Using PROV and Blockchain to Achieve Health Data Provenance

Massimiliano Masi, Abdallah Miladi
{massimiliano.masi,abdallah.miladi}@tiani-spirit.com

Andrea Margheri, Vladimiro Sassone
{a.margheri, vsassone}@soton.ac.uk

Jason Rosenzweig
jason.rosenzweig@gmail.com

—WORKING PAPER—

Abstract. *Provenance* is the foundation of data quality, usually implemented by automatically capturing the trace of data manipulation over space and time. In *healthcare*, provenance becomes critical since it encompasses both clinical research and patient safety. In this proposal we aim at exploiting and innovating existing health IT deployments by enabling data provenance queries for all kind of clinical information from anywhere. The proposed technical solution exploits the novelty and the peer-to-peer fashion of the *blockchain* technology and *smart-contracts* to instrument international standards such as IHE and HL7 with a *provenance system robust to fraudulences*.

1 Introduction

Digital healthcare, Health IT, or eHealth, are deployed worldwide. The American initiative Sequoia[1] and the European Health Digital Service Infrastructure (DSI)[2] witness how healthcare data is positively enhancing the way the patients are treated. Patients of developing and developed countries have their health records digitalised in an electronic form: the Electronic Health Record (EHR). Interoperability initiatives such as Integrating the Healthcare Enterprise (IHE)[3], or Health Level 7 (HL7)[4] enable worldwide access to medical data: records are accessible instantly from different healthcare practices. Such remote access also facilitates scientific research and improvement of public health.

In this context, *assessing the provenance of data is crucial*: the importance of data, its origins and quality have long been recognised in clinical research [2]. Creating trust relationships among the various data controllers and data processors is vital—e.g., the evidence-based medicine and the healthcare-related decisions using third-party data. However, although data exchange services have reached a high maturity level, data is still semantically non-harmonised amongst the healthcare communities. Indeed, such data is stored in various formats, e.g., XML files named Clinical Documents Architecture

[1] http://sequoiaproject.org
[2] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eHealth
[3] http://www.ihe.net
[4] http://hl7.org

(CDAs), PDF, or DICOM images. CDA data is either in prosa (Level 1), mixed prosa/-coded values (Level 2) and only coded values (Level 3). As such prosa is not necessarily ASCII, but also other data types, CDAs use different formats and templates—e.g., Continuity of Care Documents (CCD) whose values are defined by ontologies and vocabularies like SNOMED-CT and ICD-10.

CDAs are the result of multiple transformations: translation and transcoding from a language or vocabulary to another, data merge from various sources, or documents created on the fly based on other health evidences (e.g., laboratory results coming from external facilities). For this reason, international standards such as IHE or HL7 define content-agnostic data storage systems, like, e.g, IHE XDS (see Section **??**). However,

> *no international standard defines mechanisms to enforce data provenance in the CDA ecosystem*

paving the way to untrustworthy CDA data which most likely will impact the effectiveness of healthcare decisions and, most of all, patients safety.

Implementing data provenance in healthcare is cumbersome: a technical solution must address the diversity and sheer number of medical data formats currently in use! Straightforward solutions concerning the addition of inline provenance contents to the data (i.e. XML attributes or enveloped / enveloping signatures) would not solve the problem: such content can be accidentally or intentionally lost or corrupted during data manipulation. At the same time, enforcing local policies on provenance data (e.g. requiring valid electronic signatures before processing data) may open the system to frauds [9]: connection and mutual trust of distributed policy enforcement systems require organizational efforts in establishing bilateral or framework agreements, which are unfeasible in practice.

To overcome these issues, our *data provenance solution for healthcare* features

- *data-agnostic* provenance evidence to be created and processed on the fly; and
- *decentralisation* of provenance data storage and computation to prevent by-design loss and corruption of data.

This proposal seeks to solve the problem of data provenance in healthcare by building on the decentralization trait given by blockchain systems, and on well-established standards such as W3C PROV [7], HL7 CDA and FHIR, and various IHE Integration Profiles.

Proposal structure. Section 2 shows the current state-of-the-art of data provenance. Section 3 introduces key concepts on healthcare IT systems and background on the Hyperledger Fabric framework. Section 4 reports the proposed architecture. Section 5 concludes and presents related works.

2 Current data provenance solutions in healthcare

Healthcare interoperability is a key point: products must obey to international standards. Interoperability helps policy makers and project coordinators in defining long term strategies by providing software sustainability and securing the investments. Moreover, interoperability enforces security and patient safety: the quality of the patient healthcare treatment is not depending on the quality of a specific software solution (the so-called

vendor lock-in effect). Using international standards forces vendors to comply with the state-of-the-art of the security measures.

In such a context, we present the state-of-the-art of data provenance in healthcare. To this aim, we use the provenance definition given by the Office of National Coordinator

Definition *Provenance is defined as attributes about the origin of health information at the time it is first created and tracks the uses and permutations of the health information over its lifecycle*

In both the Sequoia and eHealth DSI initiatives, data provenance is enforced by using audit trails and digital signatures[5]. Audit trails assist in detection of attempted or actual security breaches, by recording on a log file details of relevant events[6]. As audit and log data are currently communicated over unreliable or untrusted messaging systems, by design, audit trails do not realise a "*chain of custody*" of healthcare data.

Digital signatures of the IHE DSG profile allows *detached* advanced and qualified electronic signatures on one or more healthcare documents. When a document consumer manipulates documents, it evaluates their associated signatures to check and enforce document integrity. However, authenticity must be proved: it is required a trust relationship established ex-ante between the author and the user of the document (such relationship is, e.g., a Public Key Infrastructure). Although technically feasible, the trust relationship comes with several scalability and organizational challenges: continuous auditing procedures to re-enforce the trust, requirements of ISO-27000-like perimeter security, etc. To achieve continental-wide trust relationships, the European Commission proposed the eIDAS regulation[7], which settles both the legal and technical requirements that a service provider must comply with to be considered trusted.

Non-repudiation is another security mechanism used to achieve provenance. Non-repudiation protocols guarantee to the participants that for each exchange message is possible to reconstruct a full custody chain (and thus, data provenance). ISO-13888 settles the definition of non-repudiation, as a set of services mandated to generate, collect, maintain, make available and validate evidence concerning a claimed event or action in order to resolve disputes about the occurrence or nonoccurrence of the event or action. Non-repudiation mechanisms provide protocols for the exchange of non-repudiation tokens, specific for non-repudiation service. These tokens shall be stored as non-repudiation information that may be used subsequently in case of disputes by so-called Trusted Third Parties (TTP)[8]. Among others, non-repudiation protocols enjoy *fairness*: each party holds the expected items at the end of the exchange process [1]; specifically, we have

- *Strong fairness*: when an item exchange is completed, sender A can prove to an arbitrator that recipient B has received (or still can receive) the item, without any further need of cooperation from either B or any trusted third party. In other words, there is no need for any further proof. For instance, using a protocol only based on digital signatures of messages is not fair. In case of disputes, one party may refuse to provide its signed non repudiation token to a judge;

[5] By leveraging on the IHE ATNA and DSG profiles, respectively

[6] `http://www0.cs.ucl.ac.uk/staff/ucacwxe/lectures/ds98-99/dsee24.pdf`

[7] Regulation (EU) No 910/2014

[8] `http://wiki.ds.unipi.gr/display/ESENS/Whitepaper+-+Non+Repudiation`

- *Weak fairness*: when an item exchange is completed, A can prove that B has received (or still can receive) the item, or otherwise an affidavit can be presented to demonstrate that B misbehaved or a network failure occurred;
- *Eventually Strong fairness*: when strong fairness is ensured but with the provision that additional assumptions about the participating parties are made.

Observation If a document (or part of it) is signed when it is generated, and its signature is stored as detached document, whenever this document is accessed and such access is under a strong fair non repudiation protocol, data provenance is guaranteed.
Discussion Let d be a new document and d_s be its detached signature document, both stored in a storage system, under the association $assoc(d, d_s)$[9]. When a document consumer obtains d using a known protocol (e.g., IHE XDS, see Section **??**), it generates a *non-repudiation of origin* token stored locally and remotely. The remote storage service creates a *non-repudiation of receipt* token stored locally and remotely. In a synchronous setting, this non-repudiation protocol enjoys fairness: everyone has the evidence that there was a message m asking for d, and that d has been delivered.

