart contracts usually comes at the cost of lower scalability and responsiveness [64]. At the time of writing, there is no single, operative and fully fledged solution that is able to cover all the needed features [8]. Thus, in order to build a sophisticated (multi-layered) software architecture, different DLTs and/or DFS need to be combined together. We discuss here a brief overview of the architecture we propose, aided by Fig. 1. From now on we will make use of italicized text to identify architecture's components and actors. The components are:

- *User client application*—Generally, we refer to *data owners* and *consumers* as the end users of our PIMS. A dedicated client application allows: (1) *data owners* to decide how/where to store the data and to handle their encryption and secret keys generation, (2) *data consumers* to submit data access requests to the system and to handle data decryption.
- *(Decentralized) file storage*—It consists of the first and also the last component with which the users interact with, since it contains a personal data storage (PDS). It basically contains the data to encrypt or decrypt and can be implemented in different ways, e.g. as a centralized cloud-based storage or as a DFS.
- *Authorization system*—We leverage a (semi-)private permissioned ledger to orchestrate the access control logic and to store the hashes of the data kept in a PDS for integrity validation. This ledger is managed by a set of predetermined *authorization servers* appointed to check the access credentials and to distribute the capsules that contain secret keys to the entitled *consumers*.
- *Audit DLT*—This component is used to provide proof of a correct audit for the (semi-)private permissioned ledger, when one or more *authorization servers* (if not all) act maliciously. It consists, mainly, of a public permissionless ledger where the states of the authorization system are logged.

3.1 Architectural drivers

Several major considerations influenced the architecture of the system we present in this paper. We describe them in the following.

3.1.1 GDPR and DLTs tensions

In particular, we devoted special attention to the tensions between the GDPR and DLTs [65, 66]. When the GDPR was first drafted in 2012, it was mainly influenced by the client/server architectures common for ISPs at that time. This view collides with DLTs, public permissionless ones in particular. In fact, the absence of a centralized authority that determines how and where data is stored, processed or used, complicates the interpretation of some GDPR rules in a completely decentralized scenario. It may be difficult to identify which participants in a DLT are the data controllers, processors or subjects, because often the information does not necessarily flow directly from one point to the other, i.e., users to providers [62]. The tensions between the GDPR and the DLT mainly concern three issues [12, 65, 66]:

- *Actor accountability* As far as the GDPR is concerned, it must be possible to identify a data controller, but it is not clear whether this is possible in permissionless DLTs [65, 66]. In fact, the more concentrated the control of the participating nodes on the DLT is, the more it is possible to identify the controllers; while, the more diffuse the control is, the more it is not possible to provide an identification [62, 65].
- *Personal data processing* So far, there are no major agreements on what is necessary to transform personal data so that the resulting output can be stored in a DLT [67], given the GDPR Recital 26 that states that data becomes anonymous if it is 'reasonably likely' that no identication of a natural person can be derived [68]. For

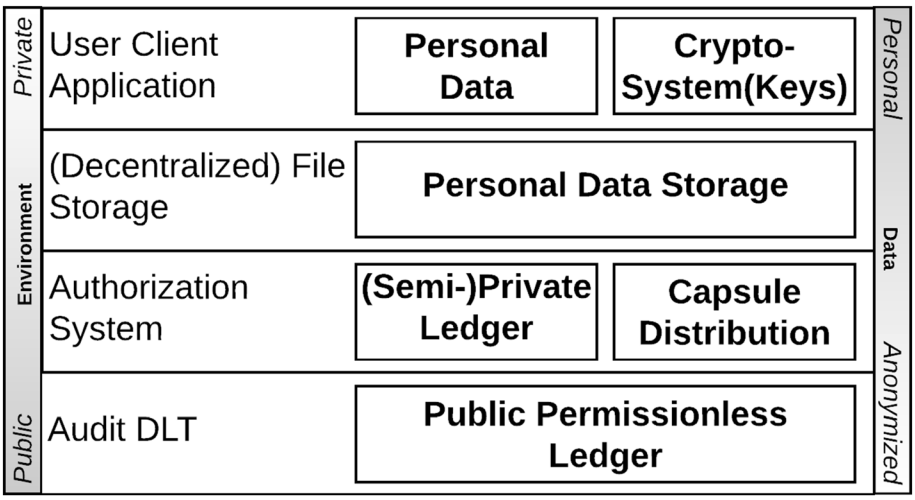

Fig. 1 Layered Architecture of the personal information management system

instance, since DLTs generally use addresses derived from public keys for identifying a person and considering the use of public keys in digital signatures as pseudonymization technique[1] [69], therefore, the address datum is still under the scope of the GDPR. However, applying such techniques to personal data can also reduce risks for data subjects and help controllers and processors meet their data protection obligations, but it depends on a case-by-case basis [5].

- *Data subject rights* The above GDPR principles were used to derive a set of rights for data subjects and obligations for data controllers. Two in particular, the "right to be forgotten" and the "right to rectification" (Articles 16 and 17), are the main breaking points between the DLT and GDPR.

3.1.2 Relation between data protection and cryptography

In this work, the implementation of the guarantees related to the application of the principles established by the GPDR is based both on the application of specific recommended cryptographic techniques and, most importantly, on the organizational and architectural measures taken following the Privacy by Design approach. With regards to Article 32, controllers and processors must comply with the implementation of "appropriate technical and organisational measures, to ensure a level of security appropriate to the risk". Pseudonymisation is an accepted data protection measure in the adoption of the GDPR (Article 4(5)) and many times referenced as a safeguard [5], whilst, anonymization techniques can generally provide strong privacy guarantee [69]. In general, however, providing only pseudonymisation and encryption technical solutions that are formally verified secure is not a sufficient and necessary condition to be able to state that data processing security is appropriate to the risk. One has to tackle the data protection problem from an higher point of view, "going beyond the 'traditional' understanding of security" [70]. In our work we focus on the data protection by design and by default by following the guidelines of different by cyber-security agencies and supervisory authorities, that also indicate appropriate pseudonymisation and encryption solutions [5, 66, 68, 69].

In particular, the use of a mixed symmetric-asymmetric crypto system, advanced cryptographic hash functions, secret sharing schemes and multi-DLT validation (all of them explained in detail in the following sections) has been covered in these recommendations as a form of pseudonymisation and/or anonymization. In the following, we are going to give the definition of a formal model for the cryptographic part of our proposal, which, however, will not be rigorously proved in its entirety. This is because we believe that a formal demonstration of it is outside the scope of our work and would take away space from another very important issue for the security of our system, as seen in this Sub-Section, namely the discussion of how we have implemented Privacy by Design in a decentralized context. We retain the possibility of addressing a full formal proof in a future work fully devoted to this topic.

3.2 Actors in the system

We define different actors that have one or more roles in the system. The focus is obviously on the definition of accountability obligations among the system actors and participants, since it represents a crucial point for the GDPR. In detail, we identify the following actors:

[1] pseudonymisation consists of replacing one attribute in a record by another and is not a method of anonymisation since the person is still likely to be identified indirectly [69].

- *Data owner* (DO)—The one that has the power to decide what to do with the personal data. In accordance with the GDPR terminology, a *data owner* can be either a subject (i.e. an entity that acts as his own controller, as in Self-Sovereign Identity paradigm [12, 71]) or a data controller that received the subject's consent for the processing of personal data.
- *Data consumer* (DC)—The one who has lawful access to certain personal or non-personal data and is authorized (by the *owner*) to use that data. Again, following the GDPR terminology, since the latter is a data controller, the *consumer* is a data processor by definition (Sect. 2.4). It is often referred to as "Data User", e.g. in [10].
- *(D)FS service provider* (SP)—The one that provides the access to the (D)FS. Even if the encrypted personal data are only stored in a (D)FS, this actor is a data controller because such data can be considered as pseudonymous (and not completely anonymous), and thus, in principle, subject to the GDPR [67, 69]. However, in practice, appropriate techniques can be used (Sect. 4.2) so that these providers handle data that are meaningless without additional information (e.g. a decryption key). In such a case, they have no obligations [68]. Finally, this actor could take on the role, defined in the Data Governance Act [10], of a data sharing provider, in that it collects and organizes data regardless of the specific use and applications.
- *Authorization server* (AS)—An actor within the authorization system that has agreed on contractual terms that define precisely the roles and duties and the privacy policy towards end users, i.e. *data owners* and *consumers*. All the *servers* within an authorization system act as joint controllers in a shared responsibility approach [71, 72] and the legal basis for the processing of personal data is guaranteed by mutual agreements. Similarly to *(D)FS service providers*, this actor may take on the role of data sharing provider as well. Moreover, from a data governance perspective, the whole set of *servers* can be thought of as a data altruism organisation operating on a non-profit basis [10].
- *Audit DLT node* (AN)—The one that takes part in the audit DLT consensus mechanism. This actor is external to the PIMS. It is in charge of registering the state updates of the authorization system into the audit DLT, i.e. a DLT full-node [20, 22].

In the following, we are going to refer to actors using the following notation. We consider only one data owner, i.e. *DO*; and one data consumer, i.e. *DC*. Then we consider a SP_j being part of a set of (D)FS service providers $SP = SP_1, ..., SP_m$, with $1 \leq j \leq m$. A AS_i being part of a set of Authorization servers $AS = AS_1, ..., AS_n$, with $1 \leq i \leq n$. And, finally, a AN_f being part of a set of audit DLT nodes $AN = AN_1, ..., AN_g$, with $1 \leq f \leq g$.

4 Architecture components design

In this section, a description of the components of the architecture will be provided. Figure 2 will be used to aid us with the description. While the previous section showed a horizontal view of the architecture, here each component will be examined vertically, with respect to the technology stack, as well as in terms of GDPR compliance.

4.1 User application

Data owners and *consumers* interact with the PIMS through a client application. Such a system component contains the software to communicate with the *(D)FS service providers* and *authorization servers* and, most importantly, it implements cryptographic operations that are used to protect personal data, right at the user's device level. We assume that each actor has its own pair of asymmetric keys (pk_{KEM}, sk_{KEM}) and an example of notation is as follows: pk_{DO} or sk_{AS_i}.

The user client application, thus, implements part of the crypto-system, a critical part of the whole PIMS that crosses vertically all the other subsystems described in our architecture. The core of our crypto-system consists of the use of a hybrid encryption scheme [73]. It consists of a scheme where an asymmetric public key pk_{KEM} is used to encrypt a symmetric secret key k_{DEM} through a key encapsulation mechanism (KEM), and then k_{DEM} is used to encrypt the actual data through a data encapsulation mechanism (DEM) (Fig. 3). This KEM/DEM technique combines the efficiency of symmetric cryptography with the benefits of asymmetric cryptography [73].

A specification is needed, at this point, for what regards personal data created directly by/on behalf of the data subject because the encryption algorithm will depend on this, namely:

- "single" datum - describe a subject's static property (because it never or rarely changes), e.g. the date of birth of the subject.
- "sequential" data - it can be employed to describe time series data or a property that changes in time, e.g. the location of a subject. In this case, each element of the sequence represents a particular value associated with that property at a given time (thus, the data structure logs all the value versions).

