Blockchain-based Data Management for Smart Transportation

Blockchain-based Data Management for Smart Transportation

Mirko Zichichi, Stefano Ferretti, and Gabriele D'Angelo

Abstract Smart services for Intelligent Transportation Systems (ITS) are currently deployed over centralized system solutions. Conversely, the use of decentralized systems to support these applications enables the distribution of data, only to those entities that have the authorization to access them, while at the same time guaranteeing data sovereignty to the data creators. This approach not only allows sharing information without the intervention of a "trusted" data silo, but promotes data verifiability and accountability. We discuss a possible framework based on decentralized systems, with a focus on four requirements, namely data integrity, confidentiality, access control and persistence. We also describe a prototype implementation and related performance results, showing the viability of the chosen approach.

1 Introduction

In the last decade, Intelligent Transportation Systems (ITS) have emerged as a way to efficiently improve mobility, travel security and increase the options for travellers. As defined in the European Union directive 2010/40/EU [12], ITS are advanced applications for the provision of innovative transport and traffic management services, with the ultimate purpose of aiding individuals within the infrastructure to make safe and timely decisions. The general idea is usually that of devising a sort

Mirko Zichichi
Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
e-mail: mirko.zichichi@upm.es

Stefano Ferretti
Department of Pure and Applied Sciences, University of Urbino "Carlo Bo", Italy
e-mail: stefano.ferretti@uniurb.it

Gabriele D'Angelo
Department of Computer Science and Engineering, University of Bologna, Italy
e-mail: g.dangelo@unibo.it

of data management middleware to build advanced applications for the provision of innovative transport and traffic management services, with the aim of enabling users "to be better informed and make safer, more coordinated and 'smarter' use of transport networks" [12]. Vehicles and transportation infrastructures are becoming increasingly "smarter", which means that they are equipped with sensors that track and process a huge amount of different types of information, e.g. data sensed by the interior of the vehicle, the surrounding environment, road conditions, etc. This enables the creation of applications "without embodying intelligence as such", which brings out the real essence of an infrastructure of this kind. The interaction processes between two individuals, or an individual and a vehicle, or an individual and the infrastructure, within the ITS, should include the least possible presence of a human intermediary. All of this constitute a network of user-owned and infrastructure devices that is usually referred as VANET (Vehicular Ad-hoc NETwork) [31]. In this vision, the intelligence shifts from that of a human third-party to that of an artificial intelligence that has been optimized for this use case. This artificial intervention leads to the creation of "innovative services relating to different modes of transport and traffic management" [12], that take advantage of faster processing and better performances. When there are no human intermediaries, indeed, traditional processes become faster to execute.

In addition, the growth of smartphones and Internet-of-Things devices enables individuals' ubiquitous connectivity and the ability to collect environmental and personal information or crowd-sensed data [42]. Thus, users become an active part of the infrastructure itself. The entirety of such crowd-sensed information is essential for building sophisticated smart services that aim at improving traffic management, transportation efficiency and safety, raising awareness about the environment, and thus improving the liveability and health status of the community of a given territory [11, 37, 18].

A variety of applications and protocols can be enforced altogether to obtain advanced and improved transportation systems. However, to fully exploit their potential and promote the development of smart mobility applications and services for social good, several novel challenges must be faced, that require substantial changes in transportation system models. The "desiderata" for such novel applications and systems revolve around data management: more in particular, the mentioned data gathering, communication, analysis and distribution among individuals' vehicles, infrastructures and services. Data sharing is placed on a middle ground between devices that produce data and the systems that process data to create new smart services. Data-driven innovation will bring enormous benefits for ITS users and not only [19]. The generation and sharing of such an amount of data create the need for trading mechanisms that are at the basis of the productivity and competitive markets for smart service providers, but also fundamentals in health, environment, transparent governance and convenient public services. In turn, this creates the need for evaluating data, in terms of interoperability and quality. These two features, together with data structure, authenticity and integrity are key elements for an effective data exploitation [19].

In the context of ITS, one of the main issues is the unreliability of the exchanged information [11, 42, 54]. This problem is typically due to the physical errors of the sensors, malfunctions, poor network and GPS coverage. Such noisy data lead to inaccurate information. Another problem is due to the fact that some users might be interested in deliberately transferring forged information. Examples are insurance frauds, as well as free-riders that decide to share false data, randomly generated without using their sensors, in order to gain some revenues/credits for such fake data sharing. Thus, one of the main goals to pursue is the identification of strategies for the generation and distribution of secure and trustable crowd-sensed information.

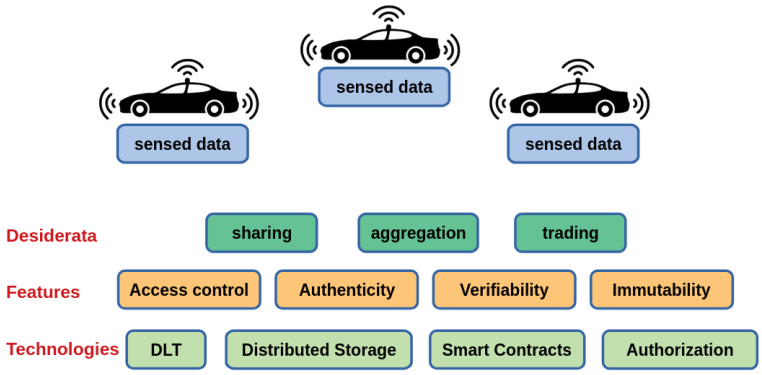

Fig. 1 Intelligent Transportation System data management schema.

The need for trustful data trading and sharing leads to the rationale that anyone should be allowed to verify the authenticity and the immutability of shared information (see Figure 1). From the point of view of the data sharer, on the other hand, the features of verifiability and access control are needed to be combined: making data completely public would make them more verifiable but would also lower their value; but on the other hand, completely closing the access to the data would lower its verifiability.

This is where a (relatively) new kind of technology can come to aid. Distributed Ledger Technologies (DLTs) are thought to provide a trusted and decentralized ledger of data. DLTs are a novel keyword that extends the famous "blockchain" buzzword, to include those technological solutions that do not organize the data ledger as a linked list of blocks. Currently, DLTs are widely utilized in scenarios where: i) multiple parties concur in handling some shared data, ii) there is no complete trust among these parties, and often iii) parties compete for the access/ownership of such data [8, 14]. This is a typical scenario of smart transportation services that exploit data sensed from multiple sources, i.e. vehicles and infrastructure.

DLTs provide the technological guarantees for trusted data management and sharing, as they can offer a fully auditable decentralized access control policy man-

agement and evaluation [33]. Indeed, these decentralized architectures are an ideal choice for data management and sharing [50] because of their features: (i) Transparency: auditability of access permissions by authorized third-parties; (ii) Security: shifting trust towards the consensus mechanism allows mitigating the vulnerabilities of (semi-)trusted intermediaries; (iii) Immutability: on-chain data will always be verifiable; (iv) Peer-to-peer interactions: potential of user-to-user agreements.

On the other hand, Decentralized File Storages (DFS), in combination with DLTs, increase the possibilities in data management and sharing as they provide a range of different but suitable features [55]. They are a potential solution for storing files while maintaining the benefits of decentralization, offering higher data availability and resilience thanks to data replication. Their combined use with DLTs allows overcoming the typical scalability and privacy issues of the latter, while maintaining the benefits of decentralization [38]. 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".

Finally, smart contracts, built upon some DLT implementations, allow checking the terms of an agreement without requiring the presence of a trusted human third-party validator. These may enable auditability of access permissions by authorized third-parties and mitigate privacy vulnerabilities of (semi-)trusted intermediaries when accompanied by off-chain security mechanisms [53].

To sum up, various technologies enable the deployment of viable and scalable systems for the support of smart services in the ITS domain. This work aims to survey the possible ways to handle data management and governance, showing their strengths and limitations, in order to provide a framework. Furthermore, we investigate the feasibility of this framework by offering an implementation based on current DLT and DFS solutions and discussing its performance in comparison to ITS needs.