Let now d be transformed into a new document d'. A new evidence is created, stored locally. In order to prove authenticity, another signature document is created as $assoc(d', d'_s)$. When a new system accesses the document, it creates and stores the same evidence. Whenever a dispute arise (e.g., a system is challenged for data provenance) an agent can recursively query all the non-repudiation storage to analyse all the evidences, being able to reconstruct the full path of a given document. This reconstruction can be done either using a trusted third party or not (depending on the fairness level)[10].

The principled use of signatures and non-repudiation protocols can ensure data provenance for healthcare. However, achieving (strong) fairness across thousands of hospitals is impracticable: to resolve a dispute an agent should crawl among thousands of hospital evidence storage systems! Therefore, instead of using agents, we rely on a blockchain system and on state-of-the-art provenance standards. Specifically, the blockchain is used to realise a peer-to-peer network among all data stakeholders (i.e. all the intermediary operating on data for which signatures must be stored), and to achieve strong fairness as a distributed, immutable infrastructure where storing and manipulating provenance data. This blockchain solution avoids centralisation and TTP, while ensuring non-repudiation.

3 Preliminaries

Before presenting the solution guide to our pilot, we introduce the key preliminary concepts. First, we outline the main IT systems at the basis of modern eHealth, then blockchain technology and the Hyperledger Fabric framework.

3.1 eHealth building blocks

The corner stone of the an eHealth system is the data management. The most well-established solution worldwide is the *Cross-Enterprise Document Sharing* (XDS). The core model of the XDS is shown in Figure 1.

[9] An association can be imagined as a logical link between the two documents, d and d_s

[10] For a detailed description of such protocol, see `http://wiki.ds.unipi.gr/display/ESENS/PR+-+PerHopProtocol`

A *document source* generates data and submits it to a *document repository*, who is extracting metadata to update the *document registry*, who in turn cooperates with a *patient identity source* in order address the relationship health data/patient identifier, that can be subsequently consumed by a *document consumer*.

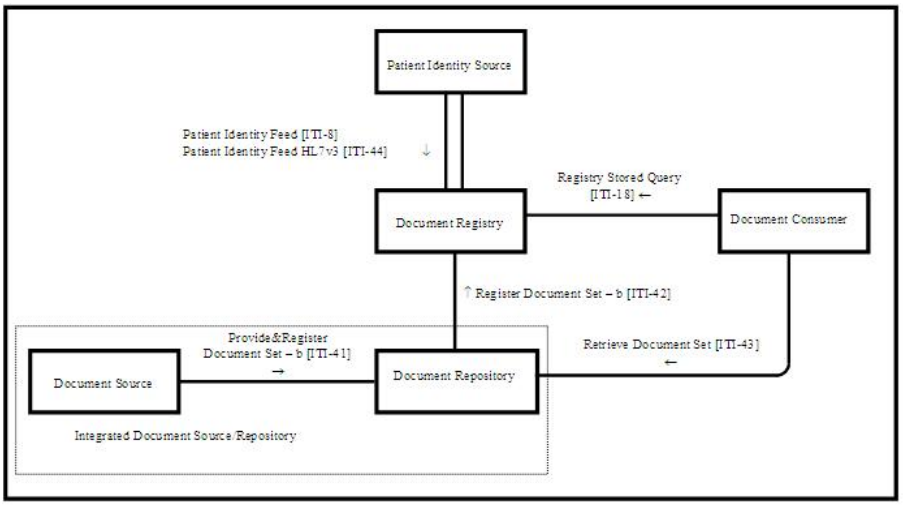

Figure 1: The Cross Enterprise Document Sharing (XDS) model

Multiple sources, repositories, and registries are allowed. The use of so-called *Affinity domain* permits harmonizing security and health semantics in order to provide basic interoperability. Affinity domains can scale to *communities* by having specific actors named *gateways* performing cross-community duties (e.g., translations, transcoding, etc).

On the other hand, the interoperability between data repository and any healthcare devices can be achieved via the HL7 standard Fast Healthcare Interoperability Resources (FHIR). FHIR combines the concepts from the former HL7 versions, and introduces the JSON and XML encoding over a set of RESTFul interfaces. IHE embraces FHIR by introducing specific profiles defining the usage of resources in the IHE stack. Notably, in its third release FHIR contains a specific provenance resource[11], which we will discuss specifically in the solution guide.

The pilot is indeed built upon XDS and FHIR for which we have used the following implementations:

- *SpiritEHR*[12], an eHealtheway-certified product[13] implementing both infrastructural (secure document exchange among healthcare facilities) and graphical services used by hospitals and professionals.

- *HAPI-FHIR*[14], an opensource HL7 Java implementation of FHIR. HAPI is used to retrieve provenance information and display in the graphical user interface for the health professional.

[11] `http://www.hl7.org/implement/standards/fhir/provenance.html`

[12] `http://www.tiani-spirit.com/spiritehr`

[13] *SpiritEHR* obtained the certification in 2011, and is listed in `http://sequoiaproject.org/ehealth-exchange/participants`, as *Inland Northwest Health Services.*

[14] `http://hapifhir.io`

The overall architecture of the pilot is reported in Section 4.2, describing the realised full-fledged ecosystem to collect, manage and check provenance of healthcare data.

3.2 Blockchain and Hyperledger

Blockchain is a novel technology that has appeared on the market in recent years. It was firstly used as a public ledger for the Bitcoin cryptocurrency [8]. It consists of consecutive chained blocks, replicated and stored by the nodes of a peer-to-peer network, where blocks are created in a distributed fashion by means of a consensus algorithm. Such algorithm, together with the use of crypto mechanisms, provides two distinguishing properties of blockchain: *decentralisation* and *democratic control of data*. This ensures that data on the chain cannot be tampered with maliciously, that operations on the chain are non-repudiable and their provenance fully tracked. All this is achieved in trust-less scenario, like the anonymous network of Bitcoin, via the consensus mechanism called Proof-of-Work (PoW). Specifically, PoW is a computational intensive hashing procedure that creates blocks with the consensus of all the network nodes. The use of PoW is indeed the key enabler of data integrity related properties of public blockchain systems.

Differently from Bitcoin, new types of blockchains such as Ethereum [15] have recently appeared featuring *smart contract*s: programs deployed and executed on blockchain. Being part of the blockchain contracts and their executions are *immutable* and *irreversible*. Smart contract permits creating so-called *decentralised applications*, i.e. applications that operate autonomously and without any control by a system entity and whose logic is immutably stored on a blockchain.

Both Bitcoin and Ethereum are public, or *permissionless*, systems whose performance (due to PoW) is really limited, but integrity and availability guarantees practically always ensured. Different deployment strategies can be followed by introducing a control on the operating users and (partially) on the context of execution. Such systems are private, or *permissioned*. This sort of blockchain ensures better performance, indeed PoW is replaced by a more effective algorithmic consensus schema. Integrity and availability are bounded by classical results of distributed systems—up to one third of malicious nodes can be tolerated in a real-world network—which have been always considered adequate in any recent modern computing systems.

Due to performance requirements, as well as privacy concerns on using a public ledger, we opted for a permissioned approach. To this aim, the proposed solution is implemented via Hyperledger Fabric[15] which, to the best of our knowledge, is the most well-established solution for permissioned blockchains. We comment on its traits and functionality below.

Hyperledger Fabric. *Fabric* is a permissioned blockchain framework implemented in Go under the Hyperledger umbrella project, and supported by the Linux Foundation.

Indeed, Fabric is used to build a permissioned blockchain: before joining the blockchain, a certain level of trust amongst participants shall be in place. Differently from other public blockchain implementations (e.g., Ethereum, Bitcoin), Fabric provides a scalable, performant, and environmentally friendly technology for highly flexible applications without any transaction fee. Most of all, Fabric follows a modular architecture

[15] `https://www.hyperledger.org/projects/fabric`

allowing blockchain core components, such as consensus algorithms, to be configured according to the needs. Smart contracts—named *chaincode* in the Fabric's jargon—are also implemented in Go and can be queried using SDK; different SDKs are available, we used the Java implementation. All these distinguishing traits makes the use of Fabric a key strength for the viability of our pilot; Section **??** will further discuss this point.

Practically, Fabric is a distributed system creating a peer-to-peer network where each peer has a replicated, consistent copy of the blockchain data structure, namely *a chained list of transaction representing invocation and executions of chaincodes*. Per se a blockchain is just a set of blocks (data structures containing signed transaction information) concatenated each other. A node named orderer ensures that blocks are distributed across all the participants with guarantees such as *atomic or total Order broadcast.*

Peers & Clients. Peers are nodes receiving ordered state updates in the form of blocks form the ordering service and maintain the state of the ledger. Peers also execute chaincode for initiating and endorsing transactions, by following a specific *endorsement policy* which defines the necessary and sufficient conditions for considering a transaction valid. Clients are entities acting on behalf of end-users. They are connected to a peer in order to communicate and operate with the blockchain.

