IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2020, accepted July 15, 2020, date of publication July 22, 2020, date of current version July 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011201

A Hierarchical Multi Blockchain for Fine Grained Access to Medical Data

VANGELIS MALAMAS[1], (Graduate Student Member, IEEE), PANAYIOTIS KOTZANIKOLAOU[1], (Member, IEEE), THOMAS K. DASAKLIS[1], AND MIKE BURMESTER[2], (Senior Member, IEEE)

[1]Department of Informatics, University of Piraeus, 18534 Pireas, Greece
[2]Department of Computer Science, Florida State University, Tallahassee, FL 32306, USA

Corresponding author: Vangelis Malamas (bagmalamas@u nipi.gr)

This work was supported by the European Union and Greek National Funds through the Operational Program Competitiveness, Entrepreneurship, and Innovation, under the call Research–Create–Innovate under Project T1EDK-01958.

ABSTRACT The health care ecosystem involves various interconnected stakeholders with different, and sometimes conflicting security and privacy needs. Sharing medical data, sometimes generated by remote medical devices, is a challenging task. Although several solutions exist in the literature covering functional requirements such as interoperability and scalability, as well as security & privacy requirements such as fine-grained access control and data privacy, balancing between them is not a trivial task as off-the-shelf solutions do not exist. On one hand, centralized cloud architectures provide scalability and interoperable access, but make strong trust assumptions. On the other, decentralized blockchain based solutions favor data privacy and independent trust management, but typically do not support dynamic changes of the underlying trust domains. To cover this gap, in this paper, we present a novel hierarchical multi expressive blockchain architecture. At the top layer, a proxy blockchain enables independently managed trust authorities to interoperate. End-users from different health care domains, such as hospitals or device manufacturers are able to access and securely exchange medical data, provided that a commonly agreed domain-wise access policy is enforced. At the bottom layer, one or more domain blockchains allow each domain (e.g. a hospital or device manufacturer) to enforce their policy and allow fine-grained access control with attribute-based encryption. This architecture is designed to provide the autonomous management of trusted medical data/devices and the transactions of mutually untrusted stakeholders, as well as an inherent forensics mechanism tailored for granular auditing. Smart contracts are used to enforce decentralized policies. Ciphertext-policy attribute based encryption (CP-ABE) is used to distribute the decryption process among end users and the system, as well as support an efficient credential revocation mechanism. We demonstrate the efficiency of the proposed architecture through a proof of concept implementation. Finally we analyse the major security and performance characteristics.

INDEX TERMS Medical data, attribute based encryption, fine-grained access control, blockchain, smart contracts, multichain, tailored forensics, distributed trust management, revocation.

I. INTRODUCTION

The health care sector has experienced a paradigm shift in recent years. Cloud computing, emerging network technologies and the broad use of innovative medical IoT devices with computing and networking functionalities, have revolutionized health care services in terms of improved operational efficiency and new services provision. A health care ecosystem is comprised of interconnected stakeholders from multiple domains such as health care providers, insurance companies, medical device manufacturers, research institutions, governmental authorities, pharmaceuticals and patients. Private data sharing among such stakeholders is a key requirement. In fact, various types of data exist within the health care ecosystem such as health, administrative, technical, statistical (research) or insurance data. For example,

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

health data may be found in the form of Electronic Health Records (EHRs) whereas technical and insurance data may refer to firmware updates and claims registries respectively.

Until recently medical systems and devices were primarily stand-alone, unconnected systems. Modern health care devices however, may be connected and continuously exchange data with other systems, offering interoperability and enhanced service provision, but at the same time may act as enablers for cybersecurity threats. Technological innovations brought forward by medical equipment manufacturers have also resulted in a plethora of customized, non-standardized product solutions that are difficult to assess in terms of security compliance. The European Commission has recently filled this security gap by specifying in the Guidelines on a Medical Devices Vigilance System [1] the cybersecurity controls and requirements to assure medical device security, and maintain an adequate level of device functionality and safety. These guidelines also clarify the methods of dissemination/publication of security incidents by manufacturers and relevant corrective actions adopted both at European and global level. However, proving vigilant use and management of an interconnected medical device is an open challenge.

Security in the health care sector remains crucial not only because security incidents may have a far-reaching impact (in terms of patients' well-being, health care revenue, or stakeholders' reputation), but because setting plausible security objectives may prove extremely difficult. For example, the very nature of the interrelationships among health care stakeholders, typically having contradicting security and privacy goals, further exacerbates the adoption of sound security approaches. This misalignment in security objectives may prove crucial since the health care sector has been a major target for information theft and security/privacy violations during the last years [2] and, as a direct consequence, has been subject to policy-oriented scrutiny and regulatory oversight (adoption of privacy-related regulations such as the General Data Protection Regulation in the European Union).

The multitude of domains and stakeholders involved in a health care ecosystem and the need for information sharing, also pose significant privacy and security challenges. At first, the stakeholders must agree on which data can be exchanged within a domain (e.g. within hospitals) or cross-domain (e.g. between medical manufacturer and hospital domains). And since assuming that all the stakeholders involved will agree on a common policy at the same time is unrealistic, technological solutions must be dynamic. They must allow for dynamic participation changes, i.e. joining (and leaving) of stakeholders, as well as for dynamic changes in the agreed policies themselves, provided that the members within a domain, or new legislation, demand policy modifications.

There is therefore a need to establish appropriate authentication, authorization and privacy preserving encryption mechanisms to ensure that the right entity can access the right patient data at the right time. Finally, since a lack of trust inherently exists, there is a need to enforce a strong and tailored forensics mechanism, so that the trust among the stakeholders will support the trust among all entities.

A. SECURITY & PRIVACY REQUIREMENTS

Interoperable data access in the health care ecosystem must provide the required functionality for the involved entities, and at the same time maintain security and privacy. Such requirements should be closely related to the needs of the health care stakeholders and the very nature of the data handled (health, administrative, technical, research or insurance related data). Clearly the security and functional requirements should be combined and context-specific, as described below.

1) DECENTRALIZED, INTEROPERABLE & ADAPTABLE TRUST MANAGEMENT

The system must allow all stakeholders to manage the trust within their domain in a distributed and self-sustained fashion. For example, it must be possible for each hospital to *internally manage* the issuing, updating or revoking of access credentials. At the same time, the system must allow *credential interoperability*, i.e. credentials issued from independent authorities should be mutually trusted, without assuming a globally trusted root authority. Finally, trust management should be *adaptable*, i.e. allow for dynamic joins and leaves of trust authorities. For example, it must be possible to add a trust authority for a new hospital within the relevant domain, provided there is consensus among the existing stakeholders, in a flexible but also secure manner.

2) FINE-GRAINED ACCESS CONTROL

The system must assure that all entities/stakeholders (patients, doctors, health care personnel, manufacturers, etc.) have *granular access* to medical data and services at a domain level (inter and cross domain) and at a role level. For example, hospitals may allow doctors from other hospitals to have access to statistical medical data; or hospitals may grant access to maintenance personnel of device manufacturers, for reading device state information. In addition, *temporal access* should be possible. For example, hospitals may need to allow temporarily full access to the medical history of a particular patient, to a doctor that is currently treating this patient in an accident & emergency department. Such temporal access should be *easily revocable*. Finally, the system must assure *access control bypassing avoidance* mechanisms.

3) DISTRIBUTED, EFFICIENT & PRIVACY PRESERVING ENCRYPTION

The system must allow each stakeholder to *independently encrypt* and store its data in a privacy preserving way, and at the same time allow for interoperable access. Crypto primitives such as Attribute Based Encryption (ABE) may be employed for this, since the data can be independently encrypted by each stakeholder, following some commonly agreed encryption policy. However encryption should not require a globally trusted ABE key authority. In addition, the system should support the *efficient decryption*, especially

for those devices that are not strong enough to perform 'expensive' crypto operations. One way to achieve this would be to distribute the decryption functionality among the end user and the system in a secure way. Finally, the system should support the *efficient revocation* of the decryption functionality, without the need to re-distribute the ABE decryption keys to all the users.

4) EXPRESSIVE AND DECENTRALIZED ACCESS POLICIES

Smart contract technologies are used to enforce decentralized policies, and allow mutually untrusted stakeholders to manage medical data in an expressive and autonomous way. Colluding stakeholders cannot affect the execution of smart contracts. In the event of disagreement on the context of a transaction (e.g., a disputed medical charge), a consensus mechanism is used (typically, a Proof of Stake (PoS)).

5) TAILORED FORENSICS

The system should provide a strong *forensics-by-design* mechanism, able to provide *transaction integrity*, i.e. all actions (data access or even requests for data access) should be globally traced. This requirement, besides the demand for accountability, is also critical for proving medical device vigilance [1]. For example, the timeline of device maintenance transactions or for instantly reporting critical security vulnerabilities will be traceable. Finally, the forensics mechanism should be able to *withstand collisions*, even of a significant portion of compromised stakeholders.

B. MOTIVATION

Several existing solutions rely on cloud-based architectures (e.g., [3], [4]), although during the last couple of years blockchain-based architectures have also been proposed (e.g., [5]). Cloud-based solutions provide sufficient interoperability and may support fine-grained access. However, data privacy and data access transaction integrity, usually rely on strong trust assumptions on the cloud operator. Possible solutions rely on privacy preserving encryption, but since fully homomorphic encryption remains impractical [6], other cryptographic primitives such as searchable encryption [7] and, Attribute-Based Encryption (ABE) [8], [9] may be used to provide solutions for particular cases. Still, trust assumptions concerning global transaction verifiability remain. On the other hand, although blockchain-based solutions may inherently support a forensics-by-design mechanism with strong integrity guarantees, they cannot support secure interoperability of data access, for mutually untrusted stakeholders. Therefore presently, the following conflicting solutions are prevalent in the literature: (a) solutions with secure interoperable access but weak privacy that make strong trust assumptions (e.g. cloud-based), and (b) privacy-preserving solutions with restricted trust assumptions, but decreased interoperability (e.g. most of the proposed blockchain-based solutions that use pre-defined queries, trapdoors, single domains etc.). Striking the right balance between secure interoperable access on data of non-trusted entities and security & privacy is not a straightforward task. This is even harder for fine-grained access in medical data, where various entities with contradicting security goals need to interoperate.

C. CONTRIBUTION

To cover this gap we propose a novel, hierarchical, multi blockchain architecture for secure interoperable access to medical data, among users originating from different domains and authorities. In this architecture, a proxy blockchain at the top layer provides a common entry point for all users. Then, domain specific blockchains allow different stakeholders (such as hospitals or device manufacturers) to implement domain or cross-domain access policies for fine-grained access control. The main contributions are:

- The use of a common entry point monitored by all stakeholders provides a tailored *forensics-by-design* mechanism allowing *global transaction verifiability* and *policy enforcement auditing*.
- The hierarchical blockchain structure enables independently managed trust authorities to issue/revoke globally acceptable credentials, both long-term and temporal, *without assuming a global root authority*.
- The system is *adaptable*, since it enables dynamic changes of the stakeholders and relevant authorities.
- Attribute Based Encryption is more efficient for end users, since the decryption is *distributed* between the user and the system.
- Finally, the distribution of the decryption functionality allows to *efficiently revoke the decryption functionality from the users*, instead of applying inefficient ABE key revocation mechanisms.

The proposed hierarchical blockchain architecture, although specifically designed for the needs of fine-grained access to medical data might be of independent interest. To the best of our knowledge, this is the first architecture that combines different blockchains operating in a hierarchical manner, which allows for scalability in dynamic changes in particular domains and at the same time for a global access point with globally controlled transaction integrity guarantees.