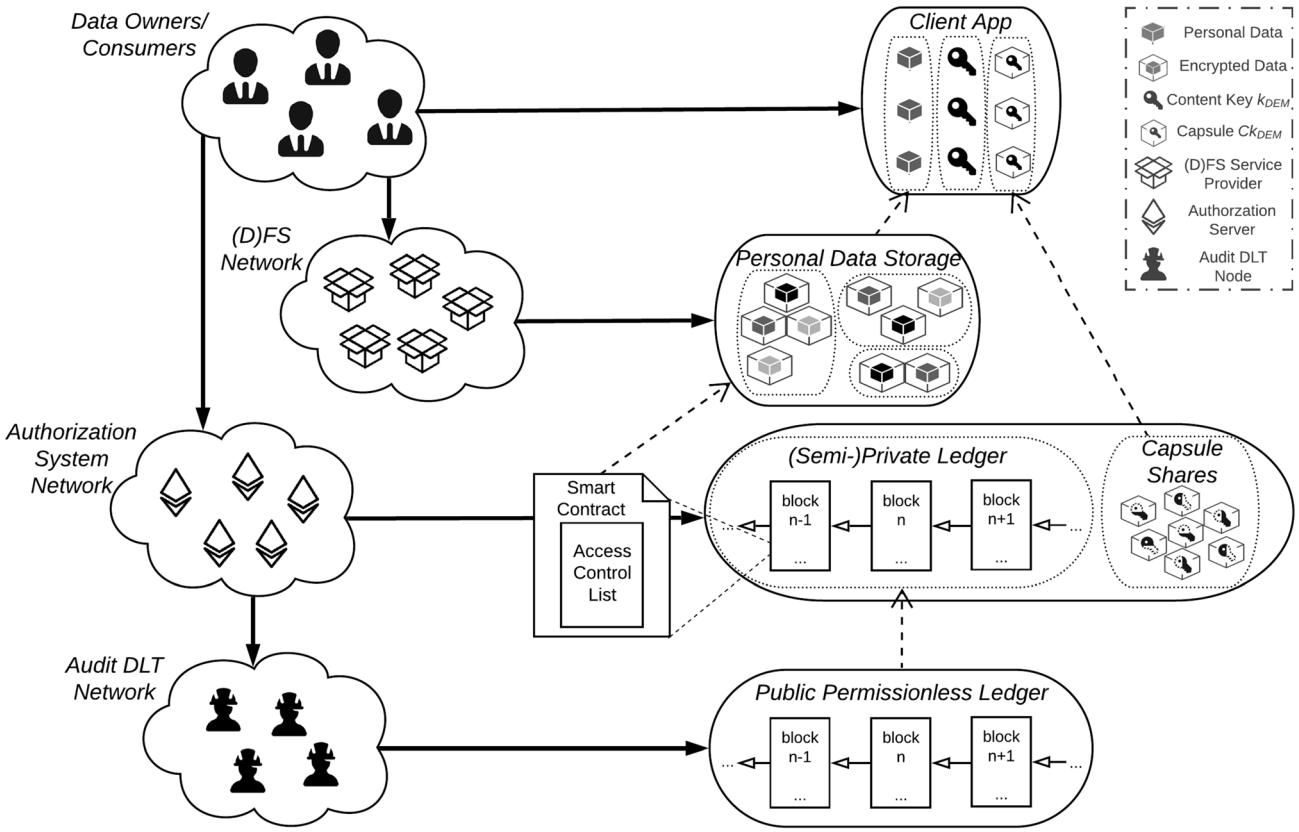

Fig. 2 Architecture of the personal information management system. The top-right legend identifies the elements in the diagram. *Solid arrows* represent interactions from an actor or a set of actors and a system or network. *Dashed arrows* represent hash pointers to elements

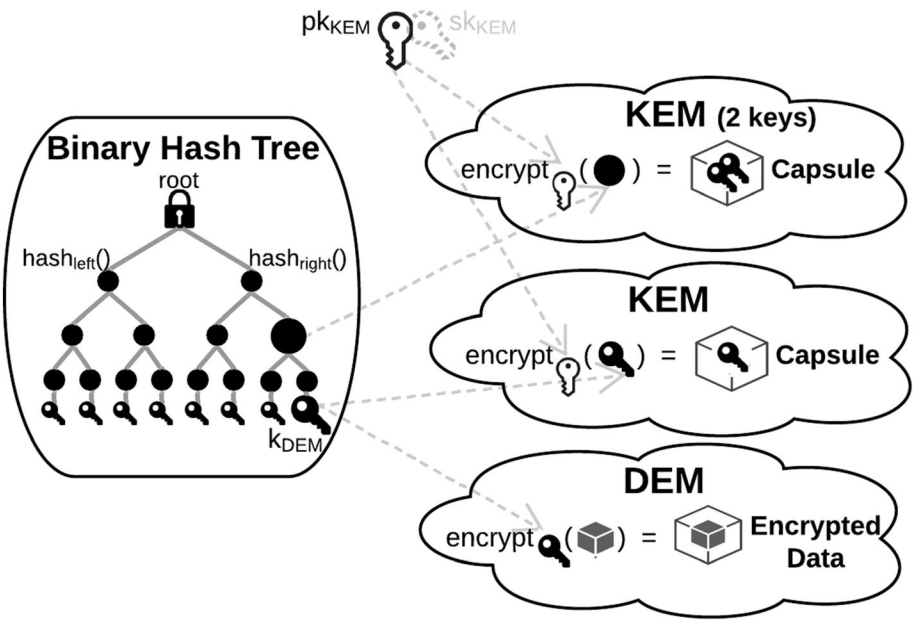

Fig. 3 Data and key encapsulation mechanisms

4.1.1 Crypto-system

Personal data are encrypted using a symmetric content key k_{DEM} (Fig. 3). Then, the key is placed in a "capsule" through a KEM, i.e. the content key is encrypted with a public key in a keypair (pk_{KEM}, sk_{KEM}). This capsule is, then, the representation of the hybrid encryption scheme and consists of $c_{k_{DEM}} = Enc_{pk_{KEM}}(k_{DEM})$.

The content key k_{DEM} associated with a single datum is randomly generated. As concerns sequential data, content keys associated with elements of the sequence are organized as a hash tree, in order to facilitate their management and retrieval. In this case, the root of the tree is a secret seed, while child nodes are constructed top-down through the use of multiple hash functions. For instance (see Fig. 3), a binary hash tree can be constructed starting from a root and then recursively creating a left and a right child using, respectively, a left hash function and a right hash function [40]. Finally, leaf nodes are used to derive the content key k_{DEM} and are assigned to each sequence element in an ordered manner, e.g. time ordered for time series data. In this case, in order to share a sequence interval, the internal tree nodes are encapsulated ($c_{node} = Enc_{pk_{KEM}}(node)$) instead of the content key k_{DEM} of each sequence element. This facilitates *data consumers* to generate the corresponding set of content keys from a single source, i.e. the internal node of the tree.

Up to now, then, we simply consider four sets of elements.

- $PD = \{pd_l \mid 1 \leq l \leq o\}$ is the data owner's set of personal data that have not been encrypted, i.e., plain text.
- $K = \{k_{pd_l} \mid Enc_{k_{pd_l}}(pd_l), 1 \leq l \leq o\}$ is the data owner's set of keys used to encrypt a piece of personal data.
- $EPD = \{epd_l \mid epd_l = Enc_{k_{pd_l}}(pd_l), 1 \leq l \leq o\}$ is the data owner's set of encrypted personal data.
- $C = \{c_{k_{pd_l}} \mid c_{k_{pd_l}} = Enc_{pk_{DO}}(k_{pd_l}), 1 \leq l \leq o\}$ is the data owner's set of capsules that contain a key used to encrypt a piece of personal data.

Notice that we generalized single and sequential data in only one case to make the model simpler to be understood.

4.2 (Decentralized) file storage

In our design, personal data is kept in a PDS associated with a data subject. This PDS consists of the set of encrypted personal data referring to the subject that is stored in a (decentralized) file storage. Such (D)FS can consist of either any commercial cloud FS service (e.g. Azure, Google Drive, etc.) or a decentralized FS service, as no operational logic is required at this level other than storing and obtaining encrypted data.

In the case of a data controller acting on behalf of a data subject, it is more likely that the former will use a private (proprietary) storage solution, instead of a (D)FS, for maintaining a PDS. This is completely compatible with our PIMS, as long as the data content can be referenced and data can be accessed by *consumers*.

4.2.1 Data off-chain & hash pointers on-chain

To guarantee data integrity and verifiability, encrypted personal data could be stored directly on the authorization system's (semi-)private permissioned ledger, i.e. on-chain. However, preventing the on-chain storage is a preferable solution, not only for retaining high data reads availability and better performances for data writes [14], but also because on-chain personal data are generally incompatible with data protection requirements (Sect. 3.1).

We follow the approach to reference data and their content on-chain, e.g. through a hash pointer, and to store them off-chain in a (D)FS [14]. Once a personal datum is hashed the resulting digest can be used as a pointer, i.e. a hash pointer. Thus, the reference, in the form of a hash pointer stored on-chain, allows to retrieve the data and to verify their integrity.

In our model we assume that all (D)FS service providers *SP* store the data owners' set of encrypted personal data *EPD* and the associated set of hash pointers to identify these, i.e. $HP = \{hp_{epd_l} \mid hp_{epd_l} = hash(epd_l), 1 \leq l \leq o\}$. Thus hp_{epd_x} is both the identifier of the epd_x datum in the (D)FS and a hash pointer that will be stored on-chain.

4.2.2 Service providers and data erasure

Taking, for instance, IPFS [18] as a DFS, one can store an (encrypted) datum by making a request to a *(D)FS service provider* that is running the IPFS protocol. Then, such provider will propagate the datum (or parts of it) to multiple nodes in the IPFS network. To retrieve the datum the hash digest is used as reference.

Regarding the deletion of content, it is another main feature to guarantee personal data deletion to a data subject. Thus, we follow the approach presented in [16] for the anonymous delegated deletion protocol, in which the deletion is not completely granted, but reasonable steps to inform IPFS nodes for data deletion can be taken.

4.3 Authorization system

In our architecture we leverage a network of *servers* to provide authorizations to *consumers* for two reasons: i) to

release the *data owner* from the burden of completely handling capsules distribution, which can be very expensive in terms of communication in case of fine-grained access; ii) to exploit smart contract shared computation while complementing it with off-chain capsules distribution mechanisms, since it is not possible to store secret keys or decrypt messages on-chain (due to its public execution).

Authorization servers perform on-chain tasks such as smart contracts executions, but also off-chain tasks such as the capsules distribution. The network uses a (semi-)private permissioned ledger as a sidechain (we will use this term from now on) and the audit DLT as the mainchain [21].

4.3.1 Smart contract access control

Access to the data stored on a PDS can be allowed by the *owner* through smart contracts (Fig. 4). These maintain a data structure to record eligible *data consumers*, i.e. those to whom to issue capsules to access the encrypted data.

In practice, each datum is referenced in a specific smart contract, as already discussed in Sects. 4.1 and 4.2. Thus, the smart contract stores a subset of the hash pointers set in the (D)FS, i.e. $HP^{on-chain} \subseteq HP$. In addition, with regard to the data, the contract also maintains a data schema. This is similar to the Sovrin scheme [62] and uPort claim spec [63], i.e. a machine-readable format for specifying what to expect from the shared data. This is needed by the *consumer* in order to better handle the data computation and it can be defined directly by the *data owner* or can be a specific standard of the authorization system. The smart contract, however, mainly consists of code to manage the data structure representing the access rights to a data packet, i.e. an access control list (ACL). The *data owner* gives consent by adding or removing *consumers* from the ACL. Once a *consumer* is listed in the ACL, i.e. authorized to access a given content, the *authorization servers* can directly verify this information and eventually release the $c_{k_{pd_l}}$ capsule that includes the k_{pd_l} content key needed to decrypt the encrypted data. This process is explained in detail in Sect. 4.3.3.

4.3.2 (Semi-)private ledger

The sidechain of reference for our architecture is Ethereum [22], however multiple authorization systems can be implemented with different configurations or DLTs, e.g. Hyperledger [23]. For instance, the consensus algorithm adopted by the network does not necessarily have to be the Proof-of-Work [20, 22], but can be chosen in order to provide a faster service. We refer to the Proof-of-Authority (PoA) consensus algorithm [74], which does not depend on solving mathematical problems, but to issue a new block this must be signed by the majority of the authorities, i.e. the nodes that are explicitly authorized to create new blocks and secure the blockchain.