2 Data Management Strategies

2.1 The classic centralized approach for crowd-sourced data aggregation

The most straightforward approach for managing data and services in ITS resorts to a cloud computing infrastructure (Figure 2) [44]. Vehicles and smartphones collect data and transmit them to the cloud, in platforms where it is possible to extract information and utilize it for models, visualizations, and/or decision-making [32]. In this scenario, cloud computing enables ubiquitous, cost-effective, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and deployed with minimal management effort. Large online platforms, backed up by data centers and centralized computing facilities, provide the advantage of efficient data aggregation, data mining, analysis optimization, storage,

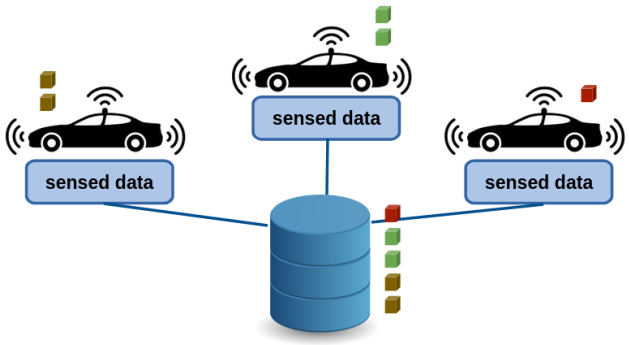

Fig. 2 Crowd-sensed data aggregated in a storage maintained by a single central entity.

batch processing, and computation, i.e. the cloud can compute the gigantic amount of data and complex computations in a very short time [44].

There is a trade-off, however, that leads to imbalances in market power as large online platforms, where a small number of players may accumulate large amounts of data, gather important insights and competitive advantages from the richness and variety of the data they hold [19]. The current practice of data controllers, i.e. entities that collect and manage data coming from users' devices and infrastructure, is to centralize resources in "silos". These controllers usually have a data-driven business model that gives them no incentive to freely share data among each other and to other entities, nor to provide users transparency of their data usage. About the users' location and activities, this information relates to the personal sphere of the individual and composes a part of the dataset called personal data, i.e. any piece of information that can identify or be identifiable to a natural person. Thus, when such a kind of system model is used, it becomes difficult for users to maintain control over their own personal data. That is, individual control, in particular with regard to one's person, has been described as a reflection of fundamental values such as autonomy, privacy, and human dignity [30]. This can indeed represent a problem. It is not by chance that regulations, such as the European Union's General Data Protection Regulation (GDPR) [13] and California Consumer Privacy Act (CCPA) [9], are being implemented, to protect the right that "natural persons should have control of their own personal data" (GDPR p. 2).

2.2 Pure P2P: keep data locally and distribute upon request

At the opposite corner, with respect to the centralized solution, there is a pure peer-to-peer (P2P) approach.

It has been years since P2P technology attracted attention for making it possible to share and exchange resources such as text files, music, videos, uncensored between

one user and another, i.e. peer-to-peer. Initially, it seemed to be expected to become the main component of the Internet, although interest in this technology had waned due to the growth of cloud computing. However, a new wave of interest has recently emerged with the advent of blockchain and cryptocurrencies.

In general, P2P applications usually run on top of an existing network, such as the Internet. This overlay network can support different P2P architectures, usually depending on the type of application that needs to be served. In a P2P environment, a node is not connected to all the other peers in the network, but instead has a limited number of connections to peers that are defined as 'neighbours'. Consequently, the fact that each node is only connected to a certain number of other nodes makes it necessary to relay multiple messages between peers in order to disseminate information to the whole network. Furthermore, there is an aspect to consider in the structure of a P2P network, namely the dynamism of peers that can (freely) join and leave the network. This often requires the use of some protocol to keep the network healthy and connected.

To summarise, there are two important aspects related to the functioning of a P2P system: (i) how messages are exchanged and relayed between peers; (ii) how the overlay is constructed and maintained to cope with churns (i.e. nodes dynamically coming and going in the system) [45, 21].

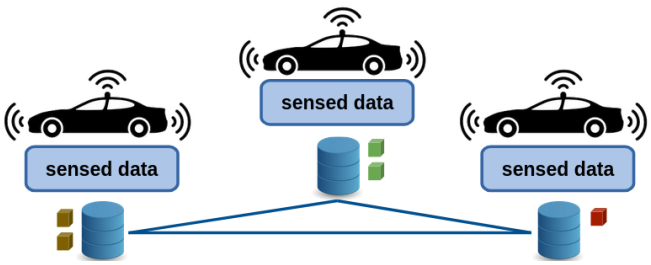

Fig. 3 Pure P2P data aggregation.

In the case of a Vehicular network, the idea here is very basic. As in classic P2P systems, each user's vehicle, or IoT device, or smartphone, maintains locally its generated data (Figure 3). Upon request, it is free to decide if sharing such data with someone else or not. At a first sight, such a solution might seem quite simple to implement. Moreover, it solves a lot of issues concerned with data sovereignty. In fact, each node maintains its data and makes decisions about sharing them.

But clearly enough, this is not a practical solution. In fact, in order to provide sharing capabilities, each user's device should be always connected, i.e. there should be a mechanism cope with churns in the vehicular network. Users' devices should provide some guarantees related to storage, computation and communication capabilities, in order to maintain, handle and transmit their data.

These issues can be solved by switching to an edge-computing like solution. Basically, each user's device has a sort of delegated agent representing it, which is located on the Internet. The device stores its data at this edge node, which is thus in

charge of handling such data. Therefore, storage, computation and communication requirements are shifted to an Internet node, which might more easily provide higher availability guarantees, rather than a user's device. Yet, the availability and reliability of a device's data is as available and reliable as its delegate. Moreover, while this solution is certainly viable, from a distributed system point of view, it still requires some additional protocols to manage the data sharing in ITS. Finally, it does not offer guarantees concerned with traceability, verifiability and immutability of data.

2.3 A Distributed Ledger Technology to register data

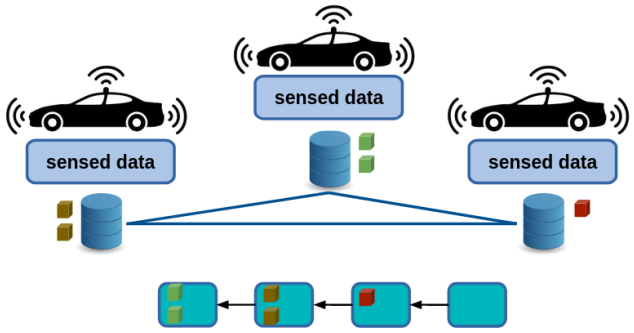

Fig. 4 DLT for data registration.

As we pointed out in the last example, user's vehicles, IoT devices and smartphones equipped with sensors can transfer data to the network, by interacting with a gateway. In the last example, this role was played by a delegated agent. However, we argue that such sensed data can be stored and managed in a DLT network. Thus, each device interacts with a DLT node, transmitting sensed data on a periodical basis. In order to provide a level of traceability, verifiability and immutability of the generated data, the data itself, or a related digest (when data consist of a large file or sensitive information) is added to a DLT [54]. According to this approach, for instance, a vehicle's on-board computing unit is able to issue messages to a DLT node, thanks to authentication. These messages are then converted to transactions added to the ledger. In general, all public DLTs provide such functionalities by exposing APIs that allow entities, external to the DLT, to send novel transactions. The main point here is that these transactions must be registered in the DLT in a fast way. Second, a good level of scalability must be guaranteed. Third, since a high amount of data is produced, the DLT should offer low fees (or no costs at all). Finally, we need to treat all these transactions as a data-stream, easy to retrieve. These main requirements make not all the existing DLTs eligible in this context.

DLTs can be distinguished for their level of scalability and responsiveness. For instance, Ethereum [8] provides a distributed virtual machine able to process any kind of computation through smart contracts. However, it is well known such a blockchain technology has some scalability issues [5]. Conversely, DLTs such as the IOTA DLT [41] provide features thought to guarantee scalability, but they lack the support for smart contracts. By design, IOTA is recognized as a responsive, scalable, feeless DLT, with tools for supporting data streams [54]. Among other solutions, it is worth mentioning the implementation of sharding techniques in DLTs. In a few words, sharding consists in breaking the ledger into smaller, more manageable chunks, and distributing those chunks across multiple nodes, in order to spread the load and maintain a high throughput. Currently, however, these technologies are still in their infancy, e.g. Radix [43], or being developed, e.g. Ethereum 2.0 [23].

2.4 A Decentralized File System for crowd-sensed data

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. To guarantee data integrity and verifiability, encrypted sensed data could be stored directly on the DLT, 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 [55], but also because on-chain personal data are generally incompatible with data protection requirements [22].

A principal example of DFS is the InterPlanetary File System (IPFS) [3], a protocol that builds a distributed file system over a P2P network. IPFS creates a resilient file storage and sharing system, with no single point of failure and without requiring mutual trust between nodes. IPFS [3] is a DFS and a protocol thought for distributed environments with a focus on data resilience. The IPFS P2P network stores and shares files and directories in the form of IPFS objects that are identified by a CID (Content IDentifier).