D. PAPER STRUCTURE

The remainder of the paper is organized as follows. In Section II we present an overview of the available blockchain-based e-Health applications. In Section III we describe the proposed hierarchical blockchain architecture and the supporting fine-grain authorization service. In Section IV we present the implementation design. In Section V we discuss the security and performance assessment of the proposed architecture. The paper ends with some concluding remarks.

II. RELATED WORK

Access control mechanisms for health care data have received significant attention during the last years, particularly due to

the very sensitive nature of health data and the adoption of privacy-related regulatory frameworks (such as the EU General Data Protection Regulation directive). Three main types of access control mechanisms to health data have been proposed in the literature so far: Role-based access control [10], [11], attribute-based access control (ABAC) [12], [13] and entity-based access control [14]. Arguably, ABAC coupled with multi-authority and/or multi-owner approaches present significant benefits in terms of data interoperability and data privacy in health care settings [13], [15], [16]. However, all the aforementioned approaches are based on cloud solutions and various security issues may arise since cloud servers are not fully trusted and may be semi-honest and curious.

To overcome the above-mentioned limitations, blockchain-based access control mechanisms for health care data have been proposed [17]–[27], or blockchains as a forensics-by-design logging mechanism for integrity and auditability [28], [29]. A limited number of studies combine multiple blockchains, with [30], [31] being the closest to our approach from an architecture point of view. Hirtan et al [30] introduce two types of blockchain, a private for storing sensitive data, and a public which stores transaction data linked with a temporal ID, but does not use privacy-preserving encryption. Furthermore, trust management is based on whitelisting trusted nodes (allowing access to sensitive information), which poses limitations on dynamically managing distributed authorities. Zhang and Lin [31] use a similar approach, including a private and a consortium blockchain for storing Patient Health Information (PHI) and records respectively. Even though this system has the benefit of controlled revocation, it fails to be fully decentralized as an external centralized authority (the *System Manager*) is assumed. Additionally trapdoors are used to create predefined queries that restrict search capabilities.

Despite the significant benefits of existing blockchain-based approaches (such as the immutability of records and the support for decentralized services), many issues remain unaddressed. For example, [30], [31] do not cover the requirements for interoperability of independently managed trust authorities and fined-grained access control with temporal access. Various works, such as [19], [20], [23], [30]–[32] do not cover aspects of fine-grained access control and revocation [21], [33]. Other studies do not take into account the contradicting needs of the various participants/stakeholders of the health care ecosystem [22], [27], [34]. Limitations may also include the fact that health-related data sets are not encrypted [30] or access to such data sets is granted only to a few stakeholders (doctors, patients) [17], [24], [29]. Last but not least, several studies rely on the important yet impractical assumption (particularly in emergencies) of patients having to provide manual approval to health personnel in order to access their data [18], [25], [26].

We next discuss blockchain-based studies that are of particular interest since they take into account health-related data sharing requirements within the health care ecosystem as described in Section 1. In [32] the authors present a blockchain-based searchable encryption framework for Electronic Health Records (EHRs). The authors apply complex logic expressions stored on the blockchain for developing a searchable EHR index. A medical data sharing scheme based on permissioned blockchains and ciphertext-based ABAC is presented in [35]. In [36] the authors present a blockchain-enabled architecture for access authorisation to medical records. The architecture provides fine-grained access to medical data and supports flexible queries from multiple authorities. An identity-based signature access control scheme with multiple entities is presented in [37]. Apart from the blockchain, the authentication scheme is based on efficient signing and verification algorithms that may resist a collusion attack. In [38], a blockchain-based data storage and sharing system is presented coupled with multi-authority ABAC and decentralized multi-authority attribute-based signatures. A secure and efficient multi-authority access control system for IoT-enabled mHealth is presented in [39]. In [40] a multi-authority ABAC scheme is introduced. Blockchain technology is used for managing the various attribute assignments, delegations as well as revocation rights. In [41] the authors present a decentralized ABAC scheme which provides efficient privacy-preserving verification of the authenticity of health care records. A multiple authority ABAC scheme is shown in [42]. The authors use a Key-policy Attribute-Based Encryption (KP-ABE) to control the access level of health and service providers and a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to control access of medical personnel to sensitive patient data. The overall architecture relies on the usage of two private blockchains. Finally, in [43] the authors present a multi-entity ABAC mechanism based on blockchain. The mechanism provides fine-grained permissions and revocation services.

It is worth noting that most of these studies rely on predefined queries and trapdoors, features that limit the versatility and flexibility in real-life settings (limited search options, its not clear how access rights are revoked etc.). In Table 1 we present a comparative assessment of the various features prevalent in the aforementioned publications and highlight the additional features of our architecture.

An interesting aspect of our architecture is its two-layer hierarchical blockchain structure. Several inter-blockchain communication mechanisms have been proposed in the literature that address the problem of direct communication between established blockchains (e.g. Bitcoin and Ethereum) [44], [45]. We also use such a mechanism, but our approach is closer to the *Notary* scheme [46], except that our model is fully autonomous: in *Notary*, a trusted individual or group is assumed.

III. A HIERARCHICAL MULTI BLOCKCHAIN ARCHITECTURE FOR E-HEalth APPLICATIONS

In order to describe the proposed architecture in a coherent way, we shall follow the guidelines proposed in ISO/IEC/IEEE 42010 [47] for system architecture description. First, some background information for the key technologies that will be used as our building blocks are briefly

TABLE 1. Comparative assessment of the various features prevalent to the relevant literature.

Relevant literature	Encryption scheme	Access revocation	Distributed trust management	Fine-grained access control	Tailored forensics	Adaptability	Data search scheme
[36]	Shamir's SSS	No	No	Yes	No	No	3 types of queries
[35]	CP-ABE	No	No	Yes	No	No	Trapdoors
[37]	MA-IBS	No	No	Yes	No	Yes	Undefined
[38]	CP-ABE and DMA-ABS	No	No	Yes	Yes [2]	No	All types of queries
[39]	MA KP-ABE	No	Yes [1]	Yes	No	Yes	All types of queries
[41]	DABE and DABS	No	No	Yes	No	No	All types of queries
[42]	KP-ABE and CP-ABE	Yes	Yes [1]	Yes	No	No	Undefined
[43]	Undefined	Yes	No	Yes	No	No	Predefined
[40]	ABE and ABAC	Yes	No	Yes	No	No	Undefined
[30]	RSA	No	No	No	No	Yes	All types of queries
[31]	PECS	Yes	No	No	No	Yes	Trapdoors
Our approach	MA CP-ABE	Yes	Yes	Yes	Yes	Yes	All types of queries

[1] Attribute Authorities independently manage their key universe. ABE allows for credential interoperability, but distributed trust management is not examined.
[2] Attribute-Based signatures offer a tool for auditing even though this tool is not tailored specifically for such applications.

described. Then, we provide a high-level description of the architecture to assist the reader to familiarize with the high level design of the underlying blockchains, the smart contracts, the security mechanisms used for data encryption and access control and the supported services and transactions. Finally, each layer of the proposed hierarchical multi blockchain architecture will be decomposed and described in detail.

A. BACKGROUND INFORMATION

Blockchain technology is among the foundational technologies of the Fourth Industrial Revolution and is expected to play a pivotal role in various domains [48]. It was first introduced as the underlying technology of the Bitcoin cryptocurrency to safeguard the validity of transactions in an untrusted distributed environment. From a practical standpoint, a blockchain is a distributed ledger that consists of a chain of blocks containing records of transactions linked in an unalterable way (using cryptographic hashes), that is peer-to-peer managed through a consensus mechanism.

Consensus mechanisms are fault-tolerant protocols that provide a non-partisan means of establishing agreement on which transactions are legitimate and should be added to the blockchain (ensuring that only the true state of the network is maintained). They guarantee not only the validity of transactions within a blockchain network but also the necessary agreement on a network's current state. Several consensus mechanisms exist so far. Among them, *Proof of Work* and *Proof of Stake* are the most common and are extensively used. Each of these has different characteristics in terms of speed, computation time, security and scalability and, therefore, choosing the appropriate mechanism highly depends on the particular blockchain network used.

There are three types of blockchain networks a) *public networks*, that are non-restrictive, permission-less distributed ledgers which may be publicly accessed by anyone who participates in the network b) *private or permissioned networks*, that are used when only selected members are participants of a blockchain network, and their consent is needed for other members to become participants of the network and c) *consortium or federated networks* that are considered partly private networks and are usually operated under the authority of a group of enterprises or organizations.

Within blockchain technology, additional functionalities may be achieved through the use of *Smart Contracts*. Smart contracts are agreements in the form of computer programs among mutually distrusting participants that may be executed in multi-peer environments, and their execution relies on the blockchain consensus mechanism. Once the predefined conditions are met, the code of a smart contract is automatically executed. Therefore, a smart contract is a self-executing code which enables computations within the blockchain; thus, it operates as a decentralized virtual machine [49].

B. HIGH-LEVEL DESCRIPTION

The goal of the proposed architecture is to develop an efficient system for managing health data throughout their medical life-cycle. In our design, we take into account the requirements of real-world medical environments and stakeholders, such as hospitals, manufacturers and insurance companies. In addition, we consider various types of users such as patients, doctors, technicians, hospital personnel, or medical devices that may need to read or write health-related data.

The system is based on a novel hierarchical multi blockchain architecture, shown in Fig. 1. At the user layer, an API allows all users (and medical devices) to interact with the system. The user API interacts with a permissioned blockchain, the *Proxy Blockchain* (PBC), acting as a request-proxy between users and services (the lower layer blockchains) that processes the requests. Besides acting as a proxy, the role of the PBC is to register users, update their trust relations with the stakeholders, and act as a global distributed transaction logger. All the functionalities of the PBC are implemented with smart contracts. The nodes of the PBC are operated by the stakeholders of the system (hospitals,

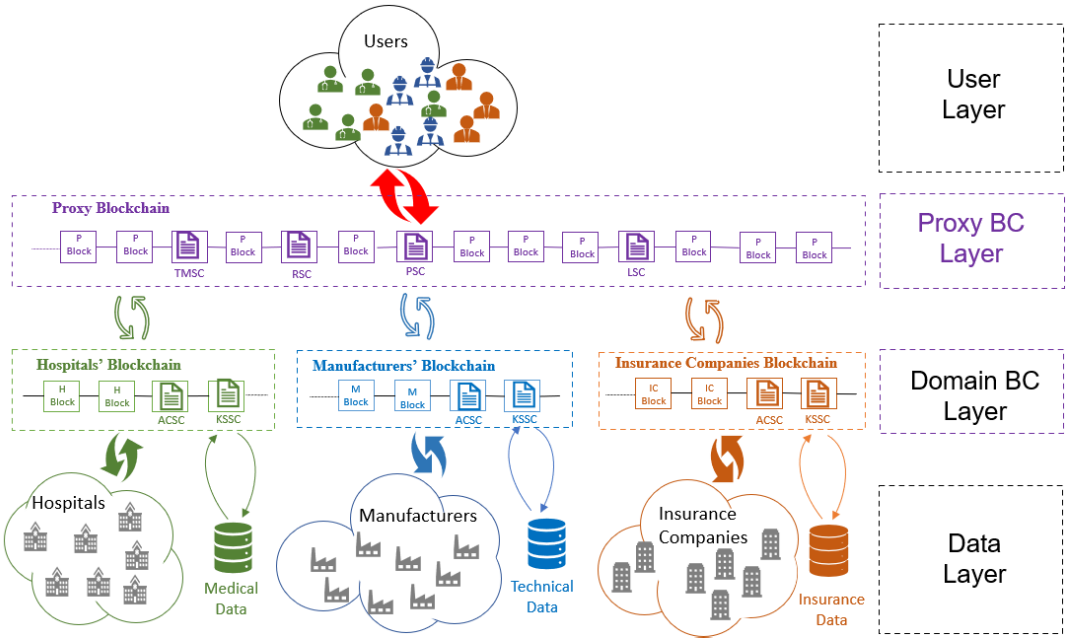

FIGURE 1. High level architecture.

device manufacturers etc) and a Proof-of-Stake mechanism is used for consensus.