Privacy by Design In the sidechain, we have adopted extensive measures to minimize the transmission and/or storage of any personal data through the ledger, following the principles of "Privacy by Design". We define the sidechain as a (semi-)private one because, even if the ledger is private, there must be a way to access some parts of the ledger by users, for instance the smart contract containing the ACL. *Authorization servers*, thus, maintain the whole copy of the ledger, but they give read access rights in three cases:

- In the case of an audit, the whole ledger can be released to the entity in charge of performing it. This can be defined in the agreements between *servers* and *owners*/*consumers*;
- *Data owners* can access their ledger metadata (e.g. Ether balance [22]) and to the data related to their smart contracts in a complete mode, i.e. the hash pointers $HP^{on-chain}$, the data schema and the ACL;
- *Data consumers* can access their ledger metadata and to all the smart contracts in the DLT in a restricted mode, i.e. the related data shown consists of the hash pointers $HP^{on-chain}$, the data schema, owner's address and the hash digest of the ACL only.

For the public access (where the public audience is composed of data consumers) then, since the ACL consists of personal data, it is stored off-chain.

4.3.3 Capsules distribution

Authorization servers act like the ones in charge of enforcing the authorizations for *data consumers* made explicit 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 we concentrate on the trust given to the entities that have to read these data and follow the correct policy. Indeed, the entities that form the *authorization servers* rely on the ACLs to release the access keys to access data. Consider the scenario where a single centralized server stores keys. Not only is this server susceptible to a single point of failure, but worse, it can act as an honest-but-curious provider. For instance, an ISP that correctly follows the protocol to share a user's geolocation data with the user's friends, if curious, can also access this information. Our proposal to decentralize this power in various *authorization servers* allows us to shift the trust from a centralized server to the protocol. Indeed, *authorization servers* may be considered semi-trusted or completely untrusted, but through the data protection

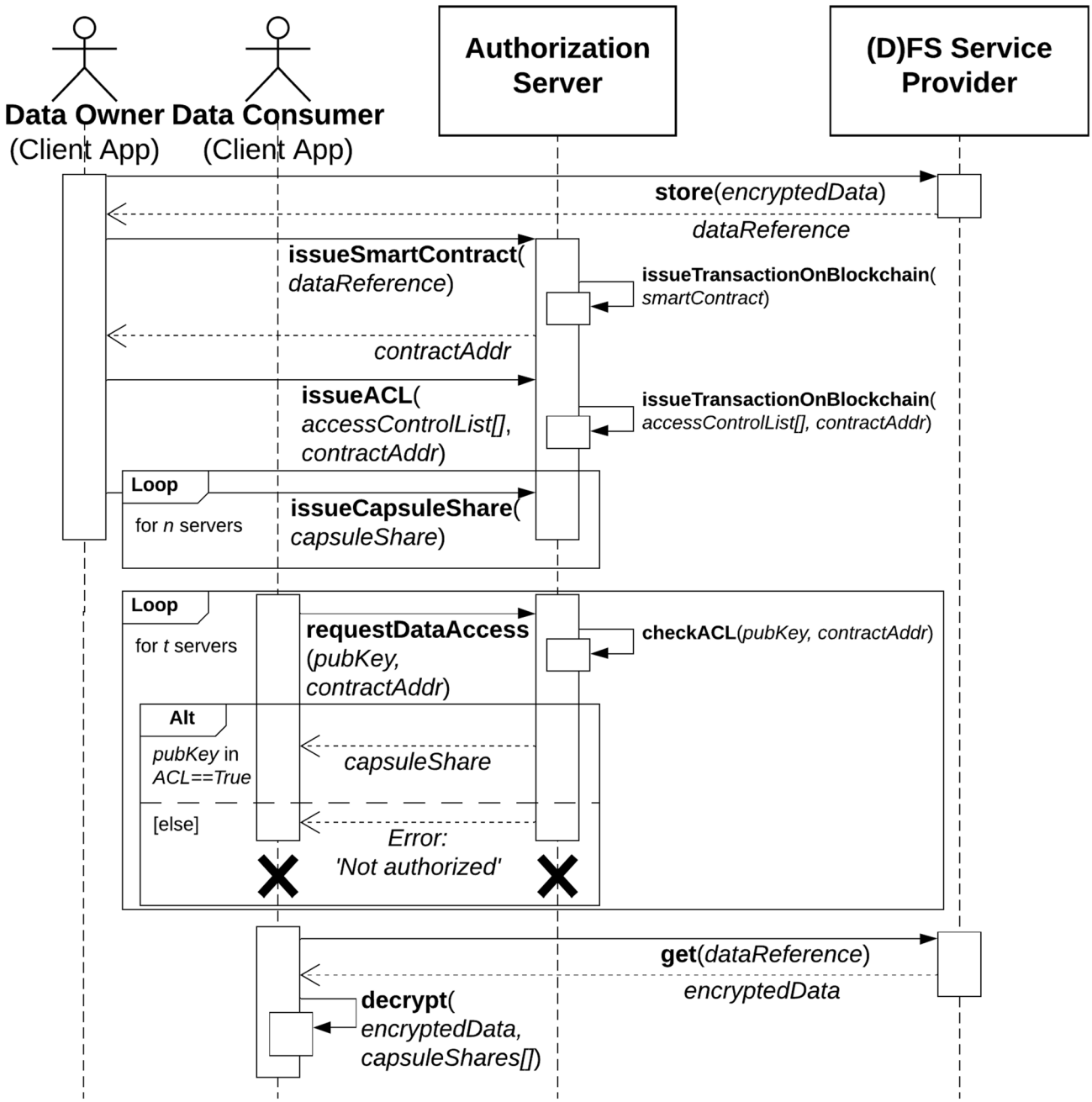

Fig. 4 Sequence Diagram describing the process of personal data storage and access by a consumer

mechanism shown here, the user benefits from a cryptographic proof of security [5, 33, 34, 52].

Algorithm 1: Data Access Request

Global Data:
i Server id
C set of capsule stored by the server
cha challenge message sent to consumer
res response message for the challenge
Input:
pk_{DC} *Data consumer*'s public key
addr smart contract address
hp_{epd_l} datum hash pointer
sign signature of a challenge response
Result:
$c^i_{k_{pd_l}}$ capsule share for the hp_{epd_l} datum

```
// validate signature to authenticate consumer
obtainedPubKey ← verify(cha, res, sign)
if obtainedPubKey == pk_DC then
    // identity confirmed
    acl ← getACL(addr)
    eligiblesSet ← getValueFromKey(acl,hp_epd_l)
    if contains(eligiblesSet, pk_DC) then
        // eligible consumer
        c^i_k_pd_l ← getCapsule(C,hp_epd_l)
        return c^i_k_pd_l
    end
end
return "Error: Not Authorized"
```

Distribution Mechanism Consider the diagram in Fig. 4. We have a *data owner* with a keypair (pk_{DO}, sk_{DO}) and a *data consumer* with a keypair (pk_{DC},sk_{DC}).

The first operation consists of storing the encrypted data epd_l in the PDS and updating the (or issuing a new) related smart contract with the reference to the data, i.e. the hash pointer hp_{epd_l}. A *data consumer* may be allowed to access the stored data simply because the *owner* updates the (or issues a new) ACL in the smart contract with pk_{DC} in it.

While this can be considered a setup, the first real capsule distribution phase occurs when the *owner* shares the capsule $c_{k_{pd_l}}$ associated to epd_l with *n authorization servers AS*, i.e. creates n shares of the capsule such that $c_{k_{pd_l}} = \sum_i^n c^i_{k_{pd_l}}$, and sends one share to each *server*. In here the sum operation represents the share aggregated function associated to the SS or TPRE methods and is discussed in the next paragraph.

The second distribution operation consists of the request by the *consumer* to the *authorization servers* for the release of the capsule, through a challenge-response message signed with pk_{DC}. The data access request is composed of

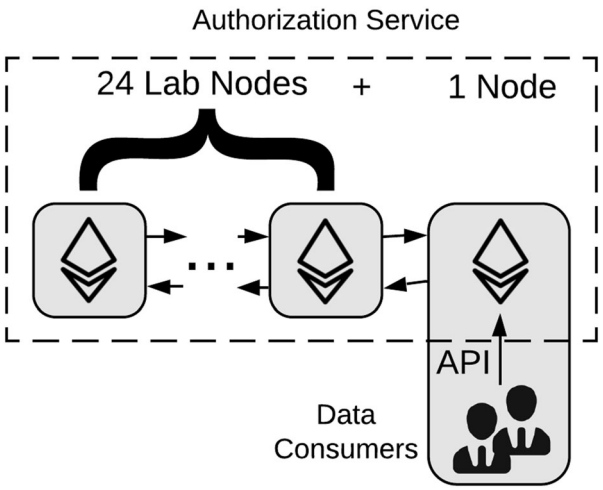

Fig. 5 Setup of the authorization system test network

these elements $dar = \{pk_{DC}, addr, hp_{epd_l}, sign\}$, where *addr* is the address of the smart contract containing the ACL and *sign* is the signature of a challenge-response message provided by each *server*. The pseudocode for this operation is shown in Algorithm 1. Upon this request, only t *servers* are needed to check the ACL for the presence of pk_{DC} and then to release their capsule part to the *consumer*. Finally, the *consumer* can "open" the capsule and obtain the k_{pd_l} content key needed to decrypt the desired data.

Capsule Share We refer to two cryptographic schemes for the *data owner*'s capsules share:

- **Secret sharing (SS)**
  - Using this scheme the sk_{DO} is "divided" into n shares and $t<n$ shares are sufficient to reconstruct sk_{DO}, decrypt the capsule and obtain k_{pd_l}.
  - Thus a capsule share $c^i_{k_{pd_l}}$ consists of the original (encrypted) capsule $c_{k_{pd_l}}$ plus the sk_{DO} share. item The secret sk_{DO} can be represented as an element a_0 of a finite field, then $t-1$ elements are chosen randomly from this field, $a_1, ..., a_{t_1}$. Using these elements this polynomial curve can be constructed $f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{t_1} x^{t_1}$. Every AS_i is given a point found in the curve $(x_i, f(x_i))$, with $1 \le i \le n$.
  - Therefore, here, an untrusted *authorization server* alone cannot open the capsule because it needs other $t-1$ shares.
  - Indeed, in order to obtain a_0, and thus sk_{DO}, a subset of cardinality t of the n points $(x_i, f(x_i))$ are needed to perform the following interpolation: $a_0 = f(0) = \sum_{j=0}^{t-1} f(x_j) \prod_{m=0, m\neq j}^{t-1} \frac{x_m}{x_m - x_j}$.

- **Threshold proxy re-encryption (TPRE)**
  - In this scheme each *authorization server* acts as a proxy and can re-encrypt the capsule it receives through a re-encryption key $pk_{DO \to DC}$ generated by the *owner*.
  - Thus, unlike the previous case, the sk_{DO} is not the one divided, but the re-encryption key is. Thus a capsule share $c^i_{k_{pd_l}}$ consists of the original (encrypted) capsule $c_{k_{pd_l}}$ plus the $pk_{DO \to DC}$ share.
  - The the $pk_{DO \to DC}$ share is obtained through the same process shown for the secret sharing, i.e. as a point $(x_i, f(x_i))$ of the constructed polynomial curve. We can refer to x_i as $kFrag_i$ (we are simplifying the operations description, but further details can be found in [75]).
  - This $kFrag_i$ is not the share that is then provided to the *consumer*, but is is used by the *server* to obtain another piece of data, i.e. the $cFrag_i = ReEncapsulate(c_{k_{pd_l}}, kFrag_i)$ [75].
  - The *consumer* will receive the re-encrypted share, i.e. the $cFrag_i$, which can be used with other $t-1$ shares to reconstruct client-side a capsule encrypted with its public key pk_{DC}. In this case the process is similar to the interpolation shown previously for the secret sharing.
  - Thus the newly obtained capsule can be "opened", i.e. decrypted, using the *consumer*'s private key sk_{DC}.
  - TPRE solves the possible PRE collusion between proxy (i.e. *server*) and receiver (i.e. *consumer*) by assuming that at most $t-1$ proxies can be colluded and not follow the policies.