Ordering Service. The Orderer creates the working environment of Fabric: a communication channel between peers and clients. A client must be enrolled into a channel to operate (via chaincodes) on the blockchain. Once enrolled and connected, the ordering service makes sure that client's messages are delivered to all the connected peers in the same logical order, namely implementing a *consensus* model. These messages represent the candidate transactions for inclusion in the blockchain state. Channels are similar to topics of a publisher/subscriber system; channels can be seen as a partition of the system, in which only peers enrolled can have access to the transactional state of the blockchain.

Therefore, the ability to create different channels allow participants in competition to hide transactions to others, effectively partitioning the transaction set. Notably, participants to channels are enrolled by special nodes named Membership Service Provider (MSP), which implements the permissioned layer of security of Fabric.

Blockchain state. The state of the blockchain is an abstraction of a Key/Value store, manipulated using `put` and `get` operations. Formally, the state is a mapping $K \rightarrow (V \times N)$ where K is a list of keys, V is a list of values and N is an infinite orderer set of version numbers. Thus, an update of the state is modelled as `put`(k,v), with $k \in K$ and $v \in V$, and has the effect of changing the blockchain state to $s \rightarrow s'$ such as $s'(k) = (v, \text{next}(s(k).\text{version}))$, where $s'(k') = s(k')$, $\forall k' ! = k$.

The overall interaction schema among mentioned components amounts to what depicted in Figure 2. Therefore, the following schema is followed to append a new transaction to the current blockchain.

1. A client C initiates the transaction tx containing its client identifier, the identifier of the chaincode to be executed, the payload of the transaction, the timestamp, and its signature.

2. This message is received by organization peers which simulate the transaction by enforcing and satisfying the endorsement policy of the chaincode.

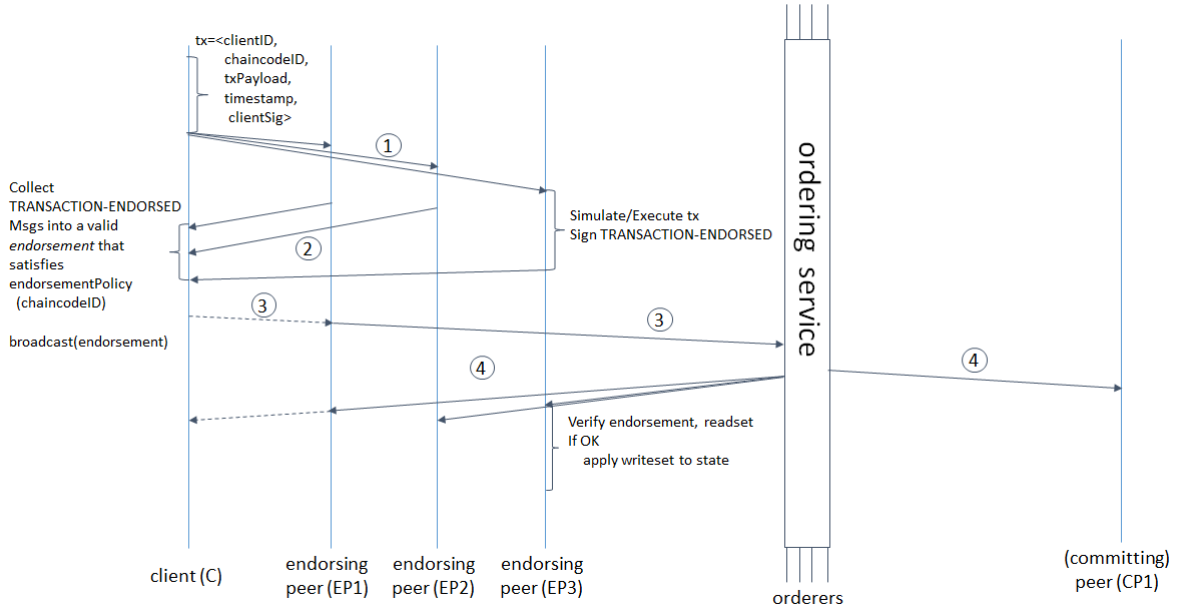

Figure 2: Hyperledger transaction flow

3. The endorsed transaction is delivered to the ordering service which then ensures that the chosen consensus schema is achieved.

4. Once order of transaction is fixed (in case of concurrent transactions) and hence a new block in created, it is then broadcasted to all peers and the new (local) state is consistently updated.

The Orderer is indeed a key element of the whole Fabric blockchain. Thanks to its modularity, Fabric ensures that the ordering services can be distributed and replicated (up to a full replication) according to the needs and the chosen consensus schema, ranging from Kafka to PBFT ones. At the same time, due to an adaptive transaction chain approach—the more transaction in input, the larger the created blocks are—Fabric ensures a throughput of thousands of transactions per second. This is clearly the main difference with respect to public blockchain systems like Bitcoin and Ethereum; we discuss in Remark 1 the ensured integrity guarantees.

From a deployment point of view, Fabric comes with Docker images[16]. Docker exploits features of the Linux kernel such as resource isolation, allowing specific deployments for peers, orderers, and chaincode. We have indeed geographically deployed a Fabric instance, developed needed chaincodes and integrated with the rest of eHealth components to build the pilot.

Remark 1 Hyperledger Fabric Integrity Guarantees. *The ordering service (together with enrolment and membership management) mainly differentiates Hyperledger Fabric with traditional public blockchain like, e.g., Bitcoin and Ethereum. Their approach is based on computational hashing power capacity, rather than an algorithm approach.*

In fact, orderers are responsible to implement what in public-like blockchains is achieved by PoW: untrusted systems (aka miners) search for a random number whose

[16] http://docker.com

hash computed with the current block is less than a "difficulty parameter" inserted in the previous block. To ensure a distributed consensus in the presence of high number of malicious actors, PoW based blockchains pay a lack of performance due to the required computing power. Indeed, both Ethereum and Bitcoin are able to process at most hundreds of transactions per seconds, throughput which can be unacceptable for eHealth-based operations. At the same, private and permissioned instances of those blockchains shall provide at any time enough hashing power to not suffer from the 51% attack [13, Section 4.2.1]: colluded malicious actors joining their forces to acquire the main part of the hashing power of the system thus causing malicious fork in the blockchain.

4 Solution Guide

Our solution for managing data provenance for healthcare documents relies on two pillars:

1. *international standards*, all IT solutions used to implement and deploy the pilot are certified and compliant with state-of-the-art standards for healthcare;

2. *decentralisation*, a principled used of a permissioned blockchain ensures non-repudiable, reliable management of data provenance according to the W3C PROV [7] standard.

The primitive data for which we track and manage provenance is the CDA. In the healthcare ecosystems, three different types can be identified: (i) XML-CDA and its sections (e.g., generic Consolidated CDAs (CCDA)[17]; HITSP C32[18]); (ii) a PDF; (iii) anything else (e.g., data content standards like X12, or DICOM, and FHIR). Our solution is compatible and interoperable with any of the previous types and indeed in our pilot we both use PDF and CCDA formats.

The core aspect of the solution is the process of signing CDAs and the subsequent creation and management of provenance information according to the W3C PROV. To ensure reliable provenance tracking across decentralised locations (i.e. hospitals and laboratories), PROV data is created, stored and distributed via a permissioned blockchain system. Figure 3 reports a high-level description of the approach.