This technology is useful to store data that is not convenient to put on DLTs, and where, in order to retrieve an object, only the file digest is needed, i.e. the result of a hash function applied on the data. The CID is the result of the application of a hash function to a file and it is used to retrieve the referenced IPFS object in the network. Put in other words, the file digest is the identifier of the IPFS object. Users that want to locate that object use this identifier as a handle. When an IPFS object is shared in the network it will be identified by the CID retrieved from the object hash, for instance a directory with CID equal to *QmbWqxBEKC3P8tqsKc98xmWNzrzDtRLMiMPL8wBuTGsMnR*. If any other node in the network tries to share the same exact directory, the CID will be always the same.

IPFS can be used together with the InterPlanetary Linked Data (IPLD) [27] to ensure that a logical object always map to the same physical digital object. IPLD consists of a set of standards and technologies leveraged to create universally addressable data structures, where the CID itself contains the hash and data decoding information. IPLD enables to link resources identified by hashes that can refer to diverse resources.

3 A Framework for Data Sharing and Management Based on DLTs and DFS

Based on the possible approaches described in the previous section, in this section, we provide a framework for the management and sharing of data in ITS. The main pillar of this proposed framework is the concept of moving the processes for the management of ITS data close to the individual that enacted their production, or at least making them completely transparent to this one. This means, for instance, that a user of a smart vehicle should have the last say on the processing of his own sensed/personal data (e.g. the geo-location while driving) and that both the sensing device manufacturer company and the user should (proportionately) benefit from the value of that data. In our vision, technologies such as DLTs and DFS can help to reach this objective.

DLTs, indeed, allow avoiding all the typical drawbacks of centralized server based approaches (censorship, single point of failure, see Section 2.1), or those of pure P2P applications (no data verifiability and traceability, see Section 2.2). The use of DLTs to represent and transact with data would also grant data validation and access control.

Crucial here is the use of smart contracts, since they provide a new paradigm where unmodifiable instructions are executed in an unambiguous manner during a transaction between two parts. Without the presence of a third-party, smart contract instructions can make sure that the constraints on how and when data are accessed are always respected. Every process is completely traced and permanently stored in the smart contract enabled DLT.

All these properties are necessary in order to create digital data spaces managed both by users and organizations. DFS can help in this sense, since they compensate for certain deficiencies in DLTs. In fact, large sized data can be better handled off-chain, as well as data that are not meant to be stored forever in a distributed ledger. In these cases, DFS are more suitable for data storing; still, this approach can be combined with the use of DLTs, as we well see in this section. Moreover, it is possible to use DFS for maintaining continuous data availability. To sum up, the framework we need must answer three main functional requirements: (i) ensure data integrity, (ii) ensure data confidentiality, (iii) control who has access to data, (iv) ensure data persistence. A solution for each one of them will be detailed in the next subsections and will consist of depicting the same framework from different points of view.

3.1 Data Integrity

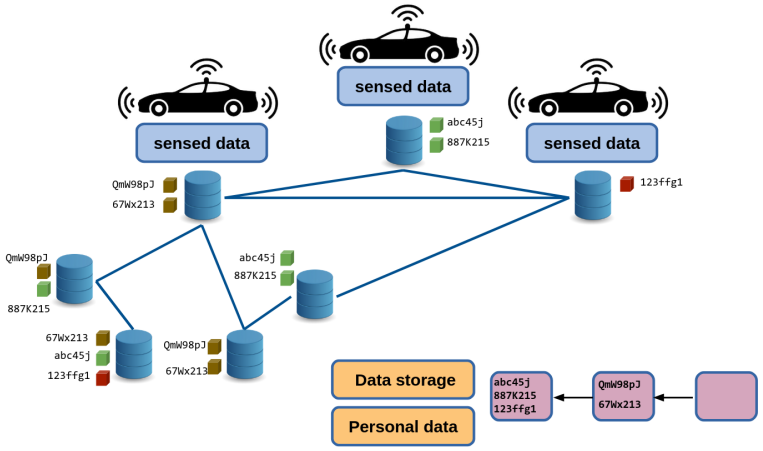

Fig. 5 Data integrity diagram.

We already mentioned that crowd-sensed data, coming from users' smartphones and IoT devices, allow building sophisticated smart services (e.g. to improve traffic management, transportation efficiency and safety) [54]. One requirement, however, is crucial for the creation of secure services and for giving a real value to the sensed data, that is data integrity. To be valuable, indeed, data sensed in an ITS must be reliable in its entirety, and this property should be easily verifiable. DLTs ensure the verification of data integrity in a simple and straightforward way, since the ledger is immutable. Of course, this does not completely assure reliability, as data integrity does not coincide with data security nor quality. Indeed, incorrect information about an assertion can be introduced into the DLT, i.e. the GIGO problem [2]. However, the ledger maintains a trace that makes it possible to investigate the insertion process of data. Thus, DLTs can be leveraged to ensure data integrity. However, this does not necessarily mean that data is stored on-chain. This consideration stems from two observations:

- Storing data into a DFS usually requires lower latencies with respect to DLTs, which typically require some time-consuming consensus mechanism, e.g. Proof-of-Work.
- On-chain data cannot be deleted or modified, becoming an issue when user intentions or regulations require the opposite. For instance, due to the GDPR right to be forgotten or to the right for rectification [13], personal data must not be stored directly in the DLT, even when encrypted [22].

With this in view, the framework considers DFS for data storage, while adopting the mechanism of storing hash pointers in DLTs for content addressed data. In content

addressing, data are identified by their content "fingerprint" instead of their location (such as in the HTTP protocol). A cryptographic hash function is used to identify the content and its result, i.e. a digest, which can be disseminated in a distributed environment to easily refer to the same piece of data. The advantages of content addressing in respect to location addressing are that: (i) links are permanent, i.e. hash pointers; (ii) the link itself does not reveal any of the content, but the content can be used to derive the link; (iii) it increases the integrity of data since altering the content would produce a new link.

It is worth noticing that, storing data off-chain (i.e. in a DFS) and the hash pointers on-chain offer the same levels of data integrity in respect to storing data completely on-chain. Having access to the off-chain stored data, indeed, enables the possibility to compute the hash function over the data and compare the result with the hash pointer that has been immutably stored on-chain.

Moreover, this mechanism enables data deletion [38] and privacy [55], since data in DFS are not immutable and not always public.

3.2 Data Confidentiality

In the previous sub-section, we referred to the process of storing data in DFS, but there is one aspect that needs to be pointed out. Since DFS protocols can be executed in public networks [3], data needs confidentiality before any sharing and/or storing. Indeed, the value of a piece of data also depends on who can access it, e.g. a private information becoming public may lose its value in certain use cases. For this reason, personal data is pre-processed by an encryption algorithm before publishing it to the DFS. We refer to the result of this operation as the "encrypted data". The encryption algorithm can assume any form, but it should be implemented in a way that it does not break data integrity, i.e. it must be possible to verify that the hash pointer and the (encrypted) data correspond. The encryption is a critical part for approaching Privacy by Design [10] and crosses vertically all the other parts of a framework for data sharing. For the sake of simplicity, here we refer to a symmetric encryption algorithm that encrypts a piece of data with a new randomly created symmetric key (but more on this can be found in [53]).

In a simple and generic approach, the personal data generated from a data source (smartphone or IoT device) or held by a data controller is encrypted using a symmetric content key (possibly using an efficient symmetric key cryptography algorithm). It is important to differentiate between two instances of personal data that are needed to protect: (i) types of data that can be defined "static", e.g. personal information regarding the name of a driver rarely changes. In this case, each datum can be protected using a content key with no particular relations to other data and that can be created in a pseudo-random way and then kept in safe. (ii) types of data that more frequently update the property of a person, e.g. the location of a subject can be updated each second in a stream of data. In this case, we mostly deal with time series data that may be more useful when aggregated. Hence it might be more useful to

have content keys related between each others. For instance employing a symmetric key derivation that exploits relations.

Up to this point, the specified framework includes a device within the ITS, e.g. vehicle, IoT device, smartphone, that:

1. fetches a piece of data from a sensor (found in the device or belonging to another device with which it has a direct or indirect communication);
2. encrypts the data with a symmetric key;
3. stores the result, i.e. the encrypted data, on a DFS (making a request to a DFS node);
4. stores the hash pointer of such encrypted data on a DLT (making a request to a DLT node).