The system architecture we present supports both techniques, and each can be chosen based on different benefits and drawbacks. While SS relieves the *owner* of any interaction after the capsule has been shared the first time, there is still the possibility that t nodes are malicious and the *owner* cannot intervene to prevent keys from being disclosed. TPRE allows greater control over possible receivers, however it has the disadvantage of requiring that a new re-encryption key $pk_{DO \to DC}$ is generated for each new *consumer*.

4.4 Audit DLT

The above specification of architectural components, as already mentioned, is strongly influenced by two main issues. First, the tension between DLTs and the GDPR [66]. This led to the use of a sidechain with a set of designated *authorization servers*. Second, the scalability issues of current public permissionless DLTs [8, 64], that led to the adoption of a more scalable consensus mechanism in the authorization system. This means that the use of a permissioned DLT instead of a public permissionless one was due to both legal and practical reasons. However, we opted for a multi-DLT architecture, including also a public permissionless DLT, to reach a strong trust on data immutability. Indeed, permissioned DLTs allow a reasonable degree of trust, but the ledger can be altered when the majority of nodes are malicious [21].

A multi-DLT architecture allows achieving decentralization as well as high throughput and low latency [47]. The base layer consists of a public permissionless DLT used to perform batch transaction validation for the second layer, which includes the sidechain. The basic idea is that the states of each authorization system, i.e. the sidechain transactions blocks, are made immutable by registering them in a public permissionless DLT in the form of a hash digest, which is a common mechanism in sidechain technologies [21].

Algorithm 2: Consensus algorithm for publishing to the audit DLT

Global Data: (constants)
n Authorization servers number
r round number
id Server id
Global Data: (variables)
$step$ round step
id_{latest} latest id of the round robin
$blocks$ list of blocks' hash digest
Input:
$block_{hash}$ hash digest of the latest block
Result:
$digest$ published in the audit DLT

```
1  blocks ← append(blocks, block_hash)
   // append latest block hash in the blocks hash
   list for this round
2  if step == r then
       // round finishes
3      if id == id_latest then
4          digest ← merkleTree(blocks)
               // returns the root of a Merkle tree
               having blocks hash digests as leaves
5          deployToAuditDLT(digest)
6      end
7      id_latest ← (id_latest + 1) mod n
8      step ← 1
9      blocks ← [ ]
10 end
11 step ← step + 1
```

Furthermore, this audit DLT, i.e. the base layer public permissionless DLT, is leveraged to store the information regarding the appointed *authorization servers*, so that all of them can be identified and known by *data owners* and *consumers*. During the development of the audit DLT, we opted for a smart-contract enabled blockchain (i.e. Ethereum), since it eases the information management (e.g. when updating the list of *servers*). On this audit DLT only non-personal data are stored, i.e.:

- a growing list of digests (i.e., roots of Merkle trees) that represent the advancement of the authorization system state, $authStates = \{authState_\tau \mid 0 \leq \tau \leq T\}$;
- a (modifiable) list of addresses that represents the entities that provide the authorization service, i.e. *authorization servers*;
- optionally, a stake for each *server* as an incentive for the correct behavior.

Several approaches might be implemented for updating the authorization system state in the audit DLT. According to our solution, in order to trace the evolution of the system in the audit DLT, we define as "round" the generation of r blocks in the sidechain. At the end of each round, we store the hash of the last block in the audit DLT. The pseudocode for this operation is shown in Algorithm 2. The r value must be chosen appropriately:

- it must be $r \neq 1$, because the sidechain grows faster than the main one, due to the different consensus algorithm [64], hence with $r = 1$ the sidechain would have to keep pace with the audit DLT;
- it must assume values $r \geq 2$, in order to ease the reaching of a secure level of data entropy to keep hashed data with a low risk of de-anonymization and thus to consider it non-personal [68];
- it cannot assume a large value, because the delay between one publication in the audit DLT and the next one may make it possible to alter the sidechain when the majority of *servers* are malicious.

In such form, the status information of the authorization system consists of non-personal data, therefore not affected by the GDPR. Finally, the transaction to be issued in the audit DLT is signed with a multi-party signature of all the *authorization servers*.

4.5 GDPR Compliance

4.5.1 On-chain hash pointers

As stated earlier, personal data is referenced on-chain and but stored off-chain in a (D)FS. From the point of view of the GDPR, the result of a hash function applied to personal data (without conscious security measures) is considered by the Art. 29 Working Party (now European Data Protection Board) as pseudonymized data because of the likelihood of deriving the input value [69], thus still in the scope of the GDPR (Recital 26). A GDPR-compliant solution consists of the use of Key Reuse Encryption (Sect. 4.1) and Single-Use Salt [68], minimizing the risks of de-anonymization [5, 65, 67, 68].

4.5.2 Personal data erasure

Personal data deletion is a right to guarantee to the data subject (Sect. 3.1). In a centralized, cloud-like, architecture, this is a simple task. Dealing with DFS is more complicated yet. For instance, in IPFS there is no way to force and verify that data has been removed from the entire network. However, the GDPR itself provides us with a helping hand for compliance here[2]. The approach for the anonymous delegated deletion protocol tackles this definition and performs the reasonable steps to inform controllers for data deletion.

4.5.3 (Semi-)private ledger and GDPR subjects rights

Finally, we have to deal with the right to be forgotten, and, above all, with the principle of data minimisation [24, 71]. Since *authorization servers* act as data controllers, we need a "forgetting" sidechain [66]. Our solution includes the use of pruning [24, 65], which consists of deleting old transactions and blocks (on demand or after a predefined period of time) and keeping the old block headers containing the hashed version of the removed data, in order to ensure the security of the sidechain. Although this technique has been judged to be weakly applicable in permissionless DLTs, pruning may provide a suitable solution for permissioned DLTs, where the operating environment is more easily controlled and regulated [76].

4.5.4 Authorization servers compliance

From the GDPR point of view, *authorization servers*, as stated earlier, act as joint controllers for the transactional data that they verify, store, and put on/off chain. However, *authorization servers* process a *data owner* request for distributing capsule shares and thus they are likely to be processors as they are acting on behalf of the *data owner* as the controller, in that instance. *Servers* can be controllers for some activities and processors for others [62, 66].

[2] Article 17(2) of the GDPR: *"the controller, [...] shall take reasonable steps, including technical measures, to inform controllers which are processing the personal data that the data subject has requested the erasure"*.

4.6 Security and privacy analysis

In this section, the security and privacy properties of the discussed model protocols are discussed but not proven formally. For such task different methods can be considered, such as game-based proof techniques [78], but we leave this task for future works. In here we analyze our model's security and privacy and we conduct a risk analysis for assessing the good practice to pursue GDPR compliance, such as in [79].

In our analysis we will also refer to the CNIL Privacy Impact Assessment methodology, that takes a qualitative approach [77]. The assessment is carried out estimating severity and likelihood using qualitative criteria. Severity represents the magnitude of a risk and it is primarily estimated in terms of the extent of potential impacts on data subjects. Likelihood represents the feasibility of a risk to occur and it is primarily estimated in terms of the level of vulnerabilities of the supporting assets concerned and the level of capabilities of the risk sources to exploit them [77]. The scale used for severity and likelihood is: (1) negligible, (2) limited, (3) significant, (4) maximum. While discussing qualitatively the risks, we will also go into the details of our model. Table 3 shows a summary of the system model and the severity and likelihood assessments related to possible threats. These threats are described in detail in the following:

(1) *Illegitimate access to personal data*—This threat is indicated as a feared event by CNIL [77], and surely represent one of the most important issues that our model tries to tackle.
(2) *Unwanted modification of personal data*—Again, another CNIL's feared event that can cause misuse, errors and malfunctions, especially for the data subject.
(3) *Disappearance of personal data*—The last CNIL's identified feared event, that can result in similar results as in the unwanted modification.
(4) *Collusion with another actor*—This is a threat that deals mostly with the security of the whole model as any actor can collude with others trying to obtain a favourable result, e.g. managing to perform one of the previous three events.
(5) *Tampering the ledger*—This threat involves actors that might want to alter the traced information regarding the system processes, e.g. granting the access to data.
(6) *Repudiation*—This threat involves the fact that repudiation can be exercised by one of the actors in order to avoid accountability for its past actions.
(7) *Denial of service*—Any of the actors providing a service can interrupt its service due to faulty or malicious behaviors. At the same time, any actor acting as client can try to attack the service and interrupt its functioning.
(8) *Lack of involvement in audit*—Finally, some actors might want to hamper the correct execution of an audit, simply by not participating in it.

In the following we will discuss the threats and risks analysis from the point of view of each model's actor.

4.6.1 Data owner

Assuming that the implemented interface for the users is not faulty (and thus it will not be considered in this analysis), *data owners* might still provide *security* threats at the level of the processes of controlling and moving personal data. In this case intentional and unintentional behaviors or implementation faults can (indirectly) provide harm to themselves as data subject and to the actors providing their services lawfully.

(iv) The *data owner* can directly or indirectly (e.g. though a system fault) collude with the *data consumer* in order to render one or more *authorization servers* accountable for having provided illegitimate access to personal data to the consumer. The severity would be significant for the *servers*. In our model, *authorization servers* (i.e. data controllers) are protected from this threat because they maintain the *ACL* and its history thanks to the (semi-)private DLT, and thus can demonstrate the *data owner*'s malicious behavior.
(v) The same holds if the *data owner* modifies the smart contract data with the aim of tampering the ledger, i.e. the history of the modification of the smart contract is kept by all the *servers* through the DLT.
(vi) Finally, the *data owner* can try to repudiate some of the actions performed, however all the *owner* interactions with the systems involve the use of a public key or an address in the form of a digital signature, thus not being repudiated.

4.6.2 Data consumer

(i) The *data consumer* can attempt to illegitimately access to personal data and thus threat the subject privacy. It seems difficult for this threat to materialize by exploiting the properties of the model, however the severity would be limited by the few keys k_{pd_l} that the *consumer* can obtain from honest *authorization servers*.

Table 2 PIMS architecture components description resumed

Component	Description	Technologies To Use	Schemas	GDPR Compliance
User application	Handles *data owners* and *consumers* keys and data	Crypto-system SDK	KEM/DEM technique	*Data owner* is *subject*, *Data consumer* is *processor*
(Decentralized) file storage	Manages the personal data storage (PDS)	DFS (*e.g. IPFS*), FS (*e.g.GoogleDrive*)	On-chain hash pointers, Anonymous delegated deletion	(D)FS service provider is *controller/processor*, Right to be forgotten
Authorization system	Validates requests and provides the means for personal data access	(Semi-)private permissioned ledger	SS + TPRE + Smart Contract ACL	*Authorization servers* are *joint controllers*, Right to be forgotten, Privacy by Design
Audit DLT	Provides the proof of a correct audit for the authorization system	Public permissionless ledger	Multi-DLT sidechain protocol	*Audit DLT node* handles only non personal data

(iv) However, collusion with another actor to access personal data, possibly through hacking or stolen credentials, is limitedly likely. But, since, each $pd_l \in PD$ is encrypted with a unique key, it would be difficult to access large quantities of data, thus limiting the severity of this threat.