The provision of the actual services is divided into multiple *domains*. Each stakeholder category forms its own domain and maintains a *Domain Blockchain* (DBC). For example the hospitals participating in this system will operate a Hospital DBC, while the medical device manufacturers will run a Manufacturer DBC. Consequently, each stakeholder acts as a node for two blockchains; the PBC and its DBC. In the proposed architecture, the DBCs are subordinate blockchains and cannot communicate directly with users. The PBC pre-processes all the requests and forwards each request to the appropriate DBC, so that the DBC can process the request and enforce the corresponding access control policy for its domain. For example, a patient's request to access their medical record maintained in a hospital's database will be handled by the Hospital DBC. Besides handling the access requests and enforcing access control, the DBC will handle the required data encryption/decryption process, by implementing an appropriate privacy-preserving encryption technology. All the functionalities are implemented through smart contracts.

The data layer is the lowest layer. Every stakeholder is allowed to maintain its data in its own database in its domain, provided the data has been appropriately encrypted. To enforce fine-grained access to the data, Attribute Based Encryption (ABE) [50] is used, before the data is stored in the appropriate database. However, each stakeholder is allowed to independently manage the ABE keys of its own database. For example a device manufacturer is able to encrypt configuration data so that: (a) only its own technical staff can access it; and (b) the hospital's technical staff can access the device software update log (to verify the vigilance of the device maintenance process). In short, users and devices can only register and interact with DBCs through the Proxy PBC. The PBC will handover a transaction initiated by a user/device to the appropriate DBC and smart contract. The PBC is maintained by all the stakeholders, while each DBC is independently maintained by the domain's stakeholders.

C. ASSUMPTIONS AND SETUP

The proposed architecture is based on the following assumptions for the deployment, data management and user management.

1) INFRASTRUCTURE DEPLOYMENT

Before system deployment, we assume that a set of stakeholders from one or more domains have agreed to collaboratively operate the infrastructure, in order to support mutual access to their data, under a predefined access policy. Each stakeholder operates one node for the Proxy Blockchain and one node for its Domain Blockchain. Since all blockchains are permissioned and the addition of new nodes is controlled, it is reasonable to assume that in all the chains each stakeholder will roughly have the same processing power. After initial deployment, the system should be able to support dynamic changes of stakeholders' participation, both at the domain and the proxy layer.

2) DISTRIBUTED DATA STORAGE

We assume that all stakeholders independently maintain their data off-chain on their own systems. Thus each stakeholder will have full control and responsibility for the proper

maintenance of their data. To support granular access to the data, each stakeholder has already encrypted their data using an Attribute Based Encryption scheme (ABE) [51]–[53].

3) ATTRIBUTE BASED ENCRYPTION KEY MANAGEMENT

In ABE a data owner is able to encrypt her data and specify access to the data as a Boolean formula over a set of attributes. There are two ABE flavors. In Key-Policy ABE the encryptor has no control over who has access to the data she encrypts, except by her choice of descriptive attributes; thus she must trust that the key issuer grants or denies access correctly to the appropriate users. In contrast with Ciphertext-Policy ABE (CP-ABE), attributes are used to describe users' credentials and the encryptor can determine a policy for who can decrypt. For this reason we will assume a CP-ABE encryption scheme.

In CP-ABE, first a master public key PK and secret key SK are generated for each authority. Data encryption takes as input the data D, the public key PK and an access structure $\mathcal{A}$ over the universe of attributes, to produce the ciphertext $C = \text{Encr}(D, \mathcal{A}, \text{PK})$. Keys are generated for (sub)sets of attributes S and are distributed to users. A key generation algorithm takes as input the master secret key SK and a set of attributes S for the access structure $\mathcal{A}$ and outputs a secret key for this attribute set, $\text{K}_\text{S} = \text{KeyGen}(\text{SK}, S)$. The authority will distribute K_S to all users with attributes S. Only users with attributes S of the access structure $\mathcal{A}$ are able to decrypt C. The decryption algorithm takes as input a ciphertext C that contains attributes S, the public key PK, the secret key K_S for the attributes S, and outputs $D = \text{Decr}(C, \text{PK}, \text{K}_\text{S})$, only if the attributes of S belong to $\mathcal{A}$.

In our system, we chose the Lewko-Waters' Multi-Authority CP-ABE (MA-CP-ABE) scheme [52], since it has many desired properties for our application. The main goal of MA CP-ABE is to allow multiple authorities to generate ABE key hierarchies, and at the same time allow users to combine attribute keys issued to them by different authorities. MA CP-ABE requires the existence of global identities, say GID, for the users.[1] The Lewko-Waters MA-CP-ABE scheme prevents collusion of attributes issued to *different* users, by linking every user's private keys with their global identifier GID, to include a distinct noise. Combining keys issued to one user (i.e. with the same GID) by different authorities will be possible since the noise is cancelled out; however the noise will not be cancelled for combinations of keys of different users. In short, MA CP-ABE is comprised of well-defined algorithms for global setup, authority setup, encryption, key generation and decryption, as shown in Fig. 2. For a detailed analysis we refer to the original work of [52].

4) INTER-/Intra- DOMAIN ACCESS POLICY

As observed above, encrypting data with MA-CP-ABE encryption requires that the authorities have a commonly agreed access policy, both inside a domain (e.g. access between hospitals) and among domains (e.g. between hospitals and device manufacturers). We assume that such a policy has been agreed among the stakeholders, taking into consideration all legal and regulatory privacy constraints [1], [54].

For each domain, the stakeholders must agree on a domain-wise list of roles and the relevant attributes that will be assigned to each role. For example in a hospital domain, the role 'Doctor' in some hospitals may allow a doctor to access Medical Health Records maintained in these hospitals' databases, but only for those patients that this particular doctor has treated in the past. Besides the roles that are 'long-term' (e.g. Doctor), we assume that the stakeholders in each domain have agreed on some *temporal roles* that may be assigned (and grant additional access) to users for short time periods. Note that the use of a temporal role must be always bounded to the corresponding long-term role, so that access will not be granted to users with only temporal roles. Although the role 'Doctor' is static, the role 'Emergency Doctor' is temporal since different doctors may be on duty for emergency incidents at different times. We assume that an 'Emergency Doctor' (e.g. doctor AND on duty) is allowed to access the medical history maintained in any hospitals' database for a patient under emergency treatment. As an example, we present the following access rules as part of an agreed inter-domain policy among hospitals:

1) **IF** α.isDoctorAtHospital(X) **AND** α.isTreating(β) **THEN** $\left(\text{X.allow}(\alpha, \text{MHR}(\beta)), \forall\ \text{MHR}(\beta) \in \text{DB}_\text{X}\right)$
2) **IF** α.isDoctorAtHospital($*$) **AND** α.onDuty($*$) **THEN** $\left(\text{X.allow}(\alpha, \text{MHR}(\beta)), \forall\ \text{MHR}(\beta) \in \text{DB}_\text{X}\right)$

The first rule allows a doctor α at hospital X to have access to the medical record of any patient in hospital X that she is treating. In the second access rule, the use of the temporal role *onDuty* allows a doctor to gain access to all the medical records from any hospital database that are related to patients under emergency treatment. In addition, the stakeholders must agree on the intra-domain access policies. For example, manufacturers may have the role 'Maintenance-Admin' that will allow users with this role to update the configuration data for all medical devices of a particular manufacturer, for all hospitals implementing this policy.

5) DISTRIBUTED TRUST MANAGEMENT

Our system does *not* require a centralized trust authority, a typical constraint for multi-authority ABE schemes. In contrast, one of the main system goals is to support the interoperability among *independent* trust domains. We assume that each (initial) stakeholder is responsible to independently manage the ABE keys as well as the authentication keys of its users. Thus each stakeholder will operate as an ABE and as a certificate authority, to manage the encryption and authentication keys of its users, respectively. Let X be a stakeholder. We denote by (pkX, skX) the public/private authentication key pair of X (to avoid confusion we use capitals for encryption keys and lowercase for authentication keys). We also

[1] We address this requirement in our system by using a global registration mechanism at the Proxy Blockchain layer.

$\text{GlobalSetup}(\lambda) \rightarrow \text{GP}$	Input=(λ: a security parameter). Output=(GP: the global parameters)
$\text{AuthoritySetup}(\text{GP}) \rightarrow (\text{SK}_\text{X}, \text{PK}_\text{X})$	Each stakeholder X produces a master secret/public key pair for its own ABE key authority.
$\text{Encr}(D, (\mathcal{A}, \rho), \text{GP}, \{\text{PK}\}) \rightarrow C$	Input=(D: the data; $(\mathcal{A}, \rho)$: an access matrix; GP: the global parameters; $\{\text{PK}\}$: the public keys of all the ABE authorities). Output= (C: the ciphertext produced for access matrix $(\mathcal{A}, \rho)$).
$\text{KeyGen}(\text{GID}, \text{GP}, \text{i}, \text{SK}_\text{X}) \rightarrow \text{K}_{\text{i,GID}}$	Input=(GID: an identity; GP: the global parameters; i: an attribute of an ABE authority X; SK_X: the secret key of Authority X). Output= ($\text{K}_{\text{i,GID}}$: the secret key for attribute-identity pair (i, GID)).
$\text{Decr}(C, \text{GP}, \{\text{K}_{\text{i,GID}}\}) \rightarrow D$	Input=(C: the ciphertext; $\{\text{K}_{\text{i,GID}}\}$: the set of attributes satisfying the access matrix corresponding to the ciphertext). Output=(D: the data)

FIGURE 2. A high-level description of the MA-CP-ABE scheme of [52], employed in our system.

denote by $CERT_X$ and CRL_X the 'root' certificate and the signed certificate revocation list of X.

All users/devices possess public/private key pairs, which have been certified independently by the trust authorities of the system stakeholders. Let $certA_X$ be the certificate issued by stakeholder X to user A. Besides the public key, the user certificate will contain all *long-term* attributes that X has assigned to A in the form of *static* roles,[2] to be later used for requesting access to the corresponding ABE keys. Users may have obtained certificates by more than one stakeholder. In that case, when users register, a certificate update process will ensure that all credentials of a user are linked to a globally unique user identifier GID. This will enable the users to issue ABE keys corresponding to their static roles issued by different authorities, in a way that enables the combination of ABE keys issued to the same user.

For the case of temporal roles however, managing their assignment with certificates is not effective, since these are dynamically assigned to users. For this reason, we assume that each stakeholder maintains and continuously updates a signed *temporal access control list* $TempACL_X$, containing entries with temporal assignment of attributes to entities.

6) COMBINING CREDENTIALS WITH ABE KEYS

Fig. 3 presents an example for the independent trust management and ABE key assignment of two hospitals. Each hospital is responsible for issuing certificates to its users, where each certificate may include the long-term roles given to the user by the issuing authority. The only requirement for the participating hospitals is to agree on a common hospital domain policy, mapping roles to specific access attributes as discussed above. As shown in Fig. 3, user A has been certified with the role 'Doctor' by hospital X, while in hospital Y he has access only to statistical data, by using the certified role 'Researcher'. Both certificates have been updated to include the unique global identifier GID_A during the initial user registration. When doctor A is on duty for emergency incidents, hospital X will temporarily assign to A the role 'onDuty', through a signed temporal access list $TempACL_X$, to allow A access to the complete medical history of all patients that are currently in emergency treatment. Each hospital X is responsible to independently update, sign and publish $TempACL_X$. Each entry in the temporal access list will contain the global user identifier, the temporal role assigned to the user and the effective time period.