It is important to note that steps 1 to 4 can be directly instantiated in a user's personal device, allowing him/her to directly control its data. Alternatively, only the first step can be instantiated in the personal device and the other three can be instantiated in one (or more) devices managed by another entity (that is then considered as a data controller). However, at this point, the framework still misses a mechanism for accessing shared data.

3.3 Data Access Control

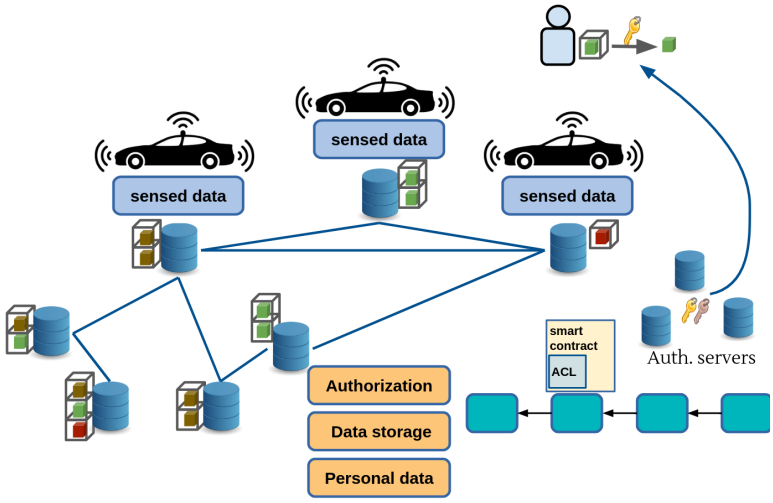

Fig. 6 Data access control diagram.

A second part of the framework, which is complementary to data confidentiality, sees the data consumer as the main actor who is willing to have access to some

shared data. Practically, this actor needs the symmetric key for the decryption of some data and, thus, an authorization service should be placed between the key and this actor. There are several methods to design this service, but here we distinguish between:

- **Centralized** - the most feasible solution, that is where only one service provider is involved in the authorization service and that holds the entire set of secret keys to access data. The data consumer contacts the server directly to retrieve the keys he is eligible to get. This design implies that users trust the server, since this entity has complete access to user data. It also covers the case in which the data provider or controller directly implements this service. The drawback of this approach is that it does not cope with the possibility that the authorization server is honest-but-curious, i.e. it follows the protocol correctly but, if curious, can decrypt and thus access data.
- **Decentralized** - the vision to decentralize the service would help to shift the trust from one entity to a protocol [29]. In this case, indeed, nodes in a decentralized network may be considered semi- or un-trusted, but a consensus mechanism together with a dedicated cryptographic mechanism, would enable the user to be more confident in the protocol [20, 47, 6, 16].

In the framework that we present here, the authentication service leverages a decentralized environment to provide authorizations to consumers for different reasons: (i) to avoid, also in this case, a single point of failure; the failure here includes both service interruption and privacy leakages; (ii) to release the data provider device from the burden of completely handling keys distribution; in fact, this service may become very expensive in terms of communication in case of fine-grained access; (iii) to exploit smart contract distributed computation for implementing a "fair" and automatic access control mechanism; (iv) to exploit DLT's transparency for the auditability of access permissions.

The protocol for the decentralized authentication service used in the framework includes two parts, an on-chain and an off-chain part [53]. On-chain smart contracts are exploited for the management of an Access Control List and for the distributed computation of the access mechanism. However, this is complemented with an off-chain keys distribution mechanism. In particular:

- **On-chain Access Control List (ACL)** - The access to the encrypted data, stored in a DFS, is managed through smart contracts, that regulate access rights to data. In practice, each piece of data can be referenced in a specific smart contract and bundles of data can be referenced through Merkle Trees [34]. The form of this data is the one we have seen in the previous sections. In addition, with regard to the data, the contract can also maintain data schemes and indications to the data kind, in order to have interoperability when interacting with such data [24]. The best way to explloit the data indeed is to provide these schemes in a machine-readable format for specifying what to expect from the them. This is needed by the a possible data consumer in order to better handle the data computation and it can be defined directly by the entity who manages the source device or by means of a specific standard. The smart contract, however, is mostly used to

maintain an Access Control List (ACL) that represents the rights to access some data. Consumers, listed in the ACL through their DLT address, can demonstrate to an authorization service their eligibility to access some particular data. Once a Consumer is eligible to obtain certain content, i.e. he is listed in the ACL, he can access such content through a content key. Service providers can directly verify this information from the ACL and release the key needed to decrypt the encrypted data. Thus, through smart contracts, access to the data can be purchased or can be allowed by the data owner. The release of keys for accessing the encrypted data, then, is authorized only to entitled users.

- **Off-chain Keys Distribution** - When this service of keys distribution is operated by a single central provider, trust must be given to this one, since the keys are kept in one place only. Assuming that this provider is honest-but-curious, such that it follows the protocol correctly (e.g. an online social network sharing a user's vehicle geolocation with a user's friends, if curious, can access to this information), decentralization of the service can be put in place in order to shift the trust to the protocol itself, instead of the single provider. In the decentralized case, nodes in a network can be considered semi- or un-trusted, but a data protection/cryptographical mechanism, built into their execution protocol, allows the whole system to be trusted [51, 56]. When a data consumer is entitled to access some data on-chain (i.e. in a smart contract ACL), it can then request the release of the associated distributed keys to the nodes of such a network. Since it is not possible to store secret keys or decrypt messages on-chain, due to its public execution, an off-chain keys distribution mechanism is needed. However, distributed computation should be used to maintain decentralization in the key distribution process, e.g. MultiParty Computation (MPC) [51, 35]. Two cryptographical schemes can be used in this case for the content keys distribution:
  - **Secret Sharing (SS)** - This scheme splits the content key in n shares, but only t shares are enough to reconstruct the key. A (t, n)-threshold scheme is employed to share a secret between a set of n participants, with the possibility to reconstruct the secret using any subset of t (with $t \leq n$) or more shares, but no subset of less than t. In a network where multiple nodes store secret shares, a consensus can be reached by t nodes to provide the shares to a data consumer, allowing him to know the secret. This can be employed to provide privacy to a user that is sharing a secret, since none of the nodes can obtain the whole secret without the help of other $t - 1$ nodes. Thus, single nodes alone are unable to reconstruct the content key because they only save a portion of this key.
  - **Threshold Proxy Re-Encryption (PRE)** - The content key can be re-encrypted by the a proxy node using the proxy re-encryption (PRE) scheme. PRE [1], is a type of encryption where a proxy entity transforms a ciphertext encrypted with a content key k_1, into a ciphertext decryptable with a key k_2, without learning anything about the underlying plaintext. This is possible using a re-encryption key rk_{1-2} generated by the data owner who has initially

encrypted the plaintext with k_1. PRE is a scheme that usually involves only one semi-trusted proxy node. However, it can be the case that this node decides to re-encrypt data immediately, rather than to apply conditional policies as instructed, or it may collude with the Consumer to attack the data owner's private key. A threshold proxy re-encryption scheme can be used to solve this problem. Instead of using a single re-encryption key, a (t, n)-threshold scheme is used to produce "re-encryption shares", in such a way that these can be combined client-side by the data consumer.

Both these techniques can be supported in a decentralized data access control and come with different advantages and disadvantages. SS relieves the user from any interaction during each key distribution, but at the same time if t nodes are malicious then the user cannot intervene to stop the keys from getting leaked. On the other side, PRE has the drawback of requiring the user to generate a re-encryption key for each new consumer, however he has the option to stop producing new re-keys if some nodes are malicious.

3.4 Data persistence

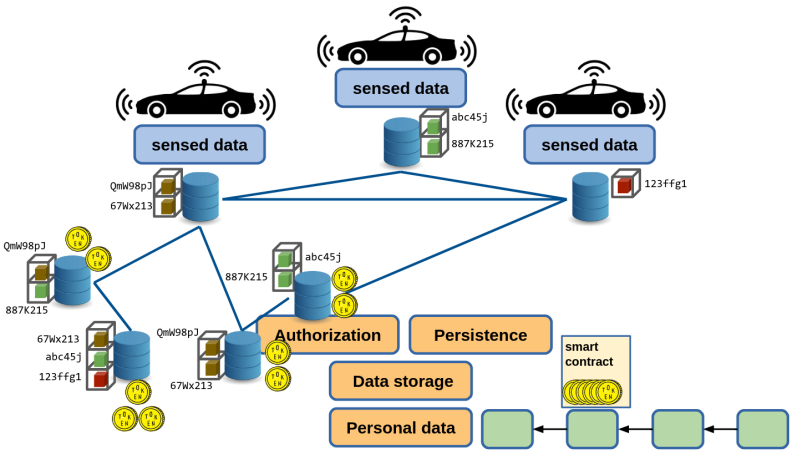

Fig. 7 Data persistence incentives diagram.