(vi) The *data consumer* can repudiate the data access request made, however this has almost no impact for the data subject (negligible severity).

(vii) Finally the *consumer* could try to limit the provision of service from the other actors in the system, i.e. $SP_1, ..., SP_m$, $AS_1, ..., AS_n$, $AN_1, ..., AN_g$. However this would be very unlikely and with limited severity for large enough $m, n$and$g$, as the computation would be highly distributed and the data highly replicated.

4.6.3 (D)FS service providers

(i) Each *(D)FS service provider* SP_j has access to the whole set of encrypted personal data *EPS* and can try to illegitimately decrypt it. Even if the severity for this treat is significant, it is not very likely for a large enough t, where t is the threshold for the capsule share scheme, because SP_j would need to collude with t malicious *authorization servers*.

(ii) Unwanted modification of data is very unlikely for a large enough m. Even in the case not decentralized, i.e. a single *FS service provider*, each modification to $epd_l^{'} \in EPD^{'}$, would show a discrepancy between the hash $hp_{epd_l^{'}}^{'} \in HP^{'}$ obtained from the new modified data and the hash $hp_{epd_l} \in HP^{on-chain}$ saved on-chain, e.g. in the case of a Cloud service could be a violation of the Service Level Agreement [80].

(iii) Disappearance of personal data is likely to happen only in the single *FS service provider*, however it would be another violation of SLA. In the decentralized use case, the high data replication would limit this treat.

(iv-v) As seen in the previous points, the *(D)FS service provider* could collude with t malicious *authorization servers* to access data illegitimately, but this is very unlikely. An SP_j could also collude with some *servers* in order to modify an hash $hp_{epd_l}^{'} \in HP^{on-chain}$ saved on-chain. This is too very unlikely because it would require to break the consensus algorithm of the (semi-)private DLT.

Table 3 Summary of the system model and related security and privacy threats

Actor	Controlled/processed data	Possible enacted threats	Severity	Lkhood
Data owner	$PD = \{pd_l\}$,	(iv) Collusion with another actor	3*	1*
DO	$K = \{k_{pd_l} \mid Enc_{k_{pd_l}}(pd_l)\}$,	(v) Tampering the ledger	3*	1*
	$C = \{c_{k_{pd_l}} \mid c_{k_{pd_l}} = Enc_{pk_{DO}}(k_{pd_l})\}$,	(vi) Repudiation	2*	1*
	all with $1 \leq l \leq o$.			
Data consumer	If authorized processes:	(i) Illegitimate access to p.data	2	1
DC	the $c_{k_{pd_l}}$ and its related pd_l	(iv) Collusion with another actor	2	2
		(vi) Repudiation	1	2
		(vii) Denial of service	2	1
(D)FS service	$EPD = \{epd_l \mid epd_l = Enc_{k_{pd_l}}(pd_l)\}$,	(i) Illegitimate access to p.dat	3	1
providers	$HP = \{hp_{epd_l} \mid hp_{epd_l} = hash(epd_l)\}$,	(ii) Unwanted modification of p.data	3	1
$SP =$	all with $1 \leq l \leq o$.	(iii) Disappearance of p.data	3	1
$SP_1, ..., SP_m$		(iv) Collusion with another actor	3	2
		(v) Tampering the ledger	3	1
		(vii) Denial of service	2	2
Authorization	$HP^{on-chain}$,	(i) Illegitimate access to p.data	3	1
servers	ACL,	(ii) Unwanted modification of p.data	2	1
$AS =$	for each AS_i with $1 \leq i \leq n$:	(iii) Disappearance of p.data	2	1
$AS_1, ..., AS_n$	$C^i = \{c^i_{k_{pd_l}} \mid 1 \leq l \leq o\}$.	(iv) Collusion with another actor	3	3
		(v) Tampering the ledger	3	1
		(vi) Repudiation	1	2
		(vii) Denial of service	2	1
		(viii) Lack of involvement in audit	2	3
Audit DLT	$authStates$	(iv) Collusion with another actor	3	1
nodes		(v) Tampering the ledger	3	1
$AN =$		(vii) Denial of service	2	1
$AN_1, ..., AN_g$		(viii) Lack of involvement in audit	2	1

The rightmost columns show some qualitative measurements in terms of severity and likelihood of the risk related to each threat. The scale for both severity and lkhood (i.e. likelihood) is: (1) negligible, (2) limited, (3) significant, (4) maximum, as in the CNIL methodology [77]. Values refer only to the risk impact for the data subject (values with the * refer also to the impact for other entities)

(vii) A denial of service for *data owners and consumers* could be limitedly likely in the case of a single *FS service provider* and unlikely in the decentralized context of a (D)FS.

4.6.4 Authorization servers

(i) Considering the whole set of encrypted personal data EPD, a single *authorization server* AS_i could try to decrypt this set to illegitimately access personal data. Also in this case, the severity for this treat is significant, but the event is unlikely because other $t-1$ malicious *servers* are needed.

(ii-iii) Considering the set of capsule shares C^i and the ACL as personal data, a single AS_i could modify or delete them without the *data owner* permission. In this case, the severity is limited because AS_i would need $n-t$ other malicious *servers* to modify/delete capsule shares so that they can be made useless, $t-1$ malicious *servers* to modify the original capsule and enough *servers* to disrupt the consensus mechanism to modify/delete the ACL.

(iv) The collusion of AS_i with other *servers* is significantly likely to happen if all these have not the right incentive to deviate from this behavior. This event would be of significant severity because it would allow the colluded *servers* to illegitimately access personal data and let other *consumers* do it too. This event depends as well on the t value. Thus a large enough t and choosing the right incentives are the key factors that enable this treat to be avoided.

(v) Tampering the ledger could lead to the modification of the *ACL* and thus to let the illegitimate access to personal data. This threat is unlikely as an AS_i would need enough *servers* to disrupt the consensus mechanism.

(vi) A AS_i can repudiate the action of sending out capsule shares on request, but capsules are digitally signed. However this has almost no impact for the data subject (negligible severity).

(vii) A denial of service for *data owners and consumers* is unlikely in the decentralized context of a the (semi-)private DLT.

(vii) Lack of involvement in an audit is significantly likely to happen for an *authorization server*, as it would mean simply to not show the ledger of the (semi-)private DLT to the auditors. This could be the only option to hamper the audidt process, as an alteration of the ledger would be detected thanks to the audit DLT *authState*. In this case the *server*, being a controller, would be liable to not cooperate, on request, with the supervisory authority (GDPR, Article 31).

4.6.5 Audit DLT nodes

(iv-v) Very unlikely the majority of *audit DLT nodes* $AN_1, ..., AN_g$ could collude and modify the *authStates*, by themselves, or on demand by colluding with an *authorization server*. This is very unlikely for large enough g, as in Bitcoin and Ethereum public blockchains.

(vii) An *audit DLT node* could deny the service of storing an *authState* to a *server*, but again with a large enough g, it is very unlikely for the *server* to not find other *nodes* providing the service.

(viii) Lack of involvement in an audit is very unlikely to happen as the audit DLT nodes might not even know about the actual meaning of the *authStates* information. being nodes of a public permissionless DLT they would simply share the ledger as it is public to anyone.

5 Performance evaluation

This section contains a description of the tests carried out to evaluate the proposed system. In this performance evaluation, we are mainly interested in the user experience provided by a PIMS, such as that we implemented. Thus, the main critical points are related to scalability, responsiveness and reliability of both DLT and DFS systems. In particular, we focus on i) the latencies due to data upload to the DFS, i.e. the time that a *data owner* would measure when publishing his own data, and ii) the time required for the access control operations and to retrieve the keys needed to decrypt data, i.e. the time that a *data consumer* must wait to read and "consume" data. The complete dataset and the reference software are stored in [81, 82], following the FAIR data principles for access and reuse of models [83].

5.1 Scenario

We conducted our experimental tests on the basis of a scenario where personal data is shared by users. In particular, this trace-driven experimental evaluation was based on a hypothetical Intelligent Transportation System application [8]. Due to our requirements for scalability and responsiveness, we focused on some typical crowdsourced application data, e.g. the location of a subject. Thus, we used a dataset containing Rio de Janeiro (Brasil) buses' real mobility traces [84] to generate user (bus passengers) traces. Users in this case are *data owners* that act both as subjects and controllers. In our simulation, a device of a user on-board a bus (one user per bus) runs a software that periodically retrieves data from sensors, e.g. temperature, air pollution, geolocation data. These data were used to generate real requests transmitted to the various systems of the architecture, with intervals provided by the real mobility traces. We consider:

- Small sized data ($\sim$100 Bytes), e.g. latitude and longitude;
- Large sized data ($\sim$1 Megabyte), e.g. photos.

In the second part of our tests also *data consumers*' devices are simulated, while a smart contract containing the *data owner*'s ACL was already issued in the blockchain. We then simulated the insertion of (10 to 100) *data consumers* to the ACL and the subsequent request to the authorization system for accessing some data (henceforth referred to as a message).

5.2 System implementation and devices setup

We implemented the PDS and the authorization system and exploited some public available decentralized networks/services.

5.2.1 User client application and personal data storage implementation

In the assessment of our PDS, we used a single *(D)FS service provider* node (Sect. 4.2.2), while varying the number of *data owners* (henceforth referred to as users). Since we are interested in performance concerning messages uploading from the user's point of view, and since users' devices generally send requests to only one node, we focused on the DFS node response latency only, instead of testing the response of the whole network. We simulated users' devices on a dedicated device (Google Cloud VM n1-standard-1, 1 Intel Skylake vCPU, 3,75 GB RAM), henceforth referred to as simulation device. Then, since users can decide their own DFS solution to request the storage of data, i.e. messages, we compared three different DFS solutions: IPFS Proprietary, IPFS Service and Sia Skynet (the difference between IPFS and SIA is explained in Sect. 2.2):

- *IPFS Proprietary* A dedicated IPFS node, specifically in charge of handling messages generated by our ITS application, that is connected to the main IPFS network. Thus, in this configuration, simulated users' devices were the only ones sending requests for storing messages to this node.
- *IPFS Service* A generic IPFS service provider, i.e. Infura [85]. This is a general purpose service that provides free access to the main IPFS public network. During the tests the utilized IPFS node was receiving other concurrent requests, coming from all over the world.
- *Sia Skynet* A special Sia public node offering free access to the Sia network, but with limited storage space available. In particular, in this case, the Sia node has already formed contracts with every available host, rewarding their file replication.

5.2.2 Authorization system implementation

The experimental tests for our authorization system were performed using a network of interconnected nodes, i.e. *authorization servers*, forming the (semi-)private Ethereum sidechain described in Sect. 4.3, and a capsule distribution service. We tested how the network responds to the simulated *data consumers*' requests, based on different system configurations.

The following evaluation focuses on the distribution of the encryption keys, in particular on the distribution of the capsule generated by the KEM (Sect. 4.1) to the network. We have measured the latency needed to perform each operation from the point of view of the *data owner* and *data consumer* client software.