Registered users are able to request their ABE keys outside the system, independently from each ABE Key Authority. The only requirement is that the ABE key authorities of all stakeholders within a domain agree on the global ABE parameters GP. In our example, user A will receive the keys $\text{K}_{\text{GID}_\text{A},\text{Doctor(X)}}$ and $\text{K}_{\text{GID}_\text{A},\text{Researcher(Y)}}$ from X and Y respectively. In addition, if at some time doctor A is on emergency duty, he will be allowed *use* of the temporal key $\text{K}_{\text{GID}_\text{A},\text{onDuty(X)}}$ (obtainable from $TempACL_X$). Note however that the actual ABE key $\text{K}_{\text{GID}_\text{A},\text{onDuty(X)}}$ that corresponds to the temporal role is *not* given to the user, since this would imply that the user would continue to have access to this key even after the expiration of the temporal role. Instead, all temporal ABE keys are stored in an isolated environment (container) and the user is only granted access to some *function* of the key through a smart contract, as explained later in Section III-E2.

D. THE PROXY BLOCKCHAIN (PBC)

Since no central trust authority exists, the system must support the distribution and dynamic update of the trust anchors, independently managed by each stakeholder. To achieve this, the root certificates $CERT_X$, the revocation lists CRL_X and the temporal access lists $TempACL_X$ of the stakeholders are stored in the PBC. Fig. 4 describes the storage, updating and indexing of the root certificates

[2]Revoking static roles is performed at authority level, by revoking the corresponding user certificate (discussed in Section III-D1).

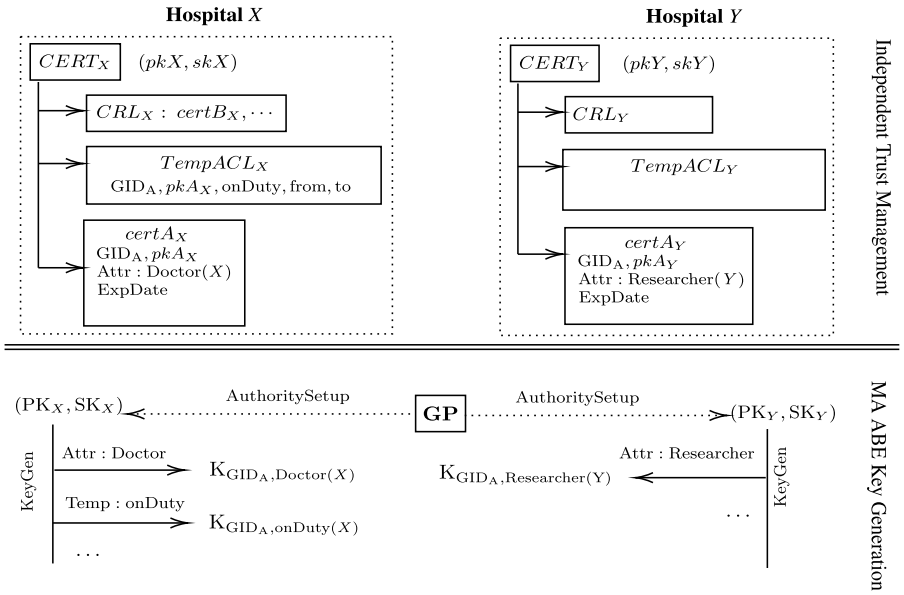

FIGURE 3. Credential issuing and ABE key generation: Each stakeholder operates a trust management authority for assigning long-term and temporal roles (upper layer) and an ABE key authority for issuing ABE keys that correspond to each role (lower layer).

(a similar process is used for the CRLs and temporal ACLs). During the initial system bootstrap, the root certificates of all stakeholders are uploaded to the PBC, say in block_i. A special counter-index Ind^0_{CERT} is also maintained. Any update, such as adding a root certificate for a new stakeholder, or refreshing/revoking an existing root certificate, must be appended to the PBC. Blocks storing root certificates of stakeholders, will also store an index to the previous block containing certificate information, to create a linked list. In a similar way, the revocation lists *CRL* and the temporal access control lists *TempACL* are maintained and updated in the PBC, as well as the corresponding indexes Ind_{CRL} and $Ind_{TempACL}$.

As pointed out in section III-B, the main goals of the PBC are to support the interoperability of users from different stakeholder domains, and act as a gateway between the users and the domain blockchains DBC. The functionality of the PBC is implemented by the following smart contracts:

1) TRUST MANAGEMENT SMART CONTRACT (TMSC)

The TMSC handles the functionality related to the dynamic update of trust anchors (involving *CERT*s, *CRL*s and *TempACL*s) and the trusted authorities as well as with the validation of the user credentials by trust anchors. The TMSC continuously tracks the blocks that contain the latest indexes Ind_{CERT}, Ind_{CRL} and Ind_{ACL} of the linked lists (in the above example this is block_l for the root certificates). The TMSC implements the following three functions:

a: UPDATE TRUST ANCHORS

This function is only accessible by users with a special role 'CA-Admin', for each stakeholder. It allows stakeholders to distribute and update their trust anchors and maintain the required indexing. Calling this function will instruct the TMSC to append the updated signed trust anchors to the PBC and if needed, to update the relevant indexes. As shown in Fig. 4, the root certificate of *X* is updated in block_j. A CA-Admin may independently update any trust anchor. Temporal access lists are expected to be updated more frequently than CRLs and root certificates.

b: ADD/REMOVE A STAKEHOLDER

This function allows current stakeholders to add new or remove existing stakeholders. Only users with the role 'CA-Admin' can access this function. Executing it is possible only if there is a consensus of stakeholders in the corresponding domain (e.g. a sufficient number of signatures by CA-Admin is received). As shown in Fig. 4, new root certificates can be added to new blocks (e.g. $CERT_Z$ in block_j) or existing ones can be revoked (e.g. the certificate of stakeholder *Y* is revoked in block_l). Again, root CA revocation is valid only if the revocation message is signed by a threshold of CA-Admin stakeholders.

c: VALIDATE USER CREDENTIALS

Any user requesting access will provide credentials, such as attribute certificates and temporal roles. For long-term attributes, this function will search the PBC, using the indexes Ind_{Cert} and Ind_{CRL} to verify that the corresponding user certificate has been signed by the appropriate authority and not been revoked (is not included in the latest CRL list of the issuing stakeholder *X*). In the same way, by using Ind_{ACL}, the function will search for the latest signed list $TempACL_X$ of the relevant stakeholder, to verify a temporal role assignment.

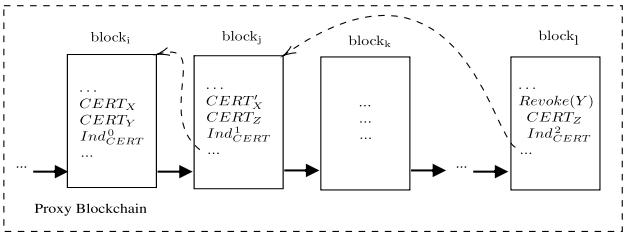

FIGURE 4. Indexing trust anchors in the Proxy Blockchain.

2) REGISTRATION SMART CONTRACT (RSC)

This smart contract handles user (or device)[3] registration. User A will send a registration request through a user API, to be handled by the RSC. This will trigger the Logging Smart Contract (discussed below) to log the request and process it. The request includes a valid user certificate $certA_X$, issued by the CA root authority of stakeholder X, along with a proof of knowledge of the corresponding private key (e.g. a signed challenge message with skA_X). The RSC will assign to this request a unique global blockchain identifier for user A, GID_A, and send it to A encrypted with A's public key pkA_X. The RSC will send to the Logging Smart Contract the newly assigned GID_A and the user's certificate(s), so that the link between these credentials and GID_A is logged on the PBC.

As an out-of-band process, user A will then request from the ABE key authority of the corresponding stakeholder X to generate and securely send the relevant ABE keys $\{\text{K}_{\text{i,GID}_\text{A}}\}$ for A's certified attributes $\{i\}$. If user A can obtain an attribute certificate from another CA authority, say Y, then A will prove possession of the global identifier GID_A, which will be included in the new certificate $certA_Y$. This will allow user A to request ABE keys from the key authority of Y with the same global identifier GID_A, and eventually combine ABE keys issued by different authorities. For user certificates that have been previously issued, a certificate refresh can be requested. As discussed above, user certificate management and ABE key management are independently processed by each stakeholder. The goal is to support the secure interoperability of such credentials.

3) PROXY SMART CONTRACT (PSC)

Registered users may request access to data only through the PSC. The PSC will verify the log-in credentials and trigger the Logging Smart Contract (discussed below) to log the request. Then the PSC will pre-process the request, in order to identify the appropriate Domain Blockchain that will process this request. When a response is received from the Domain Blockchain, the PSC will send the response to: i) the user, encrypted with the user's public key; and ii) the Logging Smart Contract to log the transaction.

[3]For simplicity we use the term 'user' both for people and devices.

4) LOGGING SMART CONTRACT (LSC)

The role of the LSC is to maintain a global transaction log that cannot be tampered, provided that a majority of stakeholders from all domains is honest. As observed above, the Logging Smart Contract is triggered any time a transaction is processed. When a user request is received, the PSC will trigger the LSC to open a transaction log. A *logging verification* variable is attached to each request, forcing the logging of every event, processed either in the PBC or in a Domain BC. To avoid maintaining open logs in the PBC until a response is received by the appropriate DBC, the Logging SC caches a request when this is received; when a transaction is complete, then the LSC will store the complete transaction details in the PBC.

E. HOSPITAL DOMAIN BLOCKCHAIN

We describe in detail the functionality of a Hospital DBC. Examples for other domains are provided in Section III-F). As explained in III-B, the stakeholders of each domain (e.g. hospitals, manufacturers, insurance companies) maintain a different DBC whose function is to provide controlled and fine-grained access to its data, according to the predefined inter-domain and intra-domain policies. This structure allows different domains to define commonly agreed, domain-wise access policies, taking into consideration business, regulatory and other constraints. For example, the access control policy for the hospital domain should be compliant with the relevant privacy regulations (e.g. GDPR or HIPAA). Through smart contracts the Hospital DBC will be able to enforce such a policy. Setting up or updating the policy within each domain (and eventually the corresponding smart contracts) will require the active involvement of the domains' stakeholders, in order to collaboratively configure the Hospital DBC nodes.

Through smart contracts, the Hospital DBC supports the following functionalities: a) enforcing fine-grained access control, by utilizing both long-term and temporal attributes from multiple CA root authorities, and b) ensuring the secure use of the ABE keys corresponding to temporal attributes and roles. The following smart contracts are implemented to provide these functions.

1) ACCESS CONTROL SMART CONTRACT (ACSC)

The ACSC enforces the predefined access policy, before granting access to encrypted data. To do this, the ACSC takes as input the verified credentials of a user (recall that the Trust Management Smart Contract has already verified both the long-term and temporal user credentials) and will approve further processing of the request, if the verified credentials are sufficient according to the access policy rules. If the policy allows the requested access, then the ACSC will send the query to the relevant database. The database will return the data, encrypted with the corresponding ABE keys.

If needed (see section bellow) the ACSC will interact with the Key Store Smart Contract to pre-process the encrypted data and will finally return the encrypted data to

the requesting Proxy SC of the PBC. In addition, the ACSC will store all transaction details in the Hospital DBC, to assure the auditability of all access requests and responses.