In P2P systems a general issue affects the availability of data when the network nodes have no incentive to keep their storage occupied. For instance, it is well known that in file-sharing systems, such as BitTorrent, the data availability of some popular content may become poor quickly, and eventually it is hard to locate and download it [25]. Similarly, when there is no incentive to maintain files, also DFS cannot offer guarantees on the persistence of data. Indeed, data is usually stored

in the DFS as long as some node has some disk space to maintain a replica. The more the nodes that maintain a copy of a given file, the higher the reliability and the higher the guarantees that the file can be properly retrieved. To cope with this issue, incentivisation mechanisms can be employed to obtain that the distributed system permanently stores files, i.e. users can reward nodes that maintain a copy of their data.

In order to provide incentives to nodes for maintaining data, some DFS integrate DLTs, bringing together clients' requests with storage nodes' offers. In practice, participants are rewarded with cryptocurrencies for serving and hosting content on their storage. This strategy does not alter the protocol on how nodes exchange data in the DFS but, "simply", network nodes are paid to store and not erase them. This payment is generally based on a proof that these nodes publish in the integrated DLT, e.g. Proof-of-Spacetime [4].

Some DFS [4, 49] also integrate smart contracts in order to reward hosts for keeping files. These contracts are usually referred to as File Contracts, i.e. a particular kind of smart contract employed to arrange an agreement between a storage provider and their clients.

In Filecoin [4], for instance, file storage is treated as an algorithmic market thanks to File Contracts. Some nodes provide the storage and the prices of the service are not controlled by a single enterprise, but rather depend on a supply-demand market model. In this model, the user pays miners to store and retrieve files:

- A storage agreement is an agreement between a user and a storage miner in which the former pays in advance and the latter periodically demonstrates that it is still storing the file until the expiry time. In addition, the user can auction the storage to miners who meet certain requirements. The storage miner stores files for users and extracts additional blockchain coins by performing storage tests. The user is guaranteed storage by storing evidence on the blockchain.
- A retrieval deal is an agreement between user and retrieval miner in which the latter retrieves files from storage for the former. This means that the retrieval miner can be a different entity from a storage miner. Unlike storage deals, retrieval deals are managed off-chain and their value may depend on the speed of retrieval. Thus, payments are made off-chain, are incremental and are typically based on reliability or speed of recovery.

In the Filecoin blockchain, storage miners require powerful hardware in addition to storage, as they are also tasked with creating new blocks (every 30 seconds) and running proofs. The blockchain acts as a ledger for proofs, agreements and coin transfers, therefore storage proofs are public. An important concept in generating proofs for Filecoin is the sector, i.e. the basic storage unit. The size and time increments for commitments are standardised and are chosen as a trade-off between security and usability, while the duration of a sector depends on the storage market. A sector is fully occupied by commitments made through File Contracts and must be sealed before proof. Unused space is considered committed by auto-deal, i.e. by the miner to itself.

The sector and its content ID (CID) are sealed in a replica, i.e. encoded in a way that can be used for proofs. Then the Proof of Replication (PoRep) is performed on the replica. This proof is based on the fact that an honest storage miner has already sealed the blocks and can respond quickly before a timeout. The sectors are sealed in a time-consuming process (due to the fact that the sectors are 32Gb or 64Gb) using a key that depends on the node hardware. If a user wants their files stored as multiple copies on different miners, each copy will have a different seal and an attacker would have to download each block of the file and reseal it.

Over time, randomly selected storage miners have random sectors questioned, from which the data is read for verification and compressed into a Proof of SpaceTime (PoSt). Both users (on demand) and miners (periodically) check that blocks are valid and storage miners are "punished" if blocks are not available before an agreement expires. Storage miners, therefore, are required to provide public (i.e., stored on-chain) proof that a given data encoding has existed in physical storage continuously for a period of time. This means that all miners are asked to perform a PoRep on a random sector of their storage when mining blocks, i.e. a WinningPoSt, or every day, i.e. WindowPoSt.

4 Implementation and Evaluation

In this section, we will discuss the feasibility of the implementation of the presented framework. Our implementation features several technologies that fall into the realm of DLTs and DFS. Moreover, we will provide an assessment of the performance of such implementation considering a specific ITS use case and its general constraints.

4.1 Implementation

4.1.1 IPFS as DFS and Filecoin for incentivization

Within the framework, the DFS plays the role of common data space, where all the system components store data. In our implementation, such a role is played by IPFS [3]. Thus, when a piece of data is uploaded to the network, the data digest is returned as a reference. This reference can be stored in the DLT and then employed to retrieve the specific data. Thus, the piece of data is published as an IPFS object and then (asynchronously) referenced through its hash into a DLT transaction. The digest, as explained before, allows verifying the integrity of the IPFS object.

To upload files on IPFS, a node running the IPFS protocol is necessary. Due to the fact that it is (still) not feasible to run an IPFS node on constrained devices (such as smartphones or vehicles on board units), other solutions must be explored. In our implementation, we resort to an IPFS service provider (i.e. Infura [26]) to let users disseminate files in the IPFS network.

Data persistence is implemented through incentivization, thanks to the use of Filecoin smart contracts [4]. Filecoin is the typical incentive used on top of IPFS, where participants are rewarded with Filecoin tokens for serving and hosting content on their storage.

4.1.2 IOTA as DLT

For what regards the DLT implementation, we refer to IOTA. The IOTA ledger is not structured as a blockchain, but instead as a Direct Acyclic Graph (DAG) called the Tangle [41]. Such public data ledger claims to be particularly targeted towards the IoT industry. The IOTA transactions validation approach, indeed, is thought to address two major pain points that are associated with traditional blockchain-based DLTs, i.e. latency and fees. IOTA has been designed to offer fast validation, and no fees are required to add a transaction to the Tangle [7]. This makes IOTA a candidate choice to support smart services built through crowd-sensed data.

An important feature offered by IOTA is the Masked Authenticated Messaging (MAM). MAM is a second layer data communication protocol that adds functionality to emit and access an encrypted data stream over the Tangle. Data streams assume the form of channels, formed by a linked list of transactions in chronological order. Once a channel is created, only the channel owner can publish encrypted messages. Data consumers that possess the MAM channel encryption key are enabled to decode the message. Messages are pushed on the channel in chronological order, and each message has a link to the next message to be created. This allows access from one message onward, while restricting access to prior messages in the channel [7].

The data access to new data may be revoked simply by using a new encryption key. We consider that the hash pointer of each piece of data stored in IPFS is stored in a MAM message, and the symmetric content key to be shared with data consumers is the MAM encryption key of the message.

4.1.3 Decentralized Access Control based on Ethereum Smart Contracts and Cryptographic Threshold Schemes

The access control is fully managed by several predetermined Authorization Servers. These nodes perform on-chain tasks related to the smart contracts execution, but also off-chain tasks such as the keys distribution. As to on-chain tasks, a set of Ethereum smart contracts [8] are used to implement the access control. These smart contracts refer to the information that is written into IOTA MAM Channels, i.e. hash pointers, using the MAM Channels addresses called “roots”. The software that has been used for managing the ACL is the OpenEthereum client [36], a popular Ethereum blockchain client. In particular, it offers the implementation of a “SecretStore” where nodes can distribute keys on the basis of smart contracts extracted information. Regarding off-chain keys distribution, we refer to two cryptographic schemes:

- **OpenEthereum Secret Sharing (SS)** - To implement this scheme, we resorted to the Secret Store provided by the OpenEthereum client [36], in which the content key k is split in n shares and $t < n$ are enough to reconstruct it. Considering the key to decrypt a data as a secret, the secret sharing among a network of nodes enables to be in a situation where, potentially, up to $t-1$ nodes can be malicious. Indeed a consensus is reached by from t nodes upward when a data consumer asks to receive a content key. Moreover, any node in the network created through the OpenEthereum cannot access the data on its own, as it would need the help of other $t-1$ nodes.
- **Umbral Threshold Proxy Re-Encryption (TPRE)** - In Umbral [16], the content key k is initially encrypted in a public key encryption scheme using the public key of the data provider. The result of such an operation is then used in a (t,n)-threshold Proxy Re-Encryption schema, i.e. it can be re-encrypted using t "re-encryption shares". This process will produce a re-encrypted key that can be decrypted using the private key of the data consumer to obtain the initial content key k. Among many schemes, Umbral uses a single-use uni-directional proxy re-encryption where the re-encryption function is one way.