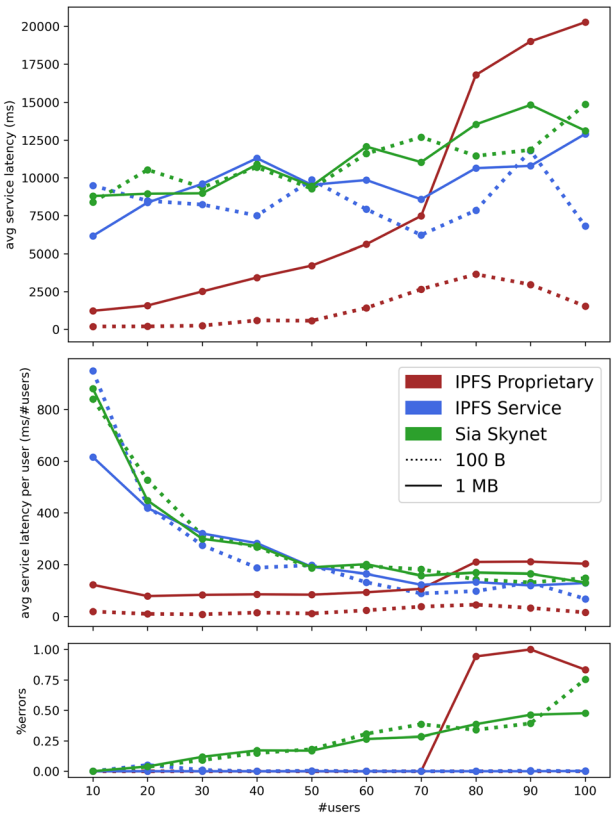

Fig. 6 Average latency between the sending of a message to *(D)FS service providers* node and the response on top. Percentage of response errors at the bottom

We configured a network of 25 nodes to form the authorization system. 24 of these nodes are hosted in the same laboratory of the University of Bologna (each one is equipped with Intel Core i5-2400 CPU and 8 GB RAM). The simulation device is external to the laboratory and acts both as the 25th node and as a *data consumer*'s device simulator. Each of the 25 nodes provides the key distribution service by running (not simultaneously) two implementations of two different software programs: i) SS scheme: the Secret Store [17] provided by the OpenEthereum application, ii) TPRE scheme: a server implementation of the Umbral library in NuCypher [52].

5.3 Results

In this subsection, we present the obtained results from the testbed we described.

5.3.1 User client application and personal data storage

To assess the time needed to distribute data and encrypt/decrypt them, we specifically focused on two aspects:

- *DFS service providers response* we conducted our performance evaluation of the three different types of DFS nodes with the purpose of conducting a stress test. The interval between one request and the next is varied by following the mobility trace dataset and varying the number of users (from 10 to 100). Each simulated user sends exactly 15 messages in 15 minutes and consecutive runs of the simulation are separated by an interval of 10 or 20 minutes.
- *Message size variation* tests were conducted in order of dimension (small messages first, then larger ones) for DFS requests. Then we carried out tests in order to assess the encryption and decryption performance with different types of messages, ranging from small text data (10 B) to larger data (10 MB).

DFS service providers response

Figure 6 reports, on average, the latency between the sending of a message to the considered DFS node and its response, together with the relative percentage of errors (i.e. HTTP status code 500 or 504). In the case of errors, the messages are not considered in the average. Results are represented as a dotted line for small messages (100 B) latencies, while solid lines show the latencies and errors when larger messages, i.e., a 1 MB-sized photo, are sent to the DFS nodes. Latencies are reported relative to the number of users sending requests, in order to assess the deviation from the usual latency in response to requests from an increasing number of users.

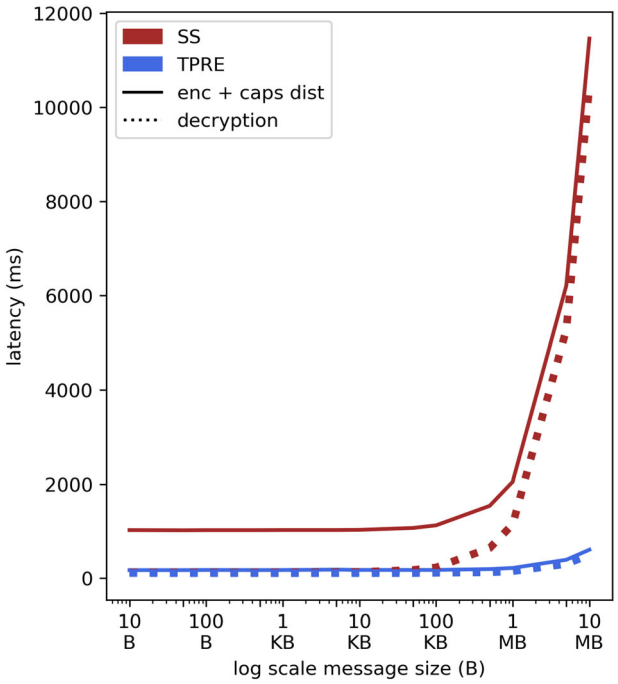

Fig. 7 Latencies when encrypting and decrypting messages varying message sizes

Overall, IPFS performs better than Sia in terms of latency and errors. In particular, IPFS Proprietary presents an average latency per user of about 40 ms in the case of small messages, while, conversely, both IPFS and Sia services averaged a latency of approximately one second. With regard to the error percentage, however, we note a high rate with Sia and a low level in IPFS configurations.

Focusing on IPFS Service and Sia, we can see that the behaviour between handling small and large files does not vary much. We can associate this fact with the large amount of computational resources that the two services have, as opposed to the Proprietary node which suffers greatly from the increase in file size. More in detail, the IPFS Service always has better or similar latencies to those of the Sia Service, and furthermore the error rate is very low. The Sia service, on the other hand, needs to reject more and more requests, i.e. generate errors, to maintain a stable latency as the number of requests increases. In fact, the number of errors seems to increase linearly with the number of users sending requests.

An interesting result is that, according to the use of *(D)FS service providers* (i.e. IPFS Service and Sia), the latency experienced for each user decreases when the number of users increases. This is probably due to the fact that the data uploads are handled periodically by these providers. Hence, the more the requests in the buffer the more the data they process per interval. However, in Sia the amount of errors increases with the number of active users, meaning that the provider is not completely able to properly process all these requests.

Conversely, the IPFS Proprietary shows a linear performance per user, since according to our implementation, data are processed as soon as they are received. For large messages, this is true only up to a certain threshold of users, after which we experienced a sudden error increase, with a corresponding latency increase. This is due to the fact that after a certain limit, the node is not able to properly handle all the requests, thus causing a cascading effect on the overall performance. This turning point is at 80 users.

Message Size Variation

On the client side, we assessed the impact of different message size values, thus reflecting more the *data owner/consumer* device's capabilities. Figure 7 shows the latency averages with respect to the message sizes for the process of encryption/decryption. The encryption also includes the key shares distribution. The figure suggests that the TPRE scheme implementation can handle the whole process better. From 10 B to 1 MB, TPRE has no issues, while SS results curve exhibits a noticeable inflection point as the message size reaches 500 KB. Then it skyrockets from 1 MB onward. In this case, there is a latency increase of 864%, from 1 MB to 10 MB, for the encryption of the

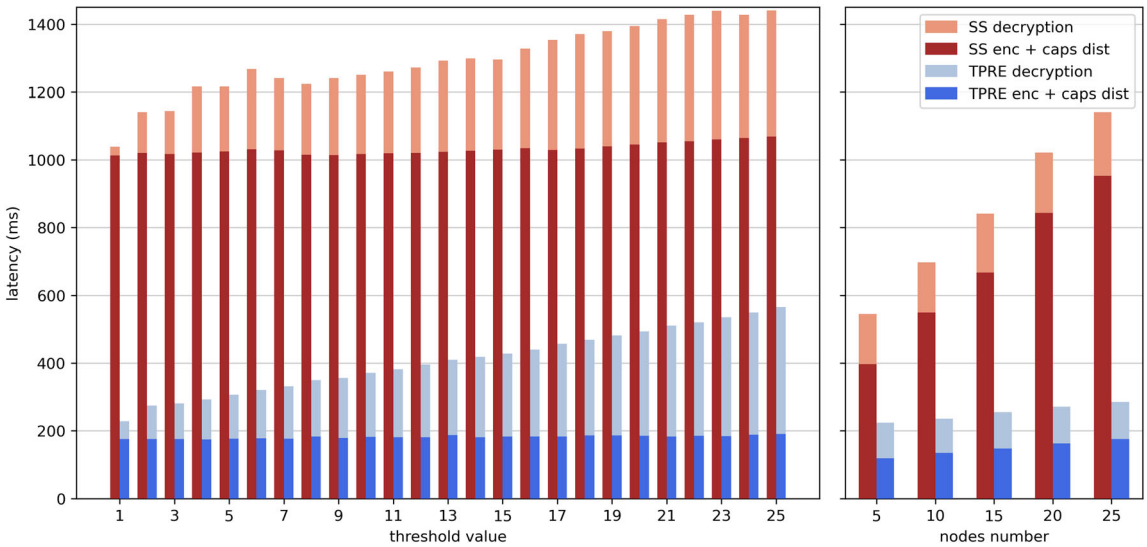

Fig. 8 Encryption and decryption latencies for SS and TPRE

message (without considering key generation) and of 809% for the decryption.

This requires two explanations. First, it is inevitable to have an increase after 1 MB for both schemes, because before this threshold the message size has almost no impact on the operations, i.e. latencies are stable. Second, we have a clear difference between SS and TPRE scheme after the 1 MB, probably due to the SS implementation that during all the tests has presented an overhead in respect to the TPRE implementation; moreover, keys shares distribution might also have a great impact at the expense of the SS schema here, since this is common also in the other tests.

5.3.2 Authorization system

The following tests were performed to evaluate the authorization system. Here we refer to n for the total number of network nodes and t for the threshold (i.e. expressed as the number of nodes) needed to retrieve a complete key. We tested:

- *Threshold Variation* this test case involves the variation of t, after having fixed $n = 25$ as the number of nodes and 30 B as the message size. Our intent here is to show the latencies measured when the threshold value increases.
- *Nodes Number Variation* we set up this test case in order to assess the different configurations of the authorization system by varying the number of *authorization server* nodes, i.e. n. The threshold value was set to $t = 2$, while 30 KB was the size of a message.
- *Blockchain Overhead Analysis* we configured the authorization system blockchain with a specific PoA rule and analyzed the transactions throughput.

For all tests, every single data point was obtained by repeating the operation 10 times (with a time interval of 300ms between each request) and then averaging the resulting latency values.

Threshold Variation

As described before, the threshold is a key parameter for both TPRE and SS. In essence, it defines how many nodes must agree on the status of the smart contract. We fixed the number of nodes on the network to 25, and then tested values of t from 1 to 25. As the left-most bar chart in Fig. 8 shows, the encryption (+ key shares distribution) time remains mostly constant ($\sim$7 ms for TPRE and $\sim$52 ms for SS). On the other hand, the decryption time increases linearly with t and both TPRE and SS present a very similar trend. This is because with the increasing of t more nodes are involved in the decryption.

While SS and TPRE schemes take approximately the same amount of time to perform the decryption operation, the same cannot be said for the whole encryption operation. The largest difference in latency can be observed in the generation and distribution of capsule shares, which is embedded in the encryption operation. SS performs a distributed capsule generation scheme based on Elliptic Curve Discrete Logarithm [86] to set up the keys needed in KEM and DEM, while TPRE capsule and shares generation is completely performed locally and then distributed to nodes. This difference is heavily reflected in results ($\sim$792 ms time difference with $n = 25$).

Nodes Number Variation

Network scalability is an important aspect to take into account in distributed systems. While the threshold value can be set by the *data owner*, choosing between higher security or faster operations, the number of nodes and the resources allocated are usually fixed. In the center chart in

Fig. 8, in fact, it is possible to see that, as expected, generally the time costs of operations increase with the number of nodes n. However, we must note the fact that the values for the SS results grow faster than the values for TPRE results. This makes the TPRE method more scalable.

In respect to the previous results (threshold value), the encryption values here are higher for SS and increase linearly with n in SS and TPRE. The decryption times remain almost constant because t is never altered. Even in these tests, we can see the effect of SS keys generation and distribution, since a larger number of nodes means an increase in the time needed to reach all of them.