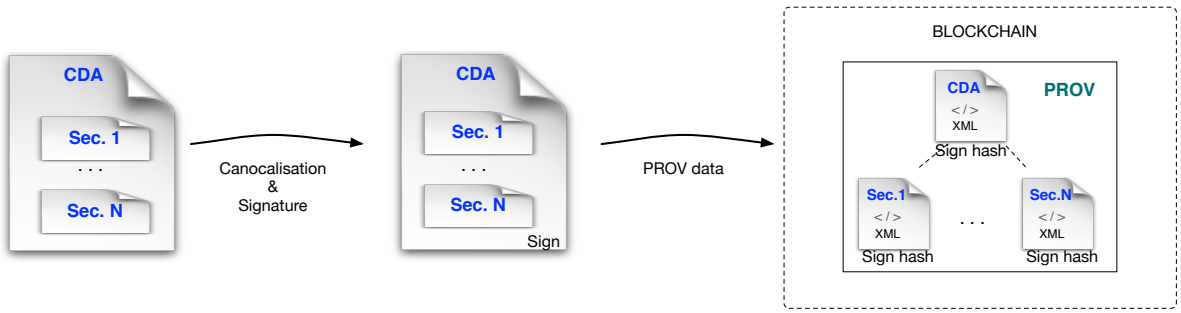

Figure 3: Overall provenance solution: example for CDA

First of all, a CDA is canonicalised and signed by using the technique corresponding

[17] http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7

[18] http://www.hitsp.org/ConstructSet_Details.aspx?&PrefixAlpha=4&PrefixNumeric=32

to its type; respectively, XaDES[19], PaDES[20], or CaDES[21]. Then, starting from the canonicalised documents, the PROV documents are created by a smart contract deployed on blockchain. The created provenance is then stored on the blockchain together with the hash outcome of the signing carried out with the corresponding *aDES procedure.

Therefore, blockchain enables the building of secure and distributed infrastructure to underpin a *decentralised* and *immutable* management of data provenance. Being the Hyperledger blockchain based on a key value store (see Section 3), provenance data corresponds to a couple

$$\langle K, V \rangle := \langle h(d), P_{\mathrm{d}} \rangle$$

where d is a CDA entity—either the overall CDA or one of its section, details below—h is the hash function as outcome of the *aDES over the document, and P_{d} is the information defined as PROV document. The PROV data is created according to a two-layer approach: a PROV document is first created for the root CDA document, and each of the contained (if any, hence only for CCDA) has its own PROV document which is linked to its root CDA.

Data Confidentiality. To manage provenance in healthcare, a key aspect to address is the management of confidential information part. The use of blockchain may open up issues of sensitive data leakage as PROV templates are stored on it. However, the signature process and the definition of the PROV template (see Section 4.2.2) ensure that no sensitive patient information is revelled. Some information needed in the PROV template (e.g. location of specific analysis) may still lead to information leaks. Thus, we assume that the distribution of the provenance of a document can be at the discretion of the patient. The implemented pilot could easily support such variant by simply tuning the input parameters of the smart contract creating the PROV documents.

Interoperability. The proposed solution supports any type of CDA and most of all by relying on FHIR it permits to automatically generate data from end-device to feed into CDA automatically. On the other hand, any end-application can retrieve and check data provenance of a document: it relies on RESTful API returning the pair $\langle h(d), P_{\mathrm{d}} \rangle$ of a given document d and then via the corresponding *aDES function checks the integrity of the document via its signature $h(d)$.

Presentation of the solution. In the following, we further present the proposed solution according to the following topics:

- Section 4.1 reports on the use case motivating the solution and implemented in the pilot;
- Section 4.2 details all the technical details regarding the definition of PROV templates, their subsequent management of them via a chaincode of Hyperledger Fabric, and integration with FHIR.

[19] https://www.w3.org/TR/XAdES/

[20] http://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf

[21] http://www.etsi.org/deliver/etsi_ts/101700_101799/101733/01.07.04_60/ts_101733v010704p.pdf

4.1 A Provenance Healthcare Use Case

The solution architecture reflects and implements the scenario described in [6, Section 4]. The user story is as follows.

> Patient *James* is feeling tired, and thus he decides to have a healthcare encounter with his General Practitioner, *Anna*. After the visit, Anna decides to further investigate for cardio-related diseases. She then forwards James to have two subsequent laboratory analysis, performed by *Lab1* and *Lab2* respectively. After the analysis, the patient is notified that results are ready, and thus James goes again to visit Anna who decides, given the results, to have a consultation with a specialist, *Bob*, a Cardiologist working in the Central Hospital. Anna prepares a patient summary for James (containing the laboratory results) and makes it available for the visit that will be held by Bob.
>
> When Bob encounters James, he checks the patient summary prepared by Anna and observes the results from the two laboratories. In order to decide, e.g., if those results are accurate enough, he checks their provenance, which confirms that effectively *Lab1* and *Lab2* performed the analysis. Given the provenance guarantees of the results and the trust of Bob in both laboratories, the analysis are considered adequate to produce medical reports and prescriptions.

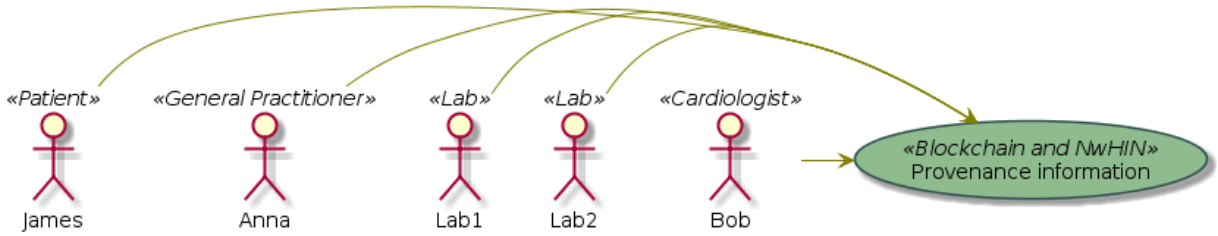

Figure 4: UML of the use case

The presented user story assumes that all the involved medical actors (i.e., the specialists and the laboratories) share data using a standard-based healthcare information exchange, e.g., the NwHIN[22]. Therefore, by relying on the our blockchain-empowered solution for ensuring data provenance, specialists can be supported towards *a trustworthy medical decision process*. Intuitively, the actors are shown in the Figure 4 by emphasising the key role of blockchain *enabling immutable and reliable data provenance tracking across geographically distributed entities.*

4.2 Solution Architecture

The architecture of the solution integrates state-of-the-art IT solutions for eHealth and the Hyperledger Fabric blockchain. Figure 5 graphically depicts the overall architecture and its components.

[22] https://www.healthit.gov/policy-researchers-implementers/nationwide-health-information-network-nwhin

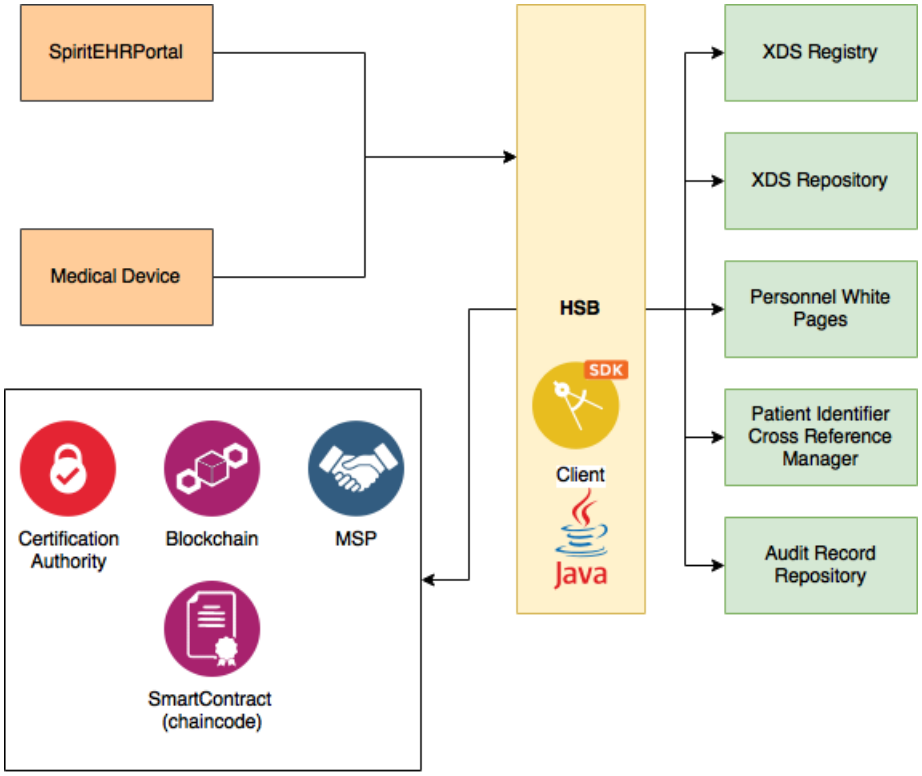

Figure 5: Architecture of the solution

SpiritEHR is the backbone of the architecture. Thanks to *SpiritEHRPortal* and *SpiritHealthServiceBus, (HSB)* solutions, SpiritEHR, together with the IHE XDS/X-CA/ATNA mandatory actors for the secure storage of medical documents, offers a certified eHealth infrastructure. The Fabric blockchain is then integrated, via its Java SDK, with HSB so to offer provenance functionality to any managed data. Most of all, to ensure adequate authentication of users acting on the blockchain, the certification authority of Fabric is integrated with that used for the XDS system.