2) KEY STORE SMART CONTRACT (KSSC)

As pointed out in Section III-C6, ABE keys that correspond to temporal roles should not be given to the user, since this would either imply that users would be able to use these keys after the role has expired or would require to continuously update the temporal ABE keys. In our system, ABE keys corresponding to temporal roles are not given to the users (as it is the case for the long-term roles). Instead, these are generated by the ABE key authorities and securely stored in the Hospital DBCs. The Key Store Smart Contract is the only component that has access to those keys. Since temporal roles are always used in combination with the corresponding long-term roles (see Section III-C4), the decryption is performed as a two-step process, as shown in Fig. 5.

Suppose that a doctor on duty is requesting access to the medical records in all hospital databases for a patient in emergency treatment. Since the ABE access matrix includes such a policy, by combining the relevant keys, it is possible to perform the decryption. In our system, the KSSC will first perform a partial decryption by applying the key $\mathrm{K}_{\mathrm{GID_A,onDuty}(X)}$, and then will send the result to the user to finish the decryption by using the key $\mathrm{K}_{\mathrm{GID_A,Doctor}(X)}$. The use of the MA CP-ABE [52] encryption scheme allows sequential data decryption.

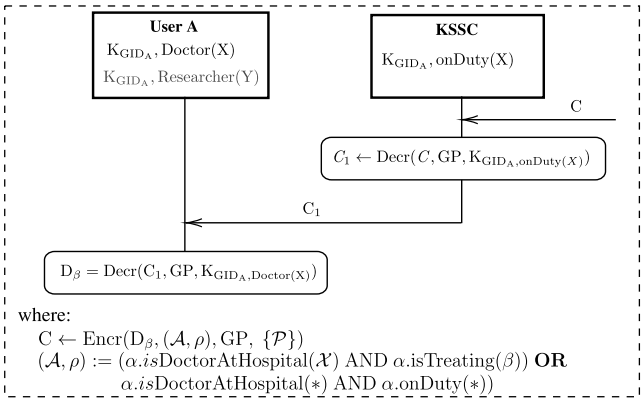

FIGURE 5. Distributing the decryption functionality among users and the KSSC.

By distributing the decryption process between the users and the KSSC when temporal roles are involved, there is no need to refresh or update/revoke the temporal ABE keys, since they are never given to the users. Finally, since the KSSC does not have full decryption capabilities (the ABE keys corresponding to the long-term attributes are not stored in the KSSC), there is no need to fully trust the KSSC in relation to the decryption of the data.

Remark: The distribution of the decryption functionality between the users and the KSSC could be extended for all accesses to data, to ensure that the users cannot bypass the system in order to access the data out of band. This could be implemented by adding a special role 'System' and then by modifying all access rules to include an AND policy with this special role. The corresponding ABE keys for the 'System' role would be created by all ABE key authorities for all their users and would be stored in the KSSC.

3) TRANSACTION FLOWS

All the services provided by the proposed system can be described as transaction flows between the users and the blockchains/smart contracts.

We describe the workflows for two main services: (a) issuing user certificates and (b) requesting access to data. A graphical representation is shown in Fig. 6. It is essential to mention that the function sequence follows a timeline e.g. issuing a user certificate and the relevant ABE keys precedes the data request access. For our example we assume that a doctor *A* from hospital *X* has been issued a temporal role *on duty* which is active within a specific time-frame. The doctor has been provided with a key $K_{GID_A,Doctor(X)}$ according to his role, granting him access to the patients he treats in hospital *X*, and also the hospital's CA-Admin has added him to the updated $TempACL_X$. When the doctor request access to data for a patient of hospital *B* the TMSC will cross-check the attributes according to the provided $certA_X$ and $TempACL_X$ and the PSC will forward the query along with the verified attributes to the ACSC of the hospital blockchain for policy enforcement. The query will be sent to the KSSC which will retrieve the two-layered encrypted data from the database. The KSSC using the Temporal keys will proceed to a partial decryption and then forwards them to the user who can then decrypt them with the Attribute-based keys he owns.

F. BLOCKCHAINS FOR OTHER DOMAINS

As in the case of the Hospital DBC, it is possible to construct DBCs to implement specific domain-wise access policies. For example, insurance companies may allow access to statistical data to other insurance companies. Based on the specified domains, it is then possible to modify the domain-specific blockchains and their smart contracts, to allow the implementation of intra-domain access policies. Some useful examples are described bellow.

1) Allow administrators of device manufacturers to update/patch medical devices installed in the premises of hospitals. For example allow manufacturers' administrators to upload signed firmware updates that will be then pulled by the devices. Since all actions will be logged in the PBC, this will allow the manufacturers to prove that they are compliant with vigilance regulations, while the hospitals will be able to prove due diligence for the maintenance of their equipment.
2) Allow insurance companies to access statistical hospital treatment data. Since all access is controlled and logged by the PBC, the insurance companies and the hospitals will be able to prove that the privacy of

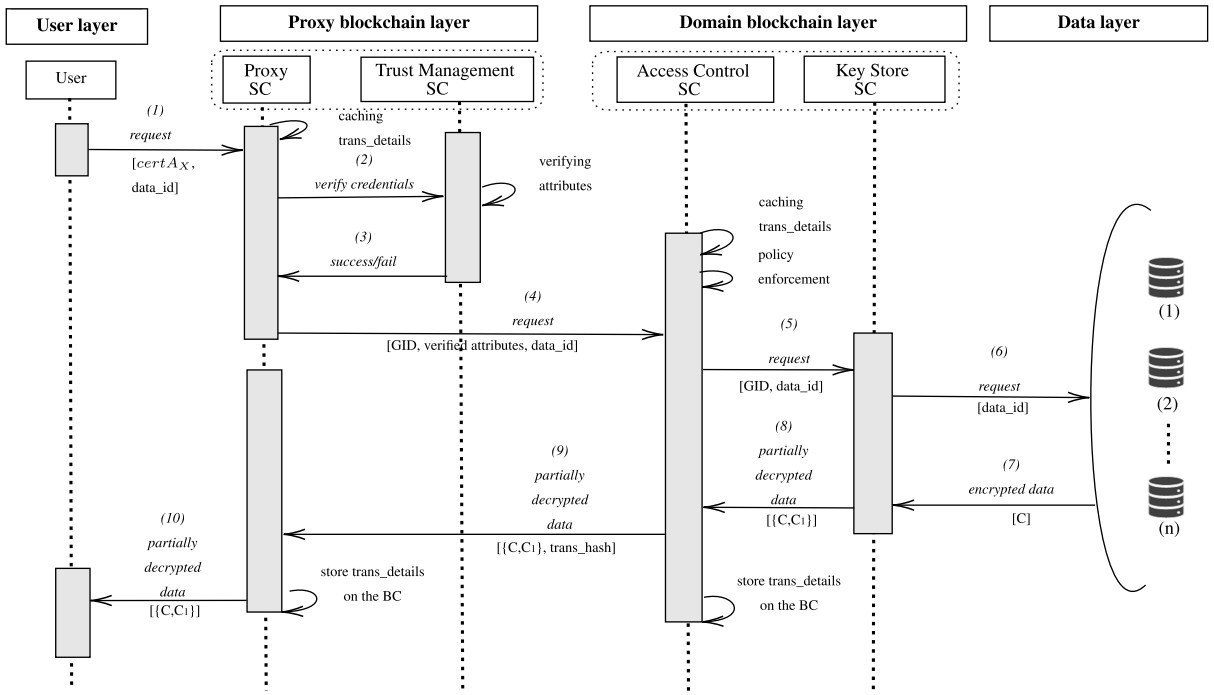

FIGURE 6. Access Control sequence diagram.

personal health records is preserved. Any modification attempt in the allowed access at a domain level (e.g. modify the relevant Access Control Smart Contract) will also be logged and traced in the Proxy Blockchain, which is controlled by all the stakeholders.

IV. IMPLEMENTATION DESIGN

Following the architecture design presented in Section III, we will analyze the most important aspects of the implementation design. In particular we provide implementation details related to: isolation and orchestration between the blockchains (Section IV-A); caching and inter-blockchain synchronization (Section IV-B); smart contract implementation (Section IV-C); and distributed ABE decryption (IV-D).

The testing environment of the proposed system, is set on a local Ethereum-based private blockchain with six Smart Contracts simulating the system functionalities. API's are used for the inter-blockchain communication and the whole infrastructure is placed inside dockers [55] in order to achieve isolation and therefore increase the security among the untrusted participants. Finally code has been developed in order to simulate the multi-authority ABE with a two-step decryption mechanism.

A. ISOLATION AND INTER-BLOCKCHAIN COMMUNICATION

The proposed system is implemented in two Ethereum-based blockchains with Proof of Stake (PoS) as the underlying mechanism to achieve consensus. Different consensus mechanisms can be used as the two blockchains are fully independent. Implementing isolation is an important security requirement, to assure that no participant can intervene during the processes of retrieving data or logging transactions in the blockchain. Such intervention could lead to unauthorized access to sensitive information or tampering logs before the storing sequence is concluded. Isolation is based on the well known *Dockers* technology that uses containers with Linux Kernel features like *namespaces* and *control groups*. With the use of namespaces, which act as a first layer of isolation, processes running within a container cannot read or alter processes running in another container or in the host system. Furthermore, a container gets its own network stack, meaning that a container does not get privileged access to the sockets or interfaces of another container. It is worth noting that from a security aspect, the use of Dockers does not solve the problem of deprecated third party libraries even though no functional problem will occur since the dependencies are already included.

Docker orchestration is based on *Kubernetes* [56]. For efficiency, Kubernetes adds redundancy to the system by enabling the dynamic use of Dockers and by balancing system resources with dynamic allocation. The proposed implementation is shown in Fig. 7. In order to control the functional components of the system (e.g the API's, Blockchains and internal Databases), the system is placed inside a Kubernetes, with the components implemented in separate Dockers. Five Dockers are used to simulate the components described in Section III. The first Docker includes an API for interacting with users (front-end) and devices (back-end), a PBC node and an Inter-blockchain API. This API acts as a global, unique entry point and performs two main actions: receiving and forwarding requests from users/devices and also balancing

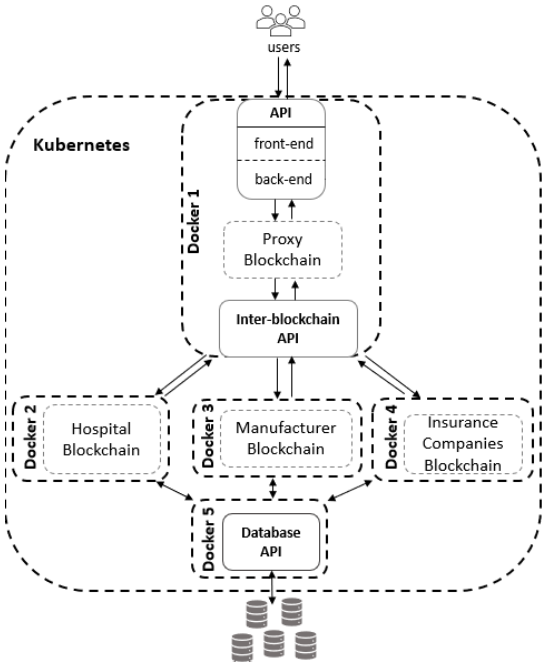

FIGURE 7. Isolation and inter-blockchain communication.

the rate at which data is forwarded to the PBC. The inter-blockchain API takes as input the output of the PBC and routes the requests to the appropriate DBC. It also works as a throttle to control the rate of data send to the DBCs and keep the flow constant.

Each stakeholder runs two nodes, one for participating in the PBC and one for the relevant DBC. The nodes of each DBC (Hospital, Manufacturer and Insurance Company) are also placed in separate Dockers. All these lower layer blockchains are connected with Docker 1 (the PBC) through the inter-Blockchain API, while they are connected with the physical databases through the Database API placed in Docker 5. It is worth mentioning that using a single database API for all the DBCs does not pose security issues due to the isolation achieved with the use of dockers.