4.2 Performance Evaluation

Our experimental scenario was based on a hypothetical real ITS application. In particular, we conducted a trace-driven experimental evaluation where traces were generated using a real dataset of mobility traces of buses in Rio de Janeiro (Brazil) [15]. Based on these traces, we simulated several users' devices on board of buses that, during their path, periodically generate sensed data. We considered one user for each bus. These data may represent temperatures, air pollution values, etc. In this case, we focused on two different types of data: (i) small-sized data, such as hash pointers or geodata, i.e. latitude and longitude (100 bytes); (ii) large-sized data, such as photos (1 MB). Here, we present a summary of results which are discussed in detail in [52, 54, 55].

Table 1 Results on IOTA, with 60, 120, 240 users. [54, 52]

# users	Heuristic	Avg Latency	Conf. Int. (95%)	Errors
60	Fixed Random	72.68 sec	[70.43, 74.94] sec	15.37%
	Adaptive RTT	22.99 sec	[22.69, 23.29] sec	0.81%
120	Fixed Random	87.75 sec	[85.38, 90.12] sec	29.49%
	Adaptive RTT	27.35 sec	[27.11, 27.58] sec	1.1%
240	Fixed Random	177.62 sec	[174.25, 181.0] sec	42.81%
	Adaptive RTT	73.26 sec	[72.68, 73.85] sec	7.55%

4.2.1 IOTA MAM Channels

One user per bus was emulated by a single process issuing messages containing a hash pointer to a MAM Channel, based on the data-trace (a MAM channel was associated with each user). Based on the bus paths, each user was set to generate approximately 45 messages/hour. This resulted in a message to be issued to the DLT every 80 sec, which is a reasonable time interval to sense data in an urban scenario. For each test configuration, we replicated the experiment 12 times for 1 hour. We considered different heuristics for the selection of a IOTA full node from a (dynamic) pool of public nodes to pair to each user ($\sim$ 60 active nodes). The rationale behind this choice was based on the assumption that users' smartphones or buses' computing units may not have computation capabilities to behave as full nodes [17]. One of the heuristics, called Fixed Random, requires each user to be assigned to a random IOTA full node from the pool for the whole duration of the test. Another heuristic was the Adaptive RTT: each user keeps a trace of past interactions with full nodes and creates a ranking based on the experienced Round Trip Time (RTT) [28]; then for each message to be uploaded on IOTA a full node is chosen based on ranking. For each MAM message, we recorded the outcome of the upload request, i.e. successful or unsuccessful, as well as the latency between the transmission of the message to a node and the confirmation of its insertion in the ledger. This interval of time is characterized mainly by two operations that are needed for storing the transaction in the IOTA ledger: (i) the tips selection consists of selecting from the Tangle two random transactions that do not have a successor yet, i.e. tips; (ii) the Proof-of-Work, that requires computation to obtain a piece of data difficult (costly and time-consuming) to produce but easy to verify [41].

Table 1 shows a summary of the results obtained for different repetitions of a specific test [54, 52]. Two main measures experienced during a series of tests are reported: (i) the average latencies including both the tips selection and PoW phases, and (ii) the percentage of errors, that is the number of messages that failed to be added to the Tangle, due to full nodes' errors. Results show that, on one side, the measured latencies are relevant. Indeed, the random selection of a full node for issuing a transaction does not lead to good results, since the amount of errors is quite high, as well as the measured latencies. On the other hand, the good news is that if we carefully select the full node to issue a transaction, the performances definitely improve. In fact, the use of the "Adaptive RTT" heuristic has a low amount of errors, on average around 0.8% and average latency amounts to 23 seconds. However, this is still far from a real-time update of the DLT and the level of acceptability of latency values truly depends on the application scenario. In terms of scalability, results in Table 1 show that in all cases, average latencies increase significantly with the number of users. There is an important difference between the 60 and 240 users scenario. In the case of 240 users, we have a message generation rate of about $\sim$ 3 msg/sec to be issued to the IOTA DLT. If we assume that the workload is evenly distributed among all the nodes in the pool, then, each node receives on average a new message request every $\sim$ 20 sec. Bearing in mind that, at best, it takes 23 sec for a full node to completely process a message, then we see that an initial overhead

of a few seconds leads to a huge increase at the end of the test, i.e. ~ 73 sec in the "Adaptive RTT" heuristic. This means that further improvements are needed to solve scalability issues in this scenario.

Table 2 Latencies and errors when sending messages to IPFS nodes [55].

# users	IPFS Node	Data Size	Avg Latency	Conf. Int. (95%)	Errors
10	Proprietary	Small	0.19 sec	[0.18, 0.2] sec	0.0%
		Large	1.22 sec	[1.17, 1.28] sec	0.0%
	Service	Small	9.49 sec	[9.09, 9.9] sec	0.0%
		Large	6.16 sec	[5.75, 6.57] sec	0.0%
40	Proprietary	Small	0.59 sec	[0.57, 0.62] sec	0.0%
		Large	3.42 sec	[3.31, 3.54] sec	0.0%
	Service	Small	7.5 sec	[7.18, 7.83] sec	0.0%
		Large	11.3 sec	[11.01, 11.58] sec	0.0%
70	Proprietary	Small	2.65 sec	[2.56, 2.74] sec	0.0%
		Large	7.48 sec	[7.3, 7.66] sec	0.0%
	Service	Small	6.22 sec	[6.09, 6.34] sec	0.0%
		Large	8.58 sec	[8.42, 8.74] sec	0.0%
100	Proprietary	Small	1.53 sec	[1.48, 1.58] sec	0.0%
		Large	20.27 sec	[19.71, 20.83] sec	83.33%
	Service	Small	6.81 sec	[6.69, 6.92] sec	0.0%
		Large	12.91 sec	[12.68, 13.14] sec	0.21%

4.2.2 IPFS

In this assessment, we used a single DFS node with two different implementations, while varying the number of users, i.e. we tested different cases with a specific number of users associated with a single DFS node. We assessed two different scenarios: i) we setup an IPFS node, i.e. Proprietary, on a dedicated device connected to other nodes in the main IPFS network and devoted it to only handle requests coming from our application; ii) we tested a public IPFS node Service, i.e. the Infura service provider [26], that offers free access to IPFS. Tests were conducted in order of dimension (small files first, then larger ones) and users number (from 10 to 100). The performance evaluation has been designed as a stress test in which each simulation sends requests to the two different types of IPFS nodes following the buses real traces. A simulation lasted 15 minutes and sent exactly 15 messages for each user.

Table 2 shows the latency recorded during the tests. In general, we noticed better performance when a dedicated node is employed. The results show that the IPFS Service has a similar behavior with both small and large files. This is due to the fact that it has more resources than the Proprietary node, which is then unable to cope with larger files. In the stress test that we implemented, the IPFS Proprietary performances get worse when increasing the number of users. Furthermore, there is a turning point with 80 users where, overall, performance degrades in the presence

of large files, while latencies with small files remain stable (or even decrease). In general, IPFS Proprietary always works better except for over 80 users in the case of large files. This means that a dedicated node is always preferable, but must be limited to a rate of 60-70 users requests per minute.

4.2.3 Decentralized Access Control

In a second experiment we measured the amount of time required to perform access control operations using the implementation of the keys distribution, i.e. OpenEthereum client [36], called SS, and Umbral [16], called TPRE. The tests were performed using a network of 25 interconnected nodes with the aim to emulate the real DLTs and the distributed systems use cases.

Table 3 Threshold latencies (mean) when encrypting (+ distributing) and decrypting messages for a Decentralized Access Control [53].

Threshold	SS Encryption	TPRE Encryption	SS Decryption	TPRE Decryption
5	1024 ms	176 ms	192 ms	130 ms
10	1017 ms	182 ms	233 ms	189 ms
15	1030 ms	183 ms	265 ms	245 ms
20	1045 ms	185 ms	349 ms	309 ms
25	1069 ms	190 ms	371 ms	376 ms

Table 4 Nodes number latencies (mean) when encrypting (+ distributing) and decrypting messages for a Decentralized Access Control [53].