5.3.3 Blockchain overhead analysis

A further consideration to take into account for previous results, is the fact that the writing and verification of data access permissions in a smart contract in the authorization system could increase the latency throughout the whole encryption/decryption phase.

In our tests, the verification phase has a poor impact on the decryption phase, requiring (on average) only 5 ms per address checked. This happens since all the blockchain nodes maintain approximately the same version of the ledger and do not have to forward the verification request to other nodes.

On the other hand, writing data access permissions on the smart contract ACL is limited by the transactions throughput of the specific blockchain taken into consideration. In our tests, we deployed a private permissioned Ethereum blockchain with 25 nodes using Proof-of-Authority (PoA). In our tests the PoA was limited to the production of a block every 5 seconds, with a gas limit of 6*million* gas units. Since, in our smart contract implementation, the cost of adding a single address to the ACL is 50562 Ethereum gas units, we can store a maximum 118 transactions in a block, leading to 23.6 transactions per second as (theoretical) throughput. A solution to further decrease the gas per address is by bundling addresses that need to be added to the (same) ACL, in a single transaction. For instance, 1 transaction with 10 addresses would require 265394 gas units, which is almost half the cost of 10 transactions with 1 address each, and having only these kinds of transactions leads to a 44 addresses per second (theoretical) throughput. Another tweak to the blockchain configuration might be to decrease the time between the creation of a block and another from 5 to 2 seconds. This would produce a (theoretical) throughput of 110 addresses per second, but the network topology must allow the nodes to be able to create, sign and disseminate blocks in under 2 seconds.

5.4 Discussion

The best case scenario for the encryption and storing of a single message of 1 MB requires, on average, approximately 1 second, by using in sequence the TPRE encryption and IPFS Proprietary storing. However, it is worth noticing that some of these operations can be executed in parallel, e.g. keys distribution and DFS storing, thus further decreasing latencies.

In our simulations, the Proprietary IPFS node produced the best results up to the threshold of 80 users when dealing with large messages. This suggests the use of a dedicated IPFS node, even with limited computational capacity, to handle the reception of 60-70 requests per minute. Thus, an adequate deployment of DFS nodes (e.g. maintained by the same nodes as the authorization system) based on the number of users requesting access to the service can properly support a PIMS.

Regarding the encryption phase, results clearly show how the TPRE scheme, implemented as a system based on NuCypher, performs better over the SS scheme, implemented as OpenEthereum Secret Store.

Finally, an essential comment is needed, related to a major difference between the considered TPRE and SS schemes. We have performed the performance evaluation assuming that the *data owner* device, that releases the re-encrypted keys, is always operational and online. However, in a real-world scenario, this device might introduce some delays when using the TPRE scheme and not when using SS, since in the SS scheme the owner device is not required to complete the operations. This aspect can influence the choice between the two schemes both from an operational point of view and from a performance perspective.

6 Conclusions

In this paper, we presented the multi-layered architecture for the management of personal information based on the use of distributed ledger technologies (DLTs), providing a trustless environment for interactions between the *data owners* and *consumers*. The rationale was to provide individuals with a tool to effectively exercise (at least part of) their rights, as in the General Data Protection Regulation (GDPR) and to enable data sharing and altruism as intended by the newly proposed European Data Governance Act. We analyzed the tensions between the GDPR and DLTs and, in the light of Self-Sovereign Identity, we introduced the following architectural components:

- a personal data storage (PDS) based on a (D)FS, i.e. a centralized or decentralized file storage, where data is protected through a hybrid encryption scheme,

including a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM);
- an authorization system based on a sidechain, i.e. a (semi-)private permissioned ledger, where a set of smart contracts allows *data owners* to define access (through an ACL) to their personal data stored in their PDS. The access to the data is controlled through two distributed mechanisms, i.e. secret sharing (SS) and threshold proxy re-encryption (TPRE);
- an audit DLT, that consists of a permissionless DLT that provides proof of a correct audit for the authorization system.

Furthermore, we provide a prototype implementation developing the sidechain as an Ethereum blockchain and leveraging IPFS and Sia as DFS. At first, we discussed their qualitative differences; then, we compared them experimentally in terms of execution time, taking an Intelligent Transportation Systems (ITS) scenario as a use case. Results from our performance evaluation show that: (i) up to a certain overload, a proprietary service, where a dedicated node is in charge of running an IPFS node, appears to provide stronger assurances for responsiveness and reliability; (ii) TPRE is faster when increasing the size of data to encrypt/decrypt and it is also more scalable, as better behaves as the number of nodes and the threshold value increases while executing the protocol; (iii) however, TPRE has the drawback of requiring the generation of re-encryption keys for each new data consumer, while SS does not. (iv) We also analyzed the transaction throughput of a private Ethereum blockchain using Proof-of-Authority, confronting smart contracts operations in terms of gas.

What we have discussed suggests that it is possible to build reliable and scalable decentralized systems for managing personal information through an appropriate composition of sub-systems within the infrastructure, and that at the same time data sovereignty can be provided. In future work, we will pursue more complex policy enforcement such as ABE and investigate how to overcome the limitations of TPRE and SS.

Author Contributions All authors contributed to the study conception and design. Implementation and data collection were performed by Mirko Zichichi. All authors participated in the analysis of the results and to the preparation of the manuscript.

Funding Open access funding provided by Università degli Studi di Urbino Carlo Bo within the CRUI-CARE Agreement. This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie International Training Network European Joint Doctorate grant agreement No 814177 Law, Science and Technology Joint Doctorate—Rights of Internet of Everything.

Availability of data and code The complete dataset associated with the manuscript and the reference software are stored in [81, 82], following the FAIR data principles for access and reuse of models.

Declarations

Conflict of interest None.

Informed consent Not applicable.

References

1. Pariser, E.: The filter bubble: what the internet is hiding from you. Penguin UK, London (2011)
2. Council of European Union: Regulation (eu) 2016/679 - directive 95/46
3. California State Legislature: California Consumer Privacy Act (2020). https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375. Accessed 3 July 2022
4. European Data Protection Supervisors: opinion on personal information management systems (2016). https://edps.europa.eu/sites/edp/files/publication/16-10-20_pims_opinion_en.pdf. Accessed 3 July 2022
5. European Union Agency for Cybersecurity: Data pseudonymisation: advanced techniques & use cases. Technical report, European Union Agency for Cybersecurity (2021). https://www.enisa.europa.eu/publications/data-pseudonymisation-advanced-techniques-and-use-cases. Accessed 3 July 2022
6. European Commission: A European strategy for data (2020)
7. Zichichi, M., Ferretti, S., D'Angelo, G.: 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. Zichichi, M., Ferretti, S., D'Angelo, G.: A framework based on distributed ledger technologies for data management and services in intelligent transportation systems. IEEE Access **8**, 100384–100402 (2020)
9. Furini, M., Mirri, S., Montangero, M., Prandi, C.: Privacy perception when using smartphone applications. Mobile Netw. Appl. **25**, 1055–1061 (2020). https://doi.org/10.1007/s11036-020-01529-z
10. European Commission: European data governance (Data Governance Act) (2020)
11. European Parliament: European Parliament resolution of 3 October 2018 on distributed ledger technologies and blockchains: building trust with disintermediation (2017). https://www.europarl.europa.eu/doceo/document/TA-8-2018-0373_EN.html. Accessed 3 July 2022

12. Giannopoulou, A.: Data protection compliance challenges for self-sovereign identity. In: Prieto, J., Pinto, A., Das, A.K., Ferretti, S. (eds.) Blockchain and applications, pp. 91–100. Springer, Cham (2020)
13. Zichichi, M., Ferretti, S., D'Angelo, G., Rodríguez-Doncel, V.: Personal data access control through distributed authorization. In: 2020 IEEE 19th International symposium on network computing and applications (NCA), pp. 1–4. IEEE (2020)
14. Zichichi, M., Ferretti, S., D'Angelo, G.: 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)
15. Maesa, D.D.F., Mori, P., Ricci, L.: A blockchain based approach for the definition of auditable access control systems. Comput. Secur. **84**, 93–119 (2019)
16. Politou, E., Alepis, E., Patsakis, C., Casino, F., Alazab, M.: Delegated content erasure in IPFS. Futur. Gener. Comput. Syst. **112**, 956–964 (2020). https://doi.org/10.1016/j.future.2020.06.037
17. OpenEthereum: Secret store (2020). https://openethereum.github.io/Secret-Store. Accessed 3 July 2022
18. Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014)
19. Vorick, D., Champine, L.: Sia: simple decentralized storage. Tech. Rep. Nebulous Inc (2014). https://sia.tech/sia.pdf. Accessed July 2022.
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009). http://www.bitcoin.org/bitcoin.pdf. Accessed 3 July 2022
21. Singh, A., Click, K., Parizi, R.M., Zhang, Q., Dehghantanha, A., Choo, K.-K.R.: Sidechain technologies in blockchain networks: an examination and state-of-the-art review. J. Netw. Comput. Appl. **149**, 102471 (2020). https://doi.org/10.1016/j.jnca.2019.102471
22. Buterin, V., et al.: Ethereum white paper (2013). https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 3 July 2022
23. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., : Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, pp. 1–15 (2018)
24. Politou, E., Casino, F., Alepis, E., Patsakis, C.: Blockchain mutability: challenges and proposed solutions. IEEE Trans. Emerg. Topics Comput. **99**, 1–1 (2019). https://doi.org/10.1109/tetc.2019.2949510
25. Mougayar, W.: The Business Blockchain: promise, practice, and application of the next internet technology. Wiley, New York (2016)
26. Li, L., Liu, J., Cheng, L., Qiu, S., Wang, W., Zhang, X., Zhang, Z.: Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. IEEE Trans. Intell. Transp. Syst. **19**(7), 2204–2220 (2018)
27. Shahid, A.R., Pissinou, N., Njilla, L., Alemany, S., Imteaj, A., Makki, K., Aguilar, E.: Quantifying location privacy in permissioned blockchain-based internet of things (iot). In: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. MobiQuitous '19, pp. 116–125. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3360774.3360800
28. Benet, J., Greco, N.: Filecoin: a decentralized storage network. Protoc, Labs (2018)
29. Naz, M., Al-zahrani, F.A., Khalid, R., Javaid, N., Qamar, A.M., Afzal, M.K., Shafiq, M.: A secure data sharing platform using blockchain and interplanetary file system. Sustainability **11**(24), 7054 (2019)
30. Hawig, D., Zhou, C., Fuhrhop, S., Fialho, A.S., Ramachandran, N.: Designing a distributed ledger technology system for interoperable and general data protection regulation-compliant health data exchange: a use case in blood glucose data. J. Med. Internet Res. **21**(6), 13665 (2019)
31. Kayem, A.V., Akl, S.G., Martin, P.: Adaptive cryptographic access control, vol. 48. Springer, New York (2010)
32. Jemel, M., Serhrouchni, A.: 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)
33. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
34. Blakley, G.R.: Safeguarding cryptographic keys. In: 1979 International Workshop on Managing Requirements Knowledge (MARK), pp. 313–318. IEEE (1979)
35. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inform. Syst. Secur. (TISSEC) **9**(1), 1–30 (2006)
36. Goddard, M.: The eu general data protection regulation (gdpr): European regulation that has a global impact. Int. J. Mark. Res. **59**(6), 703–705 (2017)
37. Rouhani, S., Deters, R.: Blockchain based access control systems: state of the art and challenges. In: IEEE/WIC/ACM International Conference on Web Intelligence. WI '19, pp. 423–428. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3350546.3352561
38. Hassanzadeh-Nazarabadi, Y., Taheri-Boshrooyeh, S., Otoum, S., Ucar, S., Özkasap, Ö.: Dht-based communications survey: architectures and use cases. arXiv preprint arXiv:2109.10787 (2021)
39. Aiello, M., Cambiaso, E., Canonico, R., Maccari, L., Mellia, M., Pescapè, A., Vaccari, I.: Ippo: A privacy-aware architecture for decentralized data-sharing. arXiv preprint arXiv:2001.06420 (2020)
40. Shafagh, H., Burkhalter, L., Duquennoy, S., Hithnawi, A., Ratnasamy, S.: Droplet: decentralized authorization for iot data streams. arXiv preprint arXiv:1806.02057 (2018)
41. Jiang, S., Liu, J., Wang, L., Yoo, S.-M.: Verifiable search meets blockchain: a privacy-preserving framework for outsourced encrypted data. In: ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2019)
42. Courtois, N.T., Mercer, R.: Stealth address and key management techniques in blockchain systems. ICISSP **2017**, 559–566 (2017)
43. Ali, M., Shea, R., Nelson, J., Freedman, M.J.: Blockstack: a new decentralized internet. Whitepaper (2017)
44. Zhang, Y., He, D., Choo, K.-K.R.: Bads: Blockchain-based architecture for data sharing with abs and cp-abe in iot. Wirel. Commun. Mobile Comput. **2018** (2018)
45. Wang, S., Zhang, Y., Zhang, Y.: A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access **6**, 38437–38450 (2018)
46. Xu, H., He, Q., Li, X., Jiang, B., Qin, K.: BDSS-FA: a Blockchain-based data security sharing platform with fine-grained access control. IEEE Access **8**, 87552–87561 (2020). https://doi.org/10.1109/access.2020.2992649
47. Chang, E.Y., Liao, S.-W., Liu, C.-T., Lin, W.-C., Liao, P.-W., Fu, W.-K., Mei, C.-H., Chang, E.J.: Deeplinq: distributed multi-layer ledgers for privacy-preserving data sharing. In: 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), pp. 173–178. IEEE (2018)
48. Zyskind, G., Nathan, O.: Decentralizing privacy: using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops, pp. 180–184. IEEE (2015)