B. CACHING AND INTRA-BC SYNCHRONIZATION

The use of multiple blockchains, besides the benefits described previously, presents several challenges from an implementation point of view. Each blockchain may use a different mechanism to achieve consensus. leading to different block validation times. This can have a serious impact in the synchronization of the interacting blockchains and could act as a bottleneck for the whole system.

In our approach the traffic between the PBC and DBCs is balanced by using the Inter-Blockchain API. This is represented in Fig. 7 by an edge gate of Docker 1, which hosts the PBC, and the lower layer DBCs. Implementing this middle API allows us to handle the exchange of data between multiple blockchains with a mechanism simulating a buffer. This buffer-like mechanism stores temporal data off-chain, but inside the Docker, waiting to be stored on-chain. The data flow includes both requests forwarded by the DBCs and responses forwarded by the PBC.

Caching and synchronization play an important role for securely implementing global logging. Recall from Section III-D4 that the Logging Smart Contract, placed in the PBC, is responsible for maintaining a global transaction log. To achieve this functionality the Logging SC is triggered from the Proxy SC every time a request is made to open a transaction session. The process remains pending until the request is completed and the response details are added to the transaction session. This procedure –if not treated properly during the implementation– may delay block publications and most certainly will cause the system to fail. A solution to this problem is presented in Fig. 8. When a request arrives from the user API to the Proxy SC, it triggers a function that sends the request details to a cache database inside Docker 1. The request is then forwarded to the Inter-Blockchain API, which in turn forwards this to the appropriate DBC (and Docker). When the response is received by the PBC, the details are sent to the cache database, which adds these to the response and forwards this for finalization to the PBC.

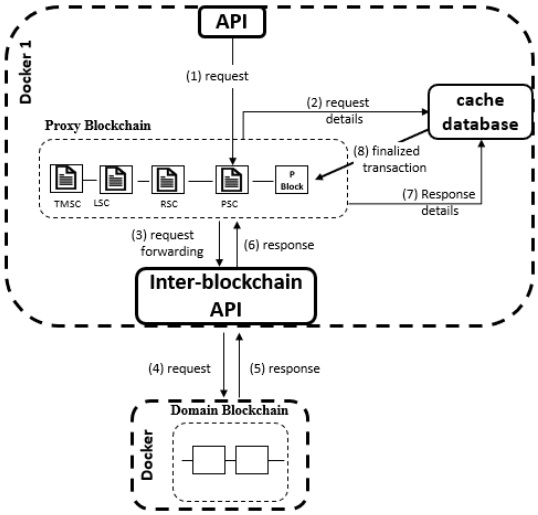

FIGURE 8. Caching transaction details.

C. SMART CONTRACTS IMPLEMENTATION

To implement the functionalities in Section III, we created an ethereum-based private blockchain using `node`[4] and `ganache-cli`,[5] and `truffle`[6] to test and deploy six functional smart contracts written in *Solidity*. These Smart Contracts simulate several system operations and take advantage of security features provided by default in the Solidity programming language, such as the *require* clause used to enforce access control in the functions and the *msg.sender*, used for account authenticity. In Table 2 and Table 3 we

[4]https://nodejs.org/
[5]https://github.com/trufflesuite/ganache-cli
[6]http://truffleframework.com

present the main functions of the PBC and DBC Smart Contracts.

As shown in Table 2 we created a *majorityConsent* function for reaching consensus among stakeholders/authorities when this is needed for specific functions, such as adding a new CA with the *addCA* function and retrieving Logs when an audit is performed with the *retrieveLog* function. Additionally we developed the *updateTrustAnchors* function accessible by the CA administrator for updating information in the Ind_{CERT}, Ind_{CRL} and Ind_{ACL} indeces for each authority.

D. ABE ENCRYPTION/ZZZZZ/WWWWW/DISTRIBUTED DECRYPTION

For encrypting data, the classic MA-CP-ABE scheme was used from the *Charm v0.50* crypto library [57] of the Python programming language. The user's Ethereum addresses are used as the Global unique ID required for MA-CP-ABE. For decryption, the scheme was transformed into a distributed two-step mechanism: first the Key Store SC performs partial decryption and then the user or device performs the final decryption over the partially decrypted data. In the partial decryption step, the function calculates the terms of the classic scheme that refer to the system attributes and keys for the ciphertext and forwards the results to the user who then calculates the rest of the terms with his own attributes and keys locally. Details related to the performance overhead are provided in Section V-B3.

V. ARCHITECTURE EVALUATION

First we analyse the security and privacy of the proposed architecture. Then, we analyse the efficiency, with an emphasis on the run-time and computational overhead of the overall decryption functionality, for varying system/user attributes.

A. SECURITY AND PRIVACY EVALUATION

We examine how the proposed architecture satisfies the desired security and privacy requirements set in Section I-A.

1) SECURITY ASSUMPTIONS

- The underlying isolation mechanism is secure, i.e. the stakeholders that operate blockchain nodes cannot bypass isolation and access/alter the blocks, and therefore the Smart Contracts running at the Proxy or Domain blockchains.
- Throughout the system operation, for each Domain BC the majority of its stakeholders are honest. Since Domain BC stakeholders also operate PBC nodes, the same assumption holds for the PBC.
- All underlying crypto primitives used (such as public key encryption, ABE encryption or hash functions) are secure.
- All relevant crypto keys that are externally managed, such as ABE keys or user credentials and certificates, have not been compromised.
- The adversary has polynomially bounded computer resources.

2) TRUST MANAGEMENT

The system provides secure interoperable trust management among independently managed credential authorities.

- *Independent and distributed trust management.* The stakeholders independently manage their users' credentials. For example a hospital may issue/revoke attribute certificates or temporal credentials for patients, doctors, on-duty doctors or medical staff. Since there is no need to store the signing keys of each stakeholder within the system, this simplifies vigilance requirements (for example for device manufacturers) [1].
- *Secure interoperability among different trust domains.* The Trust Management SC, running at the PBC, allows stakeholders to securely publish and update their trust anchors, such as CRLs and temporal credentials. This in turn allows users to combine credentials issued by different authorities. More importantly, no hard trust assumptions, such as a globally trusted PKI are required. The TMSC assures the integrity of the trust anchors.
- *Dynamic and collision resistance.* The functionality of the TMSC allows for dynamic changes of trust anchors, such as adding or removing anchors. At the same time the system is resistant to collusions of compromised stakeholders, that are needed to add or revoke new trust authorities without having their consent. Recall that such changes can be applied within each domain only if a majority of the relevant domain's stakeholders agree.

3) FINE-GRAINED ACCESS CONTROL

- *Granular policy management and enforcement.* The system provides an efficient and granular policy management mechanism. Within each domain, the stakeholders may agree on a different access policy, which can be easily implemented and enforced by the Access Control SC of the relevant domain blockchain. Cross domain access polices can also be established, by properly modifying the ACSC of the particular domains.
- *Temporal access.* The use of temporal access control lists provides an efficient mechanism to support temporal access, which can be independently managed by each stakeholder.
- *Revocable access.* Both long-term attribute certificates and temporal credentials can be efficiently and globally revoked, by updating the relevant trust anchors at the PBC (the *CRL* and *TempACL* lists).

4) PRIVACY PRESERVING ENCRYPTION

- *Independently managed encryption keys.* As in the case of trust management, the stakeholders are also able to independently issue their ABE keys and assign ABE keys to their users. Data encryption is also performed independently and outside the blockchain hierarchy.
- *Protecting access to temporal ABE keys.* The Key Store SC securely stores the ABE keys of temporal roles and partially decrypts data before forwarding it to the

TABLE 2. Proxy Blockchain SCs main functions.

Function	Smart Contract	Actuator	Input	Output	Description
constructor	All	-	-	True/ False	Initialize and store information relevant to the SC
majorityConsent	PSC	*addCA* *removeCA* *retrieveLog*	authoritySign	True/ False	Sends and receives signed challenges from the Authorities
updateTrustAnchors	TMSC	CA Admin	new Cert new CRL new TempACL	updated pointer	Updates the Trust Anchors and updates the pointer to the last object of the linked lists. Triggers the *updateLog* from the LSC
getUserValidation	TMSC	PSC	userCert	True/ False, userRole	Checks the validity of the certificate by comparing it with the Ind_{CERT}
validateUser	PSC	User	userCert	True/ False, userRole	Takes as input the given user certificate and triggers the *getUserValidation* from the TMSC
addCA	TMSC	CA Admin	majorityConsent new CA_{CERT}, new CA_{CRL}, new $CA_{TempACL}$	updated pointer	Updates the Trust Anchors and sets a new pointer to the last object of the linked lists. Triggers the *updateLog* from the LSC.
removeCA	TMSC	CA Admin	majorityConsent revocate CA_{CERT}, revocate CA_{CRL}, revocate $CA_{TempACL}$	updated pointer	Updates the Trust Anchors and sets a new pointer to the last object of the linked lists. Triggers the *updateLog* from the LSC.
userRegistration/ deviceRegistration	RSC	User / device	userCert / devCert	userAddress/ devAddress	Links a user/device certificate with a unique Ethereum address (GID). Triggers the *registrationLog* of the LSC.
requestAccess	PSC	User	userCert,data_id	userRole, data_id	First, it triggers the *requestLog* from the LSC to log the request and then calls the *getUserValidation* from the TMSC to validate the user, if the function returns true it forwards the request to the DBC.
requestLog	LSC	PSC	request details	True/ False	Triggered by the PSC when a request is made. Returns a logging verification variable
updateLog	LSC	TMSC	update details	True/ False	Triggered by the TMSC when a request is made. Returns a logging verification variable
registrationLog	LSC	RSC	registration details	True/ False	Triggered by the RSC when a request is made. Returns a logging verification variable
retrieveLog	LSC	Auditor	majorityConsent, retrieve details	Logs	When an Auditor request access to the Logs, Authorities grant access through majority voting. It is a threshold enabled function.

TABLE 3. Domain Blockchain SCs main functions.

Function	Smart Contract	Actuator	Input	Output	Description
constructor	All	-	-	True/ False	Initialize and store information relevant to the SC
policyEnf	ACSC	PSC	verified attributes, data_id	True/False	Enforces the predefined policy according to the given verified attributes
requestData	KSSC	ACSC	data_id	partially decrypted data	It is activated when temporal attributes are verified for the user. The function retrieves the data from the database and performs partial decryption with the Temporal keys before it sends them to the PSC and then to user.
accessLog	ACSC	*policyEnf*	policy enforcement details	Logs	Triggered by the ACSC when policy enforcement over a request is made. Logs the transaction details on the Domain BC.
keystoreLog	KSSC	*requestData*	partial decryption details	Logs	Triggered by the KSSC when a partial decryption request is made. Logs the partial decryption details on the Domain BC.

requesting entity. In this way, it is not possible for users to abuse temporal ABE keys (e.g. use them after a temporal role has expired), since they never get access to these keys. In addition, this removes the need to continuously update ABE keys.

- *Distribution of the decryption functionality.* The temporal ABE keys are the only private keys stored on-chain and their protection relies on the isolation mechanism. However even if these keys get compromised this will not be sufficient to grant access to data. Recall that an 'AND' encryption policy is used; the temporal ABE keys are stored in the DBC while the relevant 'long-term' ABE keys are stored at the user side.

Note that the two-step distributed decryption can be extended to cover all existing decryption policies/rules; a special purpose ABE key/role (say KSSC) is only required, which can be applied with an add-on AND rule over all existing rules.