# nodes	SS Encryption	TPRE Encryption	SS Decryption	TPRE Decryption
5	397 ms	120 ms	148 ms	104 ms
10	549 ms	135 ms	148 ms	101 ms
15	666 ms	147 ms	175 ms	108 ms
20	843 ms	163 ms	178 ms	108 ms
25	952 ms	175 ms	188 ms	110 ms

We emulated from 10 to 100 data consumers asking for access to the user's bus data after they have been added to the ACL, then we averaged the result. The results of the test carried out allow us to evaluate the goodness system in terms of performances:

- **Threshold variation**: involves the variation of t from 5 to 25, with number of nodes $n = 25$ and message size set to 30 Bytes. As the Table3 shows, the encryption latency time remains mostly constant, $\sim$180ms for TPRE and $\sim$1045ms for SS, while the decryption time increases linearly with t. The biggest time difference between the two schemes comes from the actual generation and

Table 5 Message size latencies (mean) when encrypting (+ distributing) and decrypting messages for a Decentralized Access Control [53].

Size	SS Encryption	TPRE Encryption	SS Decryption	TPRE Decryption
10B	1026 ms	174 ms	126 ms	111 ms
50B	1022 ms	174 ms	126 ms	109 ms
100B	1025 ms	177 ms	125 ms	110 ms
500B	1025 ms	175 ms	125 ms	109 ms
1KB	1027 ms	176 ms	126 ms	108 ms
5KB	1027 ms	185 ms	129 ms	109 ms
10KB	1031 ms	178 ms	135 ms	109 ms
50KB	1071 ms	177 ms	178 ms	110 ms
100KB	1127 ms	178 ms	231 ms	116 ms
500KB	1541 ms	196 ms	642 ms	127 ms
1MB	2054 ms	220 ms	1150 ms	151 ms
5MB	6214 ms	394 ms	5278 ms	305 ms
10MB	11456 ms	608 ms	10452 ms	502 ms

distribution of the key shares (in the encryption phase), i.e. a surplus of ∼792ms from TPRE to SS.

- **Number of nodes variation**: threshold value t was set to 2 and the message size was set to 30 KB. Generally, as expected, the time costs of operations increase with the number of nodes n. However, we must note the fact that the results in Table 4 grow much faster for SS rather than for TPRE. This makes the TPRE method more scalable.
- **Size of messages variation**: n was set to 25 and $t = 2$, while the size of the message varied. Results reported in the Table 5 suggest that the TPRE scheme scales better than SS. From 10 Bytes to 1 MB TPRE latency raises slightly, while SS has a clear inflection point when the message size is set to 100 KB and then skyrockets from 1 MB onward.

5 Discussion

To fully exploit their potential and to promote the development of ITS, several novel challenges must be faced. In ITS, data gathering, communication, analysis and distribution among individuals vehicles, infrastructures and services can be built in a “centralized” way or in a “pure P2P”, as shown respectively in Sections 2.1 and 2.2. However, in both cases it may become difficult for users to maintain control over their own data and guarantees for data traceability, verifiability and immutability are not always met. For this reason, we argue that the integration of DLTs, DFS and smart contracts can help with the creation of a framework providing data integrity, confidentiality, access control and persistence. An important and critical outcome of this work is concerned with the implementation and experimental assessment we performed, showing the related results of the technologies available at the moment.

It is well known that decentralized and secure DLTs that enable the distributed execution of smart contracts, such as Ethereum in our implementation, still have scalability issues [5]. Here, thus we focused on testing out DLTs and DFS where data is uploaded.

Latencies measured to store data into the considered DFS, i.e. IPFS, can be considered acceptable for general ITS scenarios. In this case, as a measure of scalability, the best performances were obtained when the number of dedicated IPFS nodes followed the equation $\#nodes = \frac{\#requests}{sec}$, where $\#requests$ is the number of IPFS upload requests generated by the users in our scenario. On the other hand, for what concerns the employed DLT, i.e. IOTA, we conclude that at the moment of the test execution the results were not viable for real-time ITS applications, but acceptable for less demanding services. Tests show a latency between 23 and 27 seconds for 0.75 to 1.5 MAM messages insert requests per second, with, at best, an experienced tps, i.e. transactions per second, of 0.13 (considering 1 MAM message roughly equal to 3 IOTA transactions). This means that, for a latency on average of $\sim$ 25 seconds, with a configuration similar to ours during tests, available IOTA nodes in such a scenario should scale following $\#nodes \geq \frac{k \times \#requests}{sec}$, with $k = 53$ and where $\#requests$ is the number of MAM messages insert requests generated by the users in our scenario. Hypothetically, having a DLT protocol that allows $tps = 1$, would require having available IOTA nodes in such a scenario that scale following the same formula but with $k = 2$.

We have focused on data protection through encryption, using For what does concern decentralized access control we employed two different schemes, i.e. SS and TPRE. At first we discussed their qualitative differences, then we compared them in terms of execution time. Our performance evaluation shows that, in respect to SS, TPRE is: (i) faster when increasing the size of the messages; (ii) more scalable, as it better manages the increase in the number of nodes executing the protocol; (iii) more efficient when increasing the threshold value, due to its shares generation method. On the other hand, TPRE has the drawback of requiring the data owner to generate a re-encryption key for each new data consumer.

Clearly enough, an adequate ITS infrastructure must be set in all the cases, in order to build a scalable architecture able to properly handle a possibly high data generation rate from multiple vehicles. In other words, we think that the issue is concerned more with the system deployment rather than on the DLT/DFS protocol. For instance, an edge computing architecture can be merged with the framework and used to geographically place node gateways, which receive data from vehicles and insert them into DLT/DFS.

5.1 New DLT proposals

To overcome the scalability issues described in the above discussion, several new approaches are emerging in the DLTs scenario. New proposals include solutions

where improvements are made directly on-chain, i.e. at layer one, and solutions that build on top of that layer and are executed off-chain, i.e. layer two.

Off-chain solutions are implemented separately from the layer one DLT protocol and require no changes, in order to derive their security directly from the layer one consensus. Optimistic Rollups, for instance, are a layer two scaling solution built for the Ethereum blockchain where computation is partly executed off-chain and data is maintained on-chain [46]. Its aim is to increase the blockchain transactions per second by a factor of a hundred, or even a thousand, and then also to decrease transaction fees. Another layer two scaling solution is the use of side chains. In particular, it consists of using another blockchain, i.e. the sidechain, that runs a faster or lighter protocol, in order to manage assets in the original blockchain, i.e. mainchain. For instance, the sidechain can ensure asset security using the Plasma framework and a decentralized network of Proof-of-Stake validators [40]. Furthermore, there is currently a rise of technologies that operate using the same protocol for executing Ethereum Smart Contracts but with fewer latencies and reduced gas price. For instance, Polygon [39] has achieved up to 10 000 transactions per seconds on a single sidechain through internal tests. In general, it has been shown that implementing an Ethereum private network using Proof of Authority consensus mechanism, with optimal configuration, can reach up to 1000 transactions per second [48].

On-chain solutions usually involve the use of different forms of ledgers, for instance a DAG such as in IOTA or a sharded ledger. Given that we already presented and discussed on the IOTA DLT latencies, furthermore we conducted preliminary tests with other possible DLTs that implement novel techniques to improve responsiveness and scalability, i.e. Radix, which is specifically based on sharding, obtaining interesting results [43]. In a few words, sharding consists in breaking the ledger into smaller, more manageable chunks, and distributing those chunks across multiple nodes, in order to spread the load and maintain a high throughput. At the time of writing, the Radix technology is still in its infancy. Nevertheless, we exploited the test network to issue transactions on the ledger. Thus, obtained results cannot be considered accurate and it is too early to give an overall judgment on this DLT. However, we obtained very low latencies (below 1 sec), with a non-negligible (but low) error rate. We stress the fact that these results cannot be compared with those obtained for IOTA. In fact, in IOTA we exploited the main network, while in Radix we had to employ a preliminary testnet, with few nodes involved in the ledger management ($\sim$ 6 nodes) and basically no additional workload, apart from our tests. As a matter of fact, comparable results can be obtained if tests are executed on the IOTA test network, where the PoW is faster (we obtained average latencies around $\sim$ 2 sec). Moreover, two possible problems must be faced in this case. The first one is related to the fact that Radix requires fees to add a transaction to its ledger. This might make the costs, for supporting a smart transportation system application, prohibitive. Moreover, an open security question arises, i.e. if we decrease the number of nodes that validate transactions (as the sharding does), then does the risk of a security breach increase?