In the following, first, we present the low-level infrastructure underlying the pilot is presented. Then, we introduce the functional components implementing provenance functionality.

4.2.1 Pilot infrastructure

The low-level infrastructure of the pilot models the use case scenario presented in Section 4.1.

Blockchain The network is composed by four virtual machines, named `blockchain1`, `blockchaingp`, `blockchainlab1`, and `blockchainlab2`, running the Healthcare Information Exchange software for the Cardiologist, the General Practicioner, and the Laboratories, respectively. A domain name service (DNS) has been installed in order on `blockchain1` to simulate as much as possible network heterogeneity. In fact, each host has been assigned to a domain name. Respectively, we have `challenge.tiani-spirit.int`, `gp.challenge.tiani-spirit.int`,

`lab1.challenge.tiani-spirit.int`, and `lab2.challenge.tiani-spirit.int`. The operating system is CentOS GNU/Linux.

The provided docker images have been restructured to fully comply with the use case as follows. Four organizations have been created, `OrdererOrg`, `GP`, `LAB1`, and `LAB2`. They all belong to the same consortium, `SampleConsortium`. Each organisation is allocated to the machine `blockchain1`, `blockchaingp`, `blockchainlab1`, and `blockchainlab2`, respectively. Each organization uses its own MSP to enrol peers and clients. Hyperledger tools created key pairs for all the systems, and a NFS share has been configured to ease the deployment of new keys and certification authorities[23]. A channel has been created, named `masab`, which all the peers joined and where the chaincode named `prov` is executed.

Integrating blockchain in the eHealth ecosystem HSB acts following the façade design pattern: it intercepts all the document-related activities (e.g., create, update, delete, store, new, etc.) and executes a chain of plug-ins to achieve other functionalities like access control, patient privacy consent enforcement, etc. In our scenario, *SpiritEHRPortal* and a generic medical device create healthcare documents. They are submitted using IHE XDS [4, Section 3.41] and intercepted by the HSB, which is acting as Hyperledger client, running the Java SDK. In fact, Hyperledger enables Java users to execute operations on the chaincode (in our scenario, the get/set) using the SDK. Notably, the SDK is not limited to invoke operations, but make it possible to manage the entire lifecycle of channels and chaincodes. SDK acts as a Hyperledger user with certificate-based credentials. Requests arrive at the interceptors of the type of *Provide and Register Document set* as in Listing 1.

The MTOM/XOP attachment is obtained and it is checked for its type for canonicalization. The hash is created (sha256) (if it is an XML, before we perform a XML canonicalization and the segmentation) and the agent information are obtained from the SAML assertion. Then the Client is executed with the extracted information and the ledger updated. The mechanism is in place for MHD/FHIR document provisions.

When querying or accessing the documents, the *Retrieve Document Set* transaction is intercepted when returning from the repository. The document is extracted again from the MTOM/XOP attachment and it is canonicalized. If it is a XML, it is segmented. For each document, and for each segment (if any), the ledger is queried using the `get` method from the Client by passing the corresponding hash, using the FHIR Provenance query. In fact, the client is grouped with the FHIR Resource server. The FHIR provenance resource is returned together with the associated PROV document following the templates and mapping in Sections 4.2.2 and 4.2.4, respectively. We return both documents to allow interoperability and in order to enable provenance consumers to collect provenance documents and still be able to execute the PROV's ontology. If the document returned is a CDA, each segment is annotated with the provenance information of its ancestors. Otherwise, the provenance information is included in the SOAP Header, following the IHE approach handling the homeCommunityId in the XCDR profile.

Notably, with this architecture we intercept all the possible documents and actions belonging to a XDS system. In fact a document has a defined lifecycle in the XDS

[23]For the sake of simplicity, the usage of tools like `http://hyperledger-fabric-ca.readthedocs.io/en/latest/` have not been used. However, it is worth noticing that every certification authority which is in place in the healthcare facilities fits for the purpose.

```
<ProvideAndRegisterDocumentSetRequest xmlns="urn:ihe:iti:xds-b:2007">
    <lcm:SubmitObjectsRequest xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0">
        <rim:RegistryObjectList xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0">
            <rim:ObjectRef id="urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1"/>...
            <rim:ExtrinsicObject id="theDocument0" mimeType="text/xml"
                objectType="urn:uuid:7edca82f-054d-47f2-a032-9b2a5b5186c1">
                <rim:Slot name="languageCode">
                    <rim:ValueList>
                        <rim:Value>en-US</rim:Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="sourcePatientId">
                    <rim:ValueList><rim:Value>RED.1515^^^&amp;1.3.1000&amp;ISO</rim:Value> </rim:ValueList>
                </rim:Slot>  ...
                <rim:Slot name="sourcePatientInfo">
                    <rim:ValueList>
                        <rim:Value>PID-3|RED.15155854163759^^^&amp;1.3.6.1.4.1.21367.13.20.1000&amp;ISO</rim
                            :Value>
                        <rim:Value>PID-5|BlockchainTest5^Test5</rim:Value> <rim:Value>PID-7|20180110</rim:
                            Value>
                    </rim:ValueList>
                </rim:Slot>
                <rim:Slot name="creationTime">
                    <rim:ValueList> <rim:Value>20180110115806</rim:Value></rim:ValueList>
                </rim:Slot>
                <rim:Name>
                    <rim:LocalizedString value="BlockchainTest"/>
                </rim:Name> ...
                <rim:Classification
                    classificationScheme="urn:uuid:a09d5840-386c-46f2-b5ad-9c3699a4309d"
                    classifiedObject="theDocument0" id="1.2.40.0.13.1.1.10.10.40.1.20180110122249046.32901"
                    nodeRepresentation="CDA/IHE 1.0">
                    <rim:Name>
                        <rim:LocalizedString value="CDA/IHE 1.0"/>
                    </rim:Name>
                </rim:Classification>
                <rim:ExternalIdentifier
                    id="1.2.40.0.13.1.1.10.10.40.1.20180110122249046.32902"
                    identificationScheme="urn:uuid:58a6f841-87b3-4a3e-92fd-a8ffeff98427"
                    registryObject="theDocument0" value="REG.1HQ117BUIL^^^&amp;1.2.3.4.0&amp;ISO">
                    <rim:Name>
                        <rim:LocalizedString value="XDSDocumentEntry.patientId"/>
                    </rim:Name>
                </rim:ExternalIdentifier>
                <rim:ExternalIdentifier
                    id="1.2.40.0.13.1.1.10.10.40.1.20180110122249046.32903"
                    identificationScheme="urn:uuid:2e82c1f6-a085-4c72-9da3-8640a32e42ab"
                    registryObject="theDocument0" value="1.2.40.0.13.1.1.10.10.40.1.20180110122249046.32892"
                        >
                    <rim:Name>
                        <rim:LocalizedString value="XDSDocumentEntry.uniqueId"/>
                    </rim:Name>
                </rim:ExternalIdentifier>
            </rim:ExtrinsicObject>
        </rim:RegistryObjectList>
    </lcm:SubmitObjectsRequest>
    <nsA:Document id="theDocument0" xmlns:nsA="urn:ihe:iti:xds-b:2007">
        <nsI:Include href="cid:theDocument0"
            xmlns:nsI="http://www.w3.org/2004/08/xop/include"/>
    </nsA:Document>
</ProvideAndRegisterDocumentSetRequest>
```

Listing 1: Sample Provide and Register document set

system, with a set of countable actions (submit, replace, approve, deprecate). Every action to document which requires provenance, it is resulted in the transactions being

intercepted. The provenance document and the FHIR resource link will be included into the CDA document, so the provenance related informations are already included in the CDA (no needed of additional queries) and the FHIR link will serve for community external consumers to fetch provenance related material.

The patient could deny the creation of the PROV document for its data. As also previously described, being PROV documents without personal information, we do not further address this concept in this paper.

4.2.2 Provenance for healthcare data

Our solution to manage provenance for healthcare data is based on the W3C PROV standard. Hence, keeping on what firstly introduced in [6], we developed our PROV templates. To this aim, we followed the guidelines from [16]. For presentation's clarity, we report in Figure 6 the overall PROV approach to provenance and we recall our approach.