- *Efficient ABE key revocation.* It is not easy to revoke ABE keys in an efficient way. However, by applying a global two-step encryption/decryption policy as described above (i.e. the KSSC partially decrypts data which are then forwarded to the user for final decryption), the ABE decryption functionality can be revoked efficiently from users and devices. The ABE keys of revoked users will not be sufficient to access the data off-chain, without also applying the KSSC keys. Note that this also acts as a mechanism that prevents attacks that bypass the blockchain mechanism, since accessing the data outside the PBC will not be sufficient to allow access to decrypted data.

5) TAILORED FORENSICS

- *Global transaction verifiability.* Any access request (successful or not) for data within a domain, will be logged at the PBC (through the Logging SC). Thus transactions are globally traceable from the PBC, which acts a global ledger, in addition to the transaction logs maintained in the domain level blockchains. This assures strong integrity guarantees for all services provided by the DBCs.
- *Globally detectable policy modification attempts.* Since all transactions are forwarded through the PBC, where they are irreversibly logged, including transactions that will modify access policies at the DBC level (e.g. modifying the Access Control SC of a DBC), it is not possible for a single domain of stakeholders to modify an access policy, without such changes being globally detected.
- *Collision resistance.* Since the stakeholders from all domains are nodes of the PBC, altering the logs of the PBC will not be possible, provided that a majority of stakeholders are honest.

B. PERFORMANCE EVALUATION

We evaluate the main performance features of the proposed architecture.

1) EXTENSIBILITY AND FLEXIBILITY

The proposed system is extensible and flexible with respect to the structure of domains and membership in each domain. New domains of stakeholders can be added and existing domains removed, provided that a majority of existing stakeholders agrees, by calling the *majorityConsent* function of the Trust Management SC at the PBC. Within each domain, stakeholders can be added or removed (e.g. adding a new hospital), provided there is consensus among existing members of the particular domain, *without* engaging members of other domains. The same holds for the underlying services (Smart Contracts) within each DBC.

The use of a two-layer blockchain hierarchy allows the fine-tuning of the consensus mechanisms used. In our case we use a Proof-of-Stake for both the PBC and the DBCs. Note however that if needed, the proposed layered blockchain architecture is flexible enough to support different consensus mechanisms for different blockchains as the PBC and the underlying DBCs are completely independent.

2) SCALABILITY

Since the role of the PBC is to act as a proxy and pre-process all requests, one concern is that the PBC may become a system bottleneck, as the number of users increases. Note however that since the stakeholders from all domains are required to operate nodes of the PBC, the number of PBC nodes will grow linearly with the number of stakeholders and therefore with the number of users. In addition, the layered blockchain architecture inherently supports the distribution of processing costs; the PBC pre-processes all requests, while each DBC performs the actual processing requiring more intensive operations, like ABE partial decryption, but only for a portion of requests.

3) SYSTEM PERFORMANCE

The system architecture is a two-layer blockchain hierarchy. At the top layer the PBC interacts with the DBCs and at the lower layer the DBCs interact with external databases. The overall run-time is affected by, a) the consensus algorithms applied in both the blockchains used in the system, b) the capacity of the Cache memory implemented in the PBC for buffering the transactions and to the API used for controlling inter-blockchain communication, and c) the decryption time needed from the system to perform partial decryption of the requested data. For running our experiments we used an AMD-Ryzen7-2920X CPU with 12 cores, 16Gb DDR4 RAM and 500Gb SSD hard disk drive.

We have already mentioned that the PBC and the DBCs are fully autonomous. Each can run different consensus algorithms with impacts on the overall time and also the efficiency of the system.

To measure the performance of our system, we used *Proof of Stake* as the underlying consensus protocol in both blockchains of the proposed system, achieving an average of 1000 *t/s* rate throughput and block publication time of approximately 12s. We should note that block publication time does not affect the system's efficiency due to the fact that a transaction can be finalized before a block is verified by the stakeholders since only the transaction hash is stored in the PBC. The security of the system is guaranteed by the trust management and access control mechanisms and the logging enforcement.

In the lower layer, where the computational intensive ABE decryption is used, the performance analysis focuses on testing the varying combinations of ABE attributes, both for the end-user side and the Key Store SC side. In addition, we examine the performance for a gradually increasing number of requests. The goal here is to measure the time required for partial decryption at the KSSC and also at the user side to finalize the decryption. This time is then compared with the original (single-step decryption) MA-ABE scheme. It is worth noting that data encryption is assumed to be performed outside the system by the relevant stakeholders (e.g.

hospitals) who have already encrypted their stored data with the MA-ABE scheme. For the experiments we used small size (text) files to simulate medical health records.

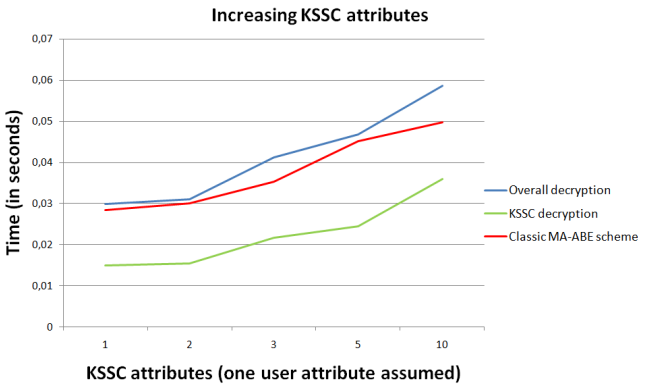

FIGURE 9. Performance analysis based on increasing KSSC attributes.

In Figures 9 and 10, we measure the decryption time needed when KSSC and user attributes increase accordingly. We have tested various combinations of KSSC and user attributes in order to calculate the effect in our system.

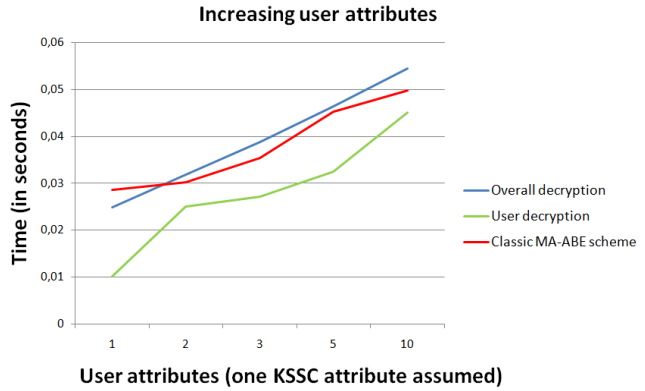

FIGURE 10. Performance analysis based on increasing user attributes.

From these findings, we conclude that there is a linear relationship (positively correlated) between the data decryption time and the increase of KSSC/user attributes. Therefore, increasing the number of attributes seems to scale proportionally with the time needed to decrypt the data.

These results are consistent with other studies [21], [33]. Compared to those studies, our approach has a slightly higher overall decryption time overhead (for both the system and the user). This is due to the fact that we use a two-step decryption function simulating the system partial decryptions. Each time a decryption request is made, an access tree policy and attribute policy is computed first for the KSSC and then for the user attributes, instead of the one-step operation of the classic MA-ABE scheme. However, the decryption time for the user is significantly lower, since part of the process is performed by the KSSC.

In addition, we tested the sensitivity of our decryption function to varying user requests. From Fig. 11 it is evident that the decryption time scales proportionally with the increase of user requests for different KSSC/user attributes scenarios.

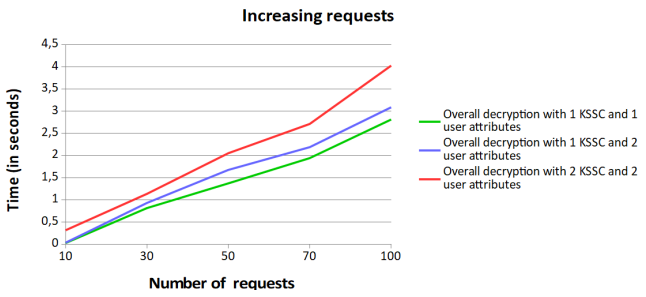

FIGURE 11. Performance analysis based on requests.

For testing the performance of our system under realistic conditions, we examined the computation burden of particular combinations of ABE user/system attribute settings, for a varying number of queries. In particular, we have assumed, on average, two ABE attributes for the KSSC (the 'system' attribute and a temporal role attribute) and two user attributes (i.e. 'doctor' and 'researcher') although these may vary based on the users' role. As shown in Fig. 11, for our test environment the overall decryption time for the above attribute setting grows from almost 1 *sec* for 30 requests to approximately 3 *sec* for 100 concurrent requests including the user side decryption times, i.e. around 30 *ms* overall decryption time per transaction. These results indicate that the system remains scalable enough to accommodate multiple user requests under different system/user attributes.

TABLE 4. Transaction execution performance for the access-to-data service.

Blockchain execution time			Cache	User	Transaction
SC	Functions	execution time	delay	execution time	Execution time
PSC	*requestAccess()* *validateUser()*	113 ms			
TMSC	*getUserValidation()*	98 ms			
LSC	*requestLog()*	88 ms			
ACSC	*policyEnf()* *accessLog()*	96 ms			
KSSC	*requestData()* *keystoreLog()*	132 ms			
		527 ms	**2 ms**	**74 ms**	**603 ms**

[a] Cache delay is an adjustable parameter, depending on the size of the cache memory assigned to the inter-blockchain API.
[b] An ABE setting of one system and two user attributes was used.

Table 4 analyzes the overall execution time for the *access-to-data* service. The measurements were made for 100 concurrent requests and the mean time per transaction was computed, including all the system (blockchain) functions involved, the cache delay of the blockchain communication and the user-side component. For clarity, we decompose the blockchain execution time (527 *ms*) to all the functions used from each smart contract involved in the access service, including: initial request handling, user/credential validation, policy enforcement and system-side partial decryption. The cache delay time is configurable since it depends on the capacity of the cache memory

assigned to the inter-blockchain API. In our experiments where 1GB cache memory was used, the cache delay for 100 concurrent transactions was negligible (only 2 *ms*) showing that this configuration may serve a significantly higher number of bursts (concurrent transactions) before it becomes saturated. Finally, for the user-side execution time we used a virtual machine of a typical low-end mobile device with 4-core CPU and 8GB RAM to simulate the end user device and 74 *ms* were required for the user-side final decryption. In total 603 *ms* were required on average for each data access transaction. Although in real world scenarios other factors such as network communication delays should be considered, the results indicate that the system is efficient under the above setting, although further studies should be made for larger scale implementations.

VI. CONCLUSION

In this paper, we have presented an efficient system for managing granular access to health-related data, based on a novel hierarchical multi blockchain architecture with advanced functionalities. The architecture is supported by a set of fully functional smart contracts, security mechanisms used for data encryption, a fine-grained access control scheme and a strong forensics-by-design mechanism with highly robust audit trails and strong integrity guarantees.

To demonstrate the applicability of our architecture, we have designed a proof-of-concept implementation. Based on our experimental results, the proposed architecture is sufficiently versatile to accommodate multiple-use case scenarios within the health care ecosystem, for managing health-related data and enforcing multi-entity, fine-grained access control policies. The architecture provides various functional benefits in terms of secure interoperable data access, while it achieves transaction completion within a reasonable time frame. Another essential benefit of our architecture relates to its adaptive capacity and its ability to be flexible enough to adjust to the dynamic changes of the health care ecosystem (adding/removing stakeholders and entities).