6 Conclusions

The framework we presented in this work shows a possible specification for taking advantage of decentralized architectures to build reliable and modern services for Intelligent Transportation Systems (ITS), on the basis of data sharing and management. The resulting framework integrates DLTs, DFS, smart contracts and authorization protocols. This was a response to the need and importance of being able to optimize the use of resources and data in ITS, but not limited to that use case. The solution we have shown optimizes data sharing from four points of view, namely data integrity, confidentiality, access control and persistence. Our experimental evaluation of the implementation of such a system using currently available technologies shows an acceptable feasibility for less demanding ITS services, i.e. non-real-time services. However, many issues are left open and pave the way for new studies on the optimization of such a system and the integration of new technologies: (i) the optimization of the use of algorithms for managing distributed ledgers (many solutions are particularly CPU and network intensive), (ii) the optimization of data placement, (iii) the optimization of the infrastructure supporting such a distributed system by an ad-hoc deployment of nodes.

Acknowledgements

This work has received funding from the European Union H2020 research and innovation programme under the MSCA ITN European Joint Doctorate grant agreement No 814177 Law, Science and Technology Joint Doctorate - Rights of Internet of Everything.

References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security (TISSEC) **9**(1), 1–30 (2006)
2. Babich, V., Hilary, G.: Om forum—distributed ledgers and operations: What operations management researchers should know about blockchain technology. Manufacturing & Service Operations Management **22**(2), 223–240 (2020)
3. Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014)
4. Benet, J., Greco, N.: Filecoin: A decentralized storage network. Protoc. Labs (2018)
5. 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)
6. Blakley, G.R.: Safeguarding cryptographic keys. In: 1979 International Workshop on Managing Requirements Knowledge (MARK), pp. 313–318. IEEE (1979)
7. Brogan, J., Baskaran, I., Ramachandran, N.: Authenticating health activity data using distributed ledger technologies. Computational and Structural Biotechnology Journal **16** (2018)

8. Buterin, V., et al.: Ethereum white paper (2013). URL https://github.com/ethereum/wiki/wiki/White-Paper
9. California State Legislature: California consumer privacy act (2020). URL https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375
10. Cavoukian, A.: Privacy by design. Take the challenge. Information and privacy commissioner of Ontario, Canada (2009)
11. Chiasserini, C.F., Giaccone, P., Malnati, G., Macagno, M., Sviridov, G.: Blockchain-based mobility verification of connected cars. In: 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC), pp. 1–6 (2020)
12. Council of European Union: Directive 2010/40/eu on the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport
13. Council of European Union: Regulation (eu) 2016/679 - directive 95/46
14. 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). DOI 10.1145/3211933.3211950
15. Dias, D., Costa, L.H.M.K.: CRAWDAD dataset coppe-ufrj/riobuses (v. 2018-03-19). Downloaded from https://crawdad.org/coppe-ufrj/RioBuses/20180319 (2018). DOI 10.15783/C7B64B
16. Egorov, M., Wilkison, M., Nuñez, D.: Nucypher kms: decentralized key management system. arXiv preprint arXiv:1707.06140 (2017)
17. Elsts, A., Mitskas, E., Oikonomou, G.: Distributed ledger technology and the Internet of Things: A feasibility study. In: Proc. of the 1st Workshop on Blockchain-enabled Networked Sensor Systems (BlockSys) (2018)
18. ETSI: Etsi en 302 637-2 v1.4.1. https://www.etsi.org (2014)
19. European Commission: A european strategy for data (2020)
20. European Union Agency for Cybersecurity: Data Pseudonymisation: Advanced Techniques & Use Cases. Tech. rep., European Union Agency for Cybersecurity (2021). URL https://www.enisa.europa.eu/publications/data-pseudonymisation-advanced-techniques-and-use-cases
21. Ferretti, S.: Shaping opportunistic networks. Computer Communications **36**(5), 481–503 (2013). DOI 10.1016/j.comcom.2012.12.006
22. Finck, M.: Blockchain and the General Data Protection Regulation: Can Distributed Ledgers be Squared with European Data Protection Law?: Study. European Parliament (2019)
23. Foundation, E., et al.: Ethereum 2.0 specifications (2021). URL https://github.com/ethereum/eth2.0-specs
24. Foundation, T.S.: Innovation meets compliance: Data privacy regulation and distributed ledger technology. Tech. rep., The Sovrin Foundation (2020)
25. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Zhang, X.: A performance study of bittorrent-like peer-to-peer systems. IEEE Journal on selected areas in communications **25**(1), 155–169 (2007)
26. Infura Inc: Infura: Secure and scalable access to ethereum apis and ipfs gateways. (2020). URL https://infura.io/
27. IPLD Team: Interplanetary linked data (ipld). *https://specs.ipld.io/* (2016)
28. Jacobson, V.: Congestion avoidance and control. In: ACM SIGCOMM Computer Communication Review, vol. 18. ACM (1988)
29. 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)
30. Koops, B.J., Newell, B.C., Timan, T., Skorvanek, I., Chokrevski, T., Galic, M.: A typology of privacy. U. Pa. J. Int'l L. **38**, 483 (2016)
31. Leiding, B., Memarmoshrefi, P., Hogrefe, D.: Self-managed and blockchain-based vehicular ad-hoc networks. In: Proc. of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (2016)

32. Lucic, M.C., Wan, X., Ghazzai, H., Massoud, Y.: Leveraging intelligent transportation systems and smart vehicles using crowdsourcing: An overview. Smart Cities **3**(2), 341–361 (2020)
33. Maesa, D.D.F., Mori, P., Ricci, L.: A blockchain based approach for the definition of auditable access control systems. Computers & Security **84**, 93–119 (2019)
34. Merkle, R.C.: A digital signature based on a conventional encryption function. In: C. Pomerance (ed.) Advances in Cryptology — CRYPTO '87, pp. 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg (1988)
35. Micali, S., Goldreich, O., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC, pp. 218–229. ACM (1987)
36. OpenEthereum: OperEthereum Secret Store (2020). URL `https://openethereum.github.io/Secret-Store`
37. Palazzi, C.E., Roccetti, M., Ferretti, S.: An intervehicular communication architecture for safety and entertainment. IEEE Transactions on Intelligent Transportation Systems **11**(1), 90–99 (2010). DOI 10.1109/TITS.2009.2029078
38. Politou, E., Alepis, E., Patsakis, C., Casino, F., Alazab, M.: Delegated content erasure in IPFS. Future Generation Computer Systems **112**, 956–964 (2020). DOI 10.1016/j.future.2020.06.037
39. Polygon: Polygon - ethereum's internet of blockchains (2021). URL `https://polygon.technology/papers/`
40. Poon, J., Buterin, V.: Plasma: Scalable autonomous smart contracts. White paper pp. 1–47 (2017)
41. Popov, S.: The Tangle (2016). URL `https://iota.org/IOTA\_Whitepaper.pdf`
42. Prandi, C., Mirri, S., Ferretti, S., Salomoni, P.: On the need of trustworthy sensing and crowdsourcing for urban accessibility in smart city. ACM Trans. Internet Technol. **18**(1) (2017). DOI 10.1145/3133327
43. RadixDLT: Radix knowledge base (2019). URL `https://docs.radixdlt.com/kb/`
44. Raza, S., Wang, S., Ahmed, M., Anwar, M.R.: A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. Wireless Communications and Mobile Computing **2019** (2019)
45. Serena, L., Zichichi, M., D'Angelo, G., Ferretti, S.: Simulation of dissemination strategies on temporal networks. In: Proc. of the 2021 Annual Modeling and Simulation Conference (ANNSIM), pp. 1–12. Society for Modeling and Simulation International (SCS) (2021)
46. Sguanci, C., Spatafora, R., Vergani, A.M.: Layer 2 blockchain scaling: a survey. arXiv preprint arXiv:2107.10881 (2021)
47. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
48. 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). DOI 10.1109/ACCESS.2020.3011876
49. Vorick, D., Champine, L.: Sia: Simple decentralized storage. Nebulous Inc (2014)
50. Xie, J., Tang, H., Huang, T., Yu, F.R., Xie, R., Liu, J., Liu, Y.: A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges. IEEE Communications Surveys & Tutorials **21**(3), 2794–2830 (2019). DOI 10.1109/comst.2019.2899617
51. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), pp. 162–167. IEEE (1986)
52. Zichichi, M., Ferretti, S., D'Angelo, G.: Are Distributed Ledger Technologies Ready for Intelligent Transportation Systems? In: Proc. of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock 2020), co-located with the 26th Annual International Conference on Mobile Computing and Networking (MobiCom 2020), ACM, pp. 1–6. ACM (2020)
53. 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)
54. 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 pp. 100384–100402 (2020)

55. 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)
56. Zyskind, G., Nathan, O., et al.: Decentralizing privacy: Using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops, pp. 180–184. IEEE (2015)