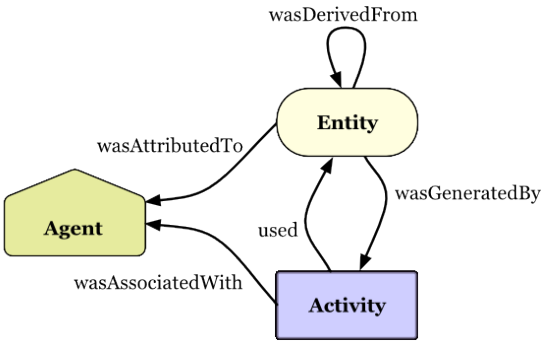

Figure 6: W3C PROV key concepts

Entity. It is a healthcare document. Provenance records can describe the provenance of entities, and an entity's provenance may refer to many other entities. In such a way, e.g., a CDA can have its provenance records, and such records linked to other records related to the derivation of the document itself.

Activity. It is how the entity comes to existence, and how their attributes change to become new entities. For instance, typical activities are: a translation, a merge, a new on demand document creations, segmentation due to access control.

Agent. It takes a role in an activity such that the agent can be assigned some degree of responsibility for the activity taking place. An agent can be a person, a piece of software, an inanimate object, an organization, or other entities that may be ascribed responsibility.

Upon the previous concepts, the self-explanatory connections in Figure 6 can be defined. We comment for each PROV template the created relationships in further details.

Provenance templates for CDA As reported in Section 4.1, healthcare data can be divided in CDA-based entities and non-CDA based data. The latter can be consider as a document formed by a single section, while in case of CDA we have an additional layer that is the internal sectioning. To this aim, we define two different PROV templates:

- *document template*, to be used for whole documents, being either PDF, DICOM or CDAs;
- *section template*, to be used for CCDA sections.

The definition of two different template ensures *high modularity and granularity of the approach*. Indeed, in case of CCDA defining a template for the whole document will not permit tracking provenance on single healthcare records part of the CDA. Practically, to track provenance of a CCDA with a single template would require updating the previous document (as the whole signature would change) critically hindering modularity and extendibility of the solution.

Document PROV template In Listing 2, is presented a sample W3C PROV XML for a generic object item (e.g., a PDF, CDA, DICOM).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <prov:document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.w3.org/ns/prov# http://www.w3.org/ns/prov.xsd"
4   xmlns:ex="urn:tiani:provenance"
5   xmlns:prov="http://www.w3.org/ns/prov#">
6      <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="theobject">
7          <prov:label>The object document</prov:label>
8          <prov:location>Nashville, TN</prov:location>
9          <prov:type>XML</prov:type>
10         <prov:value>S52fkpF2rCEArSuwqyDA9tVjawUdrkGzbNQLaa7xJfA=</prov:value>
11         <ex:locationId>urn:oid:1.2.3</ex:locationId>
12         <ex:locationName>General Hospital</ex:locationName>
13     </prov:entity>
14     <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="theobjectcreation">
15         <prov:type>ex:CREATE</prov:type>
16     </prov:activity>
17     <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="agentidentifier">
18         <prov:type>1.2.3.4</prov:type>
19         <hpd:doctorid xmlns:hpd="IHEHPD">agentidentifier</hpd:doctorid>
20         <hpd:doctorname xmlns:hpd="IHEHPD">7.8.9</hpd:doctorname>
21         <hpd:idp xmlns:hpd="idp">urn:tiani-spirit:sts</hpd:idp>
22     </prov:agent>
23     <prov:wasGeneratedBy>
24         <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobject"/>
25         <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobjectcreation"/>
26         <prov:time>2018-11-10T12:15:55.028Z</prov:time>
27     </prov:wasGeneratedBy>
28     <prov:wasAssociatedWith>
29         <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobjectcreation"/>
30         <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="agentidentifier"/>
31     </prov:wasAssociatedWith>
32     <prov:wasAttributedTo>
33         <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobject"/>
34         <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="agentidentifier"/>
35     </prov:wasAttributedTo>
36 </prov:document>
```

Listing 2: PROV document for CDA and non-CDA documents

The presented XML follows the W3C PROV core schema[24]. The lines 1-5 defines the declaration of a document in the `prov` namespace. Another namespace, `ex` has been defined in order to strictly specify additional elements introduced by the alignment with

[24] Available at: `http://www.w3.org/ns/prov-core.xsd`

IHE profiles. The document is divided as follows: it begins with the declaration of one entity, the activity that generated it, and the agent who performed the activity. Then, the relationships between those three PROV actors are defined.

Entity. The definition of the entity (lines 6-13) starts the template. The identifier (fixed value) is named "theobject", and its label (fixed value) "The object document". To give a hint on the datatype to the user, the `type` is extended as related to the XDS mime type element (e.g., XML, PDF, JPG). Indeed, a more elaborated type may be used, e.g., the classCode/typeCode elements from the XDS metadata. As last element of the entity we have the `value`, representing the hash of the element itself. The `location` element contains the physical location where the entity faced the activity, and we add two additional values `ex:locationId` and `ex:locationName` which corresponds to unique identifier of the Client performing the action.

Activity. The next element is the named activity (lines 14-16) which has only a direct child containing all the possible activities permitted by the IHE profiles: CREATE, UPDATE, DELETE, TRANSLATE, TRANSCODE, ONDEMAND (see Table 3.20.4.1.1.1-1 of [4]). Notably, being not part of the PROV namespace, they are defined in the `ex` namespace.

Agent. Next, the agent (lines 17-22) is defined, namely the principal performing the activity on the entity. The agent belongs to the actor who is bearing the IHE transaction. In fact, in all IHE messaging, the only method to formally prove a link between an identity and a community is the relationship between the user and its identity provider in the SAML assertion (see the XUA profile, [4, Section 3.40]).

This principal can be either a user or an application acting on behalf of the user. In any case, the link on how the subject (the principal) can be univocally verified is using the SAML's `SubjectConfirmation` method. These properties inherited from the SAML protocol [10] allow us to define the agent as the SAML role, Subject Identifier, XSPA Subject ID [11], and the Issuer name as `type`, `doctorid`, `doctorname`, and `idp`, respectively. Notably the `type` is defined in the PROV namespace.

Relationships. The template is then completed by the definition of the relationships. Specifically, we have

- `wasGeneratedBy` states the time when the entity has been created by the activity, at which time;
- `wasAssociatedWith` states how the object relates to the agent;
- `wasAttributedTo` states how the entity has been attributed to the agent.

Notably we do not use a time period on the activity since we believe that the considered actions are precisely identified by using a timestamp. This is the reason why the time of the action is reflected semantically in the `wasGeneratedBy`: to answer the question "when the object has been created?".

This template is used for both CDA and non-CDA documents. Specifically, as highlighted in Figure 3, the document is first signed and then the provenance info created. Non-CDA documents are hashed using SHA-256, while non-CDA documents are first digitally signed via XMLDSG [14] to provide canonicalization of the XML, then the signatureÕs digest value obtained is used (the rest of the signature element is removed as

not necessary for the following steps). Notably, that the intermediate XMLDSG step is mandatory since many XML implementations handle XML differently by adding spurious space, tabulations, or namespace that could irremediably change the hash value of the signature.

Section PROV template Here, we define a PROV template for fine-grained management of section provenance. In order to define the template, we first outline the overall process of managing a CCDA, identifying its sections and creating the provenance items.

First of all, given a CCDA, *data segmentation* is applied to achieve the canonical form of [6, Section 4]. The obtained document is a normalized CDA document $doc \Rightarrow (d, d', d'', \ldots)$ a document whose parts (d) requiring provenance are selected using the DS4P framework (e.g. a user XACML policy). The signature is then calculated by applying XML digital signature to the full document and to each of its sections. We briefly comment the PROV document for the section reported in Listing 3.

Differently from Listing 2, Listings 3 presents two entities (lines 6-21): the *master document* and the *segment* to which this provenance document refers to. The reason of having two separate entities is to provide the provenance "chain": when a section is used to create another CDA, we guarantee the possibility to locate the corresponding provenance document for the originating CDA.

The parts related to activity, agent, and relations `wasGeneratedBy` and `wasAssociatedWith` (lines 22-39) are the same as Listing 2. Instead, this document features two more relations: `used` and `wasDerivedFrom`. In fact, those relations state that the activity `theobjectcreation` used the entity `theobject`, and that `thesegment`, was derived from `theobject`.

To sum up, the master document will have a provenance document similar to Listing 2, and each section will have a document similar to Listing 3. It is worth noticing that this segmentation fully respect the RIM model[25].

Remark 2 CDA sectioning - *It is worth pointing out that our implementation does not depend on the DS4P method, but any template and segmentation strategies could be plugged in or designed on demand. However, for the sake of the pilot, we consider DS4P the default segmentation strategy for CDA sections. Therefore, whenever a CDA document is submitted to the XDS repository (the* Provide and Register Document Set *transaction) it is intercepted as per Section 4.2.1 and segmented accordingly. On each of the generated section the provenance documents are created.*

4.2.3 Managing provenance on blockchain

The provenance documents presented before are then autonomously created and stored by a smart contract deployed on blockchain. Therefore, to Being the smart contract state a key value store (see Section 3). we have:

$$\langle K, V \rangle := \langle h(d), P_{\mathrm{d}} \rangle$$

hence where K represents the list hash of PROV entities and V the associated PROV document in XML format. To act on this state, "get" and "set" operations are defined.

[25] `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7`