From a security/privacy point of view, the proposed architecture presents various benefits, including the support for temporal access roles and for efficient revocation of the decryption functionality for the end users, instead of an ABE key revocation which usually implies a non-efficient data re-encryption process. Another important aspect of our architecture is its distributed trust management functionalities. Unlike cloud-based solutions in which a trusted authority centrally manages the trust, our architecture allows all stakeholders to manage the trust within their domain in a distributed and self-sustained fashion (manage the issuing, updating or revoking of access credentials). Finally, it incorporates a reliable forensics mechanism which is tailored to the accountability needs of all the stakeholders involved. This forensic mechanism can keep all actions and requests made to the system with strong integrity guarantees, thus improving regulatory compliance while eliminating opportunistic behavior of the parties involved.

Despite the significant benefits of our architecture, several of its aspects could be further improved. For example, instead of using the same consensus mechanisms (Proof-of-Stake) in our two blockchains, we could consider the possibility of using different consensus mechanisms for achieving a better balance between security and scalability. Since our experiments were conducted in a small scale proof-of-concept implementation, we could also consider larger scale implementation with extended use cases to further validate the scalability of the proposed architecture. For example, validating the overall system performance for medical data types of large size such as images or video could be implemented efficiently by first encrypting the data with symmetric encryption and then perform ABE encryption over the symmetric encryption keys. As future work, we plan to extend our proof-of-concept implementation, in order to further assess various implementation aspects related with intra-blockchain communication, consensus mechanism consolidation and Docker orchestration, and to achieve an optimal balance between the desired security and scalability.

REFERENCES

[1] E Commission. (2019). *Additional Guidance Regarding the Vigilance System as Outlined in Meddev 2.12-1 rev. 8, Directorate-General for Internal Market, Industry, Entrepreneurship and SMEs. Unit GROW D.4–Health Technology and Cosmetics*. Accessed: Feb. 6, 2019 from. [Online]. Available: https://ec.europa.eu/docsroom/documents/36292/attachments/1/translations/en/renditions/native

[2] B. L. Filkins, J. Y. Kim, B. Roberts, W. Armstrong, M. A. Miller, M. L. Hultner, A. P. Castillo, J.-C. Ducom, E. J. Topol, and S. R. Steinhubl, "Privacy and security in the era of digital health: What should translational researchers know and do about it?" *Amer. J. Transl. Res.*, vol. 8, no. 3, p. 1560, 2016.

[3] H. A. Al Hamid, S. M. M. Rahman, M. S. Hossain, A. Almogren, and A. Alamri, "A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography," *IEEE Access*, vol. 5, pp. 22313–22328, 2017.

[4] J.-J. Yang, J.-Q. Li, and Y. Niu, "A hybrid solution for privacy preserving medical data sharing in the cloud environment," *Future Gener. Comput. Syst.*, vols. 43–44, pp. 74–86, Feb. 2015.

[5] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, and A. Faxvaag, "Blockchain in healthcare and health sciences—A scoping review," *Int. J. Med. Informat.*, vol. 134, Feb. 2020, Art. no. 104040.

[6] M. Marwan, A. Kartit, and H. Ouahmane, "Using homomorphic encryption in cloud-based medical image processing: Opportunities and challenges," in *Innovations in Smart Cities and Applications* (Lecture Notes in Networks and Systems), vol. 37. New York, NY, USA: Springer, 2018, pp. 824–835.

[7] Y. Wu, X. Lu, J. Su, and P. Chen, "An efficient searchable encryption against keyword guessing attacks for sharable electronic medical records in cloud-based system," *J. Med. Syst.*, vol. 40, no. 12, p. 258, Dec. 2016.

[8] J. A. Akinyele, M. W. Pagano, M. D. Green, C. U. Lehmann, Z. N. J. Peterson, and A. D. Rubin, "Securing electronic medical records using attribute-based encryption on mobile devices," in *Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices (SPSM)*, 2011, pp. 75–86.

[9] L. Liu, J. Lai, R. H. Deng, and Y. Li, "Ciphertext-policy attribute-based encryption with partially hidden access structure and its application to privacy-preserving electronic medical record system in cloud environment," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 4897–4913, Dec. 2016.

[10] D. M. Eyers, J. Bacon, and K. Moody, "OASIS role-based access control for electronic health records," *IEE Proc. Softw.*, vol. 153, no. 1, pp. 16–23, Feb. 2006.

[11] B. Tiwari and A. Kumar, "Role-based access control through on-demand classification of electronic health record," *Int. J. Electron. Healthcare*, vol. 8, no. 1, pp. 9–24, 2015.

[12] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: Efficient policy-hiding attribute-based access control," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018.

[13] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation," *Int. J. Inf. Secur.*, vol. 14, no. 6, pp. 487–497, Nov. 2015.

[14] J. Bogaerts, M. Decat, B. Lagaisse, and W. Joosen, "Entity-based access control: Supporting more expressive access control policies," in *Proc. 31st Annu. Comput. Secur. Appl. Conf.*, vols. 7–11, 2015, pp. 291–300.

[15] Y. Miao, J. Ma, X. Liu, F. Wei, Z. Liu, and X. A. Wang, "m^2-ABKS: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting," *J. Med. Syst.*, vol. 40, no. 11, p. 246, Nov. 2016.

[16] H. Pussewalage and V. Oleshchuk, "A distributed multi-authority attribute based encryption scheme for secure sharing of personal health records," in *Proc. SACMAT*, Jun. 2017, pp. 255–262.

[17] S. Amofa, E. B. Sifah, K. O.-B. Obour Agyekum, S. Abla, Q. Xia, J. C. Gee, and J. Gao, "A blockchain-based architecture framework for secure sharing of personal health data," in *Proc. IEEE 20th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Sep. 2018, pp. 1–6.

[18] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology," *Sustain. Cities Soc.*, vol. 39, pp. 283–297, May 2018.

[19] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "MedBlock: Efficient and secure medical data sharing via blockchain," *J. Med. Syst.*, vol. 42, no. 8, p. 136, Aug. 2018.

[20] H. Li, L. Zhu, M. Shen, F. Gao, X. Tao, and S. Liu, "Blockchain-based data preservation system for medical data," *J. Med. Syst.*, vol. 42, no. 8, p. 141, Aug. 2018.

[21] H. Wang and Y. Song, "Secure cloud-based EHR system using attribute-based cryptosystem and blockchain," *J. Med. Syst.*, vol. 42, no. 8, p. 152, Jul. 2018.

[22] A. E. Gutiérrez Díaz, J. Armas, J. M. Madrid Molina, and C. A. Natividad Peña, "Security model to protect patient data in mHealth systems through a blockchain network," in *Proc. 17th LACCEI Int. Multi-Conf. Eng., Edu., Technol., 'Ind., Innov., Infrastruct. Sustain. Cities Communities'*, 2019, pp. 1–6.

[23] V. Ramani, T. Kumar, A. Bracken, M. Liyanage, and M. Ylianttila, "Secure and efficient data accessibility in blockchain based healthcare systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 206–212.

[24] B. L. Radhakrishnan, A. S. Joseph, and S. Sudhakar, "Securing blockchain based electronic health record using multilevel authentication," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Mar. 2019, pp. 699–703.

[25] B. Shen, J. Guo, and Y. Yang, "MedChain: Efficient healthcare data sharing via blockchain," *Appl. Sci.*, vol. 9, no. 6, p. 1207, Mar. 2019.

[26] T. T. Thwin and S. Vasupongayya, "Blockchain-based access control model to preserve privacy for personal health record systems," *Secur. Commun. Netw.*, vol. 2019, pp. 1–15, Jun. 2019.

[27] Y. Zhao, M. Cui, L. Zheng, R. Zhang, L. Meng, D. Gao, and Y. Zhang, "Research on electronic medical record access control based on blockchain," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 11, Nov. 2019, Art. no. 155014771988933.

[28] V. Malamas, T. Dasaklis, P. Kotzanikolaou, M. Burmester, and S. Katsikas, "A forensics-by-design management framework for medical devices based on blockchain," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2019, pp. 35–40.

[29] U. Morelli, S. Ranise, D. Sartori, G. Sciarretta, and A. Tomasi, "Audit-based access control with a distributed ledger: Applications to healthcare organizations," in *Proc. Int. Workshop Secur. Trust Manage.*, 2019, pp. 19–35.

[30] L. Hirtan, P. Krawiec, C. Dobre, and J. M. Batalla, "Blockchain-based approach for e-Health data access management with privacy protection," in *Proc. IEEE 24th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw. (CAMAD)*, Sep. 2019, pp. 1–7.

[31] A. Zhang and X. Lin, "Towards secure and privacy-preserving data sharing in e-Health systems via consortium blockchain," *J. Med. Syst.*, vol. 42, no. 8, p. 140, Aug. 2018.

[32] L. Chen, W.-K. Lee, C.-C. Chang, K.-K.-R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Gener. Comput. Syst.*, vol. 95, pp. 420–429, Jun. 2019.

[33] X. Yang, T. Li, X. Pei, L. Wen, and C. Wang, "Medical data sharing scheme based on attribute cryptosystem and blockchain technology," *IEEE Access*, vol. 8, pp. 45468–45476, 2020.

[34] A. Al Omar, M. S. Rahman, A. Basu, and S. Kiyomoto, "MediBchain: A blockchain based privacy preserving platform for healthcare data," in *Proc. Int. Conf. Secur., Privacy Anonymity Comput., Commun. Storage*, 2017, pp. 534–543.

[35] S. Niu, L. Chen, J. Wang, and F. Yu, "Electronic health record sharing scheme with searchable attribute-based encryption on blockchain," *IEEE Access*, vol. 8, pp. 7195–7204, 2020.

[36] X. Zhang and S. Poslad, "Blockchain support for flexible queries with granular access control to electronic medical records (EMR)," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[37] F. Tang, S. Ma, Y. Xiang, and C. Lin, "An efficient authentication scheme for blockchain-based electronic health records," *IEEE Access*, vol. 7, pp. 41678–41689, 2019.

[38] G. Li and H. Sato, "A privacy-preserving and fully decentralized storage and sharing system on blockchain," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2019, pp. 694–699.

[39] Q. Li, H. Zhu, J. Xiong, R. Mo, Z. Ying, and H. Wang, "Fine-grained multi-authority access control in IoT-enabled mHealth," *Ann. Telecommun.*, vol. 74, nos. 7–8, pp. 389–400, Aug. 2019.

[40] H. S. Gardiyawasam Pussewalage and V. A. Oleshchuk, "Blockchain based delegatable access control scheme for a collaborative E-health environment," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1204–1211.

[41] Y. Sun, R. Zhang, X. Wang, K. Gao, and L. Liu, "A decentralizing attribute-based signature for healthcare blockchain," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–9.

[42] S. M. Pournaghi, M. Bayat, and Y. Farjami, "MedSBA: A novel and secure scheme to share medical data based on blockchain technology and attribute-based encryption," *J. Ambient Intell. Humanized Comput.*, Jan. 2020, pp. 1–29.

[43] J. P. Dias, H. S. Ferreira, and A. Martins, "A blockchain-based scheme for access control in e-health scenarios," in *Proc. Int. Conf. Soft Comput. Pattern Recognit.*, 2020, pp. 238–247.

[44] L. Deng, H. Chen, J. Zeng, and L. J. Zhang, "Research on cross-chain technology based on sidechain and hash-locking," in *Proc. Int. Conf. Edge Comput.*, 2018, pp. 144–151.

[45] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K.-R. Choo, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review," *J. Netw. Comput. Appl.*, vol. 149, Jan. 2020, Art. no. 102471.

[46] K. Wang, Z. Zhang, and H. S. Kim, "ReviewChain: Smart contract based review system with multi-blockchain gateway," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1521–1526.

[47] *Systems and Software Engineering—Architecture Description*, Standard ISO/IEC/IEEE 42010:2011, 2017. [Online]. Available: https://www.iso.org/standard/50