49. Yan, Z., Gan, G., Riad, K.: 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)
50. Truong, N.B., Sun, K., Lee, G.M., Guo, Y.: Gdpr-compliant personal data management: a blockchain-based solution. IEEE Trans. Inf. Forensics Secur. **15**, 1746–1761 (2020). https://doi.org/10.1109/TIFS.2019.2948287
51. Onik, M.M.H., Kim, C.-S., Lee, N.-Y., Yang, J.: Privacy-aware blockchain for personal data sharing and tracking. Open Comput. Sci. **9**(1), 80–91 (2019)
52. Egorov, M., Wilkison, M., Nuñez, D.: Nucypher kms: decentralized key management system. arXiv preprint arXiv:1707.06140 (2017)
53. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: International Workshop on Public Key Cryptography, pp. 53–70. Springer, New York (2011)
54. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans. Parallel Distrib. Syst. **22**(7), 1214–1221 (2010)
55. Meessen, P., Venema, M., Sonnino, A., Bano, S.: D3.8 decentralised models for data and identity management: Blockchain and abc mvps. Decode H2020, Decode consortium, DECODE Project, Tech. Rep. H2020-ICT-2016-1 (2018)
56. Cavoukian, A.: Privacy by design. Take the challenge. Information and privacy commissioner of Ontario, Canada (2009)
57. Davari, M., Bertino, E.: 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)
58. Koscina, M., Manset, D., Negri, C., Perez, O.: 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)
59. Molina, F., Betarte, G., Luna, C.: A blockchain based and gdpr-compliant design of a system for digital education certificates. arXiv preprint arXiv:2010.12980 (2020)
60. Ahmed, J., Yildirim, S., Nowostawski, M., Abomhara, M., Ramachandra, R., Elezaj, O.: Towards blockchain-based GDPR-compliant online social networks: challenges, opportunities and way forward. In: Future of Information and Communication Conference, pp. 113–129. Springer, New York (2020)
61. Kondova, G., Erbguth, J.: Self-sovereign identity on public blockchains and the gdpr. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 342–345 (2020)
62. Foundation, T.S.: Innovation meets compliance: data privacy regulation and distributed ledger technology. Technical report, The Sovrin Foundation (2020)
63. Lundkvist, C., Heck, R., Torstensson, J., Mitton, Z., Sena, M.: Uport: a platform for self-sovereign identity. https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf(2017)
64. Bez, M., Fornari, G., Vardanega, T.: The scalability challenge of ethereum: an initial quantitative analysis. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 167–176. IEEE (2019)
65. Finck, M.: Blockchain and the General data protection regulation: can distributed ledgers be squared with European data protection law?: Study. European Parliament, Brussels (2019)
66. Lyons, T., Courcelas, L., Timsit, K.: Blockchain and the gdpr. In: The European Union Blockchain Observatory and Forum (2018)
67. Finck, M., Pallas, F.: They who must not be identified-distinguishing personal from non-personal data under the GDPR. Int. Data Privacy Law **10**(1), 11–36 (2020). https://doi.org/10.1093/idpl/ipz026
68. Agencia Espanola Proteccion Datos: Introduction to the hash function as a personal data pseudonymisation technique. Technical report, Agencia Espanola Proteccion Datos (2019). https://edps.europa.eu/sites/edp/files/publication/19-10-30_aepd-edps_paper_hash_final_en.pdf. Accessed 3 July 2022
69. Article 29 Working Party: Opinion 05/2014 on Anonymisation Techniques (2014). https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/index_en.htm. Accessed 3 July 2022
70. European Union Agency for Cybersecurity: Guidelines for SMEs on the security of personal data processing. Technical report, European Union Agency for Cybersecurity (2017). https://www.enisa.europa.eu/publications/guidelines-for-smes-on-the-security-of-personal-data-processing. Accessed 3 July 2022
71. CNIL—Commission nationale de l'informatique et des libertés: Solutions for a responsible use of the blockchain in the context of personal data (2018). https://www.cnil.fr/sites/default/files/atoms/files/blockchain_en.pdf. Accessed 3 July 2022
72. Rieger, A., Guggenmos, F., Lockl, J., Fridgen, G., Urbach, N.: Building a Blockchain Application that Complies with the EU General Data Protection Regulation. MIS Q. Exec. **18**(4), 263–279 (2019). https://doi.org/10.17705/2msqe.00020
73. Herranz, J., Hofheinz, D., Kiltz, E.: Kem/dem: Necessary and sufficient conditions for secure hybrid encryption. IACR Cryptology ePrint Archive (2006)
74. Toyoda, K., Machi, K., Ohtake, Y., Zhang, A.N.: Function-level bottleneck analysis of private proof-of-authority ethereum blockchain. IEEE Access **8**, 141611–141621 (2020). https://doi.org/10.1109/ACCESS.2020.3011876
75. Nunez, D.: Umbral: A threshold proxy re-encryption scheme (2018). https://raw.githubusercontent.com/nucypher/umbral-doc/master/umbral-doc.pdf. Accessed 3 July 2022
76. Palm, E.: Implications and impact of blockchain transaction pruning. Tech. Rep. (2017). http://www.diva-portal.org/smash/get/diva2:1130492/FULLTEXT01.pdf
77. French Data Protection Authority (CNIL): Privacy Impact Assessment (PIA)—knowledge bases (2018. Technical report, French Data Protection Authority (CNIL) (2018). https://www.cnil.fr/sites/default/files/atoms/files/cnil-pia-3-en-knowledgebases.pdf. Accessed 3 July 2022
78. Unterweger, A., Taheri-Boshrooyeh, S., Eibl, G., Knirsch, F., Küpçü, A., Engel, D.: Understanding game-based privacy proofs for energy consumption aggregation protocols. IEEE Trans. Smart Grid **10**(5), 5514–5523 (2018)
79. Campanile, L., Cantiello, P., Iacono, M., Marulli, F., Mastroianni, M.: Risk analysis of a gdpr-compliant deletion technique for consortium blockchains based on pseudonymization. In: International Conference on Computational Science and Its Applications, pp. 3–14. Springer, New York (2021)
80. D'Angelo, G., Ferretti, S., Marzolla, M.: A blockchain-based flight data recorder for cloud accountability. In: Proc. of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock) (2018). https://doi.org/10.1145/3211933.3211950
81. Zichichi, M.: miker83z/decentralizedAuthTests: authorization system. Zenodo (2021). https://doi.org/10.5281/zenodo.4572552
82. Zichichi, M.: miker83z/testingIPFS: IPFS and SIA user client application tests. Zenodo (2021). https://doi.org/10.5281/zenodo.4572578
83. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L.B., Bourne, P.E.: The fair guiding principles for scientific data management and stewardship. Sci. Data **3**(1), 1–9 (2016)
84. Dias, D., Costa, L.H.M.K.: CRAWDAD dataset coppe-ufrj/RioBuses (v. 2018-03-19). https://crawdad.org/coppe-ufrj/RioBuses/

20180319 (2018). https://doi.org/10.15783/C7B64B. Accessed 3 July 2022
85. Infura Inc: Infura: Secure and scalable access to ethereum apis and ipfs gateways. (2020). https://infura.io/. Accessed 3 July 2022
86. Tang, C.: Ecdkg: A distributed key generation protocol based on elliptic curve discrete logarithm. sE· CURECOMM, 353–364 (2005)

Mirko Zichichi is a doctoral researcher in the Law, Science and Technology Joint Doctorate - Rights of Internet of Everything, funded by Marie Sk?odowska-Curie Actions. He received a Bachelor degree in Computer Science from the University of Palermo in 2017 and a Master degree in Information Science for Management from the University of Bologna in 2019 (both summa cum laude). He joined the Ontology Engineering Group (OEG) of the Universidad Politécnica de Madrid later in 2019. His doctoral research focuses on the location privacy and inference in online social networks and on the use of Distributed Ledger Technologies and Smart Contracts for the protection and distribution of individuals' personal data.

Stefano Ferretti is an Associate Professor at the Department of Pure and Applied Sciences, University of Urbino "Carlo Bo", since 2020. Earlier, he was Associate Professor at the Department of Computer Science and Engineering of the University of Bologna. He received the Laurea degree (summa cum laude) and the Ph.D. in Computer Science from the University of Bologna respectively in 2001 and in 2005. His current research interests include distributed systems, complex networks, data science, fintech and blockchain technologies, multimedia communications, hybrid and distributed simulation. He is in the editorial board of the Simulation Modelling Practice and Theory (SIMPAT) journal, Elsevier, and of the Encyclopedia of Computer Graphics and Games, published by Springer. He is in the technical committee of Computer Communications, Elsevier, as well as Online Social Networks and Media, Elsevier. He acted as editor of special issues on other international journals (i.e., Wiley CPE, Elsevier ComCom). He acted as chairs for several conferences and workshops within flagship conferences, e.g., ACM Mobisys, IEEE InfoCom.

Gabriele D'Angelo received the Laurea degree (summa cum laude) in Computer Science in 2001, and a Ph.D. in Computer Science in 2005, both from the University of Bologna, Italy. He is an Assistant Professor at the Department of Computer Science and Engineering, University of Bologna. His research interests include parallel and distributed simulation, distributed systems, online gaming and computer security. Since 2011 he is in the editorial board of the Simulation Modelling Practice and Theory (SIMPAT) journal published by Elsevier and he is a Technical Program Committee member of INFOCOM since 2019.

Víctor Rodríguez-Doncel received his B.Sc. degree in telecommunication engineering from the University of Valladolid, Spain, in 2001 and his Ph.D. degree in computer science from the Technical University of Catalonia, Barcelona, Spain, in 2010. He is an associate professor in the Artificial Intelligence Department at the Universidad Politécnica de Madrid, Spain. He is conducting research within the Ontology Engineering Group in different areas of the Semantic Web, language technologies, and law. He is a coauthor of the Media Value Chain Ontology.