```
<?xml version="1.0" encoding="UTF-8"?>
<prov:document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.w3.org/ns/prov# http://www.w3.org/ns/prov.xsd"
 xmlns:ex="urn:tiani:provenance"
 xmlns:prov="http://www.w3.org/ns/prov#">
    <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="theobject">
        <prov:label>The object document</prov:label>
        <prov:location>Nashville, TN</prov:location>
        <prov:type>XML</prov:type>
        <prov:value>S52fkpF2rCEArSuwqyDA9tVjawUdrkGzbNQLaa7xJfA=</prov:value>
        <ex:locationId>urn:oid:1.2.3</ex:locationId>
        <ex:locationName>General Hospital</ex:locationName>
    </prov:entity>
    <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="thesegment">
        <prov:label>The CDA Segment</prov:label>
        <prov:location>Nashville, TN</prov:location>
        <prov:type>XML</prov:type>
        <prov:value>E0nioxbCYD5AlzGWXDDDl0Gt5AAKv3ppKt4XMhE1rfo</prov:value>
        <ex:locationId>urn:oid:1.2.3</ex:locationId>
        <ex:locationName>General Hospital</ex:locationName>
    </prov:entity>
    <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="theobjectcreation">
        <prov:type>ex:CREATE</prov:type>
    </prov:activity>
    <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#" ns1:id="agentidentifier">
        <prov:type>1.2.3.4</prov:type>
        <hpd:doctorid xmlns:hpd="IHEHPD">agentidentifier</hpd:doctorid>
        <hpd:doctorname xmlns:hpd="IHEHPD">7.8.9</hpd:doctorname>
        <hpd:idp xmlns:hpd="idp">urn:tiani-spirit:sts</hpd:idp>
    </prov:agent>
    <prov:wasGeneratedBy>
        <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobject"/>
        <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobjectcreation"/>
        <prov:time>2018-11-10T12:15:55.028Z</prov:time>
    </prov:wasGeneratedBy>
    <prov:wasAssociatedWith>
        <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobjectcreation"/>
        <prov:agent xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="agentidentifier"/>
    </prov:wasAssociatedWith>
    <prov:used>
        <prov:activity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobjectcreation"/>
        <prov:entity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobject"/>
    </prov:used>
    <prov:wasDerivedFrom>
        <prov:generatedEntity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="thesegment"/>
        <prov:usedEntity xmlns:ns1="http://www.w3.org/ns/prov#" ns1:ref="theobject"/>
    </prov:wasDerivedFrom>
</prov:document>
```

Listing 3: PROV document for CDA sections

Notably, in case of CDA sections, we may have collision of hash (when the same section is used in multiple CDAs), thus we rely on version history to handle different PROV templates.

The operations on the state are implemented in the Go language[26] as part of a *chaincode*. Such program, executed in a versioned, separated and isolated docker instance by all the peers, has four main methods: `Init` executed when the chaincode is firstly deployed in the blockchain, `Invoke` which is executed by either the command line interface or by the Clients, and `set` and `get`, executed by the `Invoke` flow, as shown in Listing 4.

The `Invoke` expects one argument, `shim`, the stub to handle the blockchain trans-

[26] At the moment of writing Hyperledger only supports chaincode written in Go. Future support for Java and Python is foreseen.

```go
func (t *SimpleAsset) Invoke(stub shim.ChaincodeStubInterface) peer.Response {
	// Extract the function and args from the transaction proposal
	log.Printf("Invocation of the chaincode called")
	fn, args := stub.GetFunctionAndParameters()

	if args == nil {
		return shim.Error("No arguments passed")
	}
	log.Printf("Args len is %v", len(args))
	creator, erro := stub.GetCreator()
	if erro != nil {
		return shim.Error(erro.Error())
	}

	// check who's calling
	log.Printf("Transaction from %s", creator)
	// In fn I know if it is a query (get) or a update (set)
	var result string
	var err error

	if fn == "set" {
		log.Println("Obtanied a set")
		result, err = set(stub, args)
		log.Printf("Obtained result %v", result)
	} else {
		log.Println("Obtanied a get")
		resultDocument, errno := get(stub, args)
		if errno != nil {
			return shim.Error(errno.Error())
		}

		m := ReturnedMessage{string(resultDocument)}
		b, errMarshal := json.Marshal(m)
		log.Printf("The value of the marshalled %s", b)
		if errMarshal != nil {
			return shim.Error(errMarshal.Error())
		}
		return shim.Success([]byte(b))
	}
	if err != nil {
		return shim.Error(err.Error())
	}
	return shim.Success([]byte("PROCESSED_OK"))
}
```

Listing 4: Go code for the Invoke

action, and returns a object `Response`. The `shim` contains the arguments that will be processed in the blockchain as JSon-encoded structure. As example, Listings 5 contains the invocation for a transaction with id "1", to "set" information about a document with hash starting with `S52...`, with an `agent` whose role is "medical doctor", whose identifier (as National Provider Identifier[27], npi) is `npi:1.2.3.4`, named "John Doe", and identified by an identity provider named `urn:oregon-state:comm-a:tiani-spirit:sts`.

```go
	res := stub.MockInvoke("1", [][]byte{[]byte("set"), []byte("
	    S52fkpF2rCEArSuwqyDA9tVjawUdrkGzbNQLaa7xJfA="),
	[]byte("agentInfo.atype"),[]byte("Medical Doctor"),
	[]byte("agentInfo.id"),[]byte("npi:1.2.3.4"),
	[]byte("agentinfo.name"),[]byte("John Doe"),
	[]byte("agentinfo.idp"),[]byte("urn:oregon-state:comm-a:tiani-spirit:sts"),
	[]byte("action"),[]byte("ex:CREATE"),
```

[27] https://www.cms.gov/Regulations-and-Guidance/Administrative-Simplification/NationalProvIdentStand/

```
7          []byte("date"),[]byte("2018-11-10T12:15:55.028Z")})
```

Listing 5: Sample Invoke parameters

By recognizing the "set" transaction, the chaincode persists this information in the blockchain by calling the orderer and performing the necessary protocol steps to propagate the information to all peers connected to the channel. Note that if there is a collision, e.g., in the event that the same document is inserted more than one time, the world state of the blockchain is updated and versioned, thus to all the subsequent "get" operation the full history of the provenance will be returned. After validating the input, the chaincode creates the provenance XML to be persisted.

Similarly, the "get" operation expects only one input, a hash. If the hash is not found in the blockchain, an error is returned. Otherwise the state and its history (if available) is returned. If the Client receives a PROV document for a segment (as in Listing 3), it will recursively call the "get" operation until a full document (as in Listing 2) is returned. This effectively enables the "chaining" of provenance records. Responses are returned as JSon-encoded objects containing the Base64-encoded provenance XML of the related hash and its history.

The complete source code can be found in `http://github.com/mascanc`.

4.2.4 Mapping to FHIR provenance resource

The provenance resource is built to define a standardized way to exchange provenance information (either in XML or JSON) about a specific entity out of a set of actions. FHIR specifies the boundaries and the relationships about on how the resources shall be used. Here, we report how we mapped PROV templates with FHIR provenance resources.

For instance, if a resource already contain elements that represent information about how the resource was obtained, those elements should always be preferred than the provenance resource. On contrast, we always use the PROV document in our model over other information. In fact, provenance data can be embedded in CDAs[28], or in Audit Trail/Security Events. In both cases, such data is not accountable: if a CDA is not signed and thus integrity-protected, anyone can alter the provenance information. Similarly, audit trails are also not signed.

The FHIR provenance format contains several concepts that are mapped to RIM (the model behind the CDA), to PROV, and to Audits. Amongst the many, FHIR defines the *reason the activity took place, the activity that took place, the role that a provenance agent played with respect to the activity, the type of relationship between two agents, and how an entity was used in an activity.* We used the PROV mappings and we extended them to fully represent our templates (refer to Listings 2, and 3). The used mappings are shown in Table 1.

In August, 2017, IHE promoted two integration profiles addressing provenance as option, namely, *Mobile Cross-Enterprise Document Data Element Extraction (mXDE)* [3] and *Query for Existing Data for Mobile* [5]. These two profiles, together with *Mobile Access to Health Documents, MHD*[29] create the ecosystem for an IHE-based scenario that perfectly adapts to the proposed settings. In Figure 7, two actors are defined: the provenance consumer, and the document element extractor.

[28] `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=420`

[29] `https://wiki.ihe.net/index.php/Mobile_access_to_Health_Documents_(MHD)`

PROV	FHIR	Reason
Target	URL of the entity resource	Contains also the entity type/hash
Period	empty	All the actions that we consider are happening in one point in time
Recorded	wasGeneratedBy/time	Generation is the completion of production of a new entity by an activity [7]. In all the actions that are considered, this sentence holds.
Policy	empty	No particular policy has been evaluated for this pilot. However, implementations, are encouraged to use and define policy identifiers on, e.g., how to disclose this information, or depending on a specific standard, e.g., [3, 5]
Location	empty	In our model the provenance information is traveling together with the document itself. This is enforced by the intercept-at-repository concept, thus the location is not reported in this field. Moreover, the recursive calls for provenance allows to reconstruct the full chain of information
Reason	activity/type	This is the definition of the activity performed
Agent	It is constructed on the basis of the agent as follows: role is constructed as per IHE XUA [4, 3.40] as `doctorname<doctorid@idp>`, onBehalfOf not used since we rely on the same information which are already presented in the SAML assertion (if in case), relatedAgentType is empty	

Table 1: Mapping PROV templates and FHIR Provenance resource

By using the IHE grouping mechanism (in which integration profiles' requirements are "merged" together to face complex clinical use cases), the provenance consumer can be implemented as XDS Document Consumer (as in our case) or a MHD document consumer. By intercepting the MHD Provide Document Bundle transaction [4], and various MHD queries for documents, our architecture fulfils the profile's dictates, and can be integrated in such deployments. In particular, on server side, the Data Element Extractor is logically grouped as XDS document registry, or MHD document responder. Notably, QEDm actors are abstraction layers used to respect the "grouping" methodology of IHE. They do not provide any message functionality, but only content-level semantic definitions.

5 Conclusions

This proposal builds on existing international standards and innovate them using blockchain technology, which is the de-facto standard for security and immutability of records. By using a performant and secure-by-design approach, the propose solution scale up to billions of records and citizens. This scalability is proven by the existing standards (e.g., the Sequoia project, the NwHIN), and well-established blockchain applications.

Related work. Data Provenance and blockchain are not a novel field of research. In par-

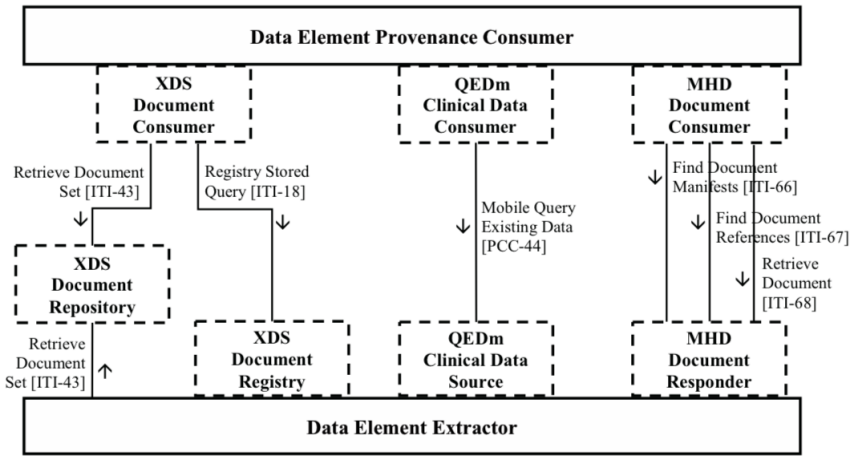

Figure 7: Figure of the IHE profiles mXDE, MHD, and QEDm

```
<Provenance xmlns="http://hl7.org/fhir">
    <id value="0%2Fi%2FSYqvJrKmuUOQCjOtBMJYy2JNqcMidJHNWIeY6fc%3D"/>
    <recorded value="2018-01-19T17:19:35.489+01:00"/>
    <reason>
        <system value="http://hl7.org/fhir/v3/DataOperation"/>
        <code value="CREATE"/>
        <display value="create"/>
    </reason>
    <agent>
        <role>
            <coding>
                <system value="http://hl7.org/fhir/provenance-entity-role"/>
                <code value="source"/>
                <display value="Source"/>
            </coding>
        </role>
        <whoReference
            id="Some Dr. accessing ELGA&amp;lt;[ GP Hospital IDP @ GP Hospital ]@urn:gda:tiani-spirit:sts>"
        />
    </agent>
</Provenance>
```

Listing 6: FHIR Provenance Resource Example

ticular, for blockchain-specific software, in [12] it is acknowledged the need for better tools and techniques for blockchain applications, effectively naming it as Blockchain-Oriented Software Engineering. The Provenance company[30] offers a blockchain-based solution to reconstruct the provenance tracks for food. Also the Hyperledger Fabric offers a solution for provenance based on storing hashes values of an asset[31]. Many other solutions are offered also in healthcare, like, e.g., Ernst & Young[32], or Ark Invest Research[33], or IBM[34]. Our proposal differs from these approaches because it is entirely based on worldwide open standards, thus avoiding the vendor lock-in effect and guaranteeing interoperability.

[30] http://www.provenance.org

[31] https://www.hyperledger.org/community/projects/sawtooth/info

[32] http://www.ey.com/Publication/vwLUAssets/ey-blockchain-in-health/$FILE/ey-blockchain-in-health.pdf

[33] http://research.ark-invest.com/blockchain-and-healthcare

[34] https://www.ibm.com/blogs/watson-customer-engagement/2017/04/11/blockchain-supply-chain/

References

[1] N. Asokan. *Fairness in electronic commerce.* PhD thesis, Waterloo, 1998.

[2] Vasa Curcin, Elliot Fairweather, Roxana Danger, and Derek Corrigan. Templates as a method for implementing data provenance in decision support systems. *Journal of Biomedical Informatics*, 65:1 – 21, 2017.

[3] IHE. Mobile cross-enterprise document data element extraction (mxde). Web, July 2007.

[4] IHE. Ihe technical framework, `http://www.ihe.net/uploadedFiles/Documents/ITI/IHE_ITI_TF_Vol2b.pdf`. Webpage, 2017.

[5] IHE. Query for existing data for mobile. Web, July 2017.

[6] Massimiliano Masi, Abdallah Miladi, Andrea Margheri, Vladimiro Sassone, and Jason Rosenzweig. Using blockchain technology to achieve health data provenance, 2017.

[7] Paolo Missier, Khalid Belhajjame, and James Cheney. The w3c prov family of specifications for modelling provenance, 2013.

[8] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at `https://bitcoin.org/bitcoin.pdf`.

[9] NEPCon. Is chain of custody certification a "myth"? `http://www.nepcon.org/newsroom/chain-custody-certification-myth`. Webpage, October 2014.

[10] OASIS Security Services TC. Assertions and protocols for the OASIS security assertion markup language (SAML) v2.02, 2005. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.

[11] OASIS Web Services Security TC. Cross enterprise security and privacy authorization profile for xacml for healthcare, 2009.

[12] Simone Porru, Andrea Pinna, Michele Marchesi, and Roberto Tonelli. Blockchain-oriented software engineering: Challenges and new directions. 2017.

[13] Matthew Tegart W Kuan Hon, John Palfreyman. Distributed ledger technology and cybersecurity. Technical report, ENISA, 2016.

[14] W3C. Xml signature syntax and processing- `https://www.w3.org/TR/xmldsig-core1/`, 2013.

[15] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2017.

[16] Simon Miles Yolanda Gil. Prov model primer. Webpage, April 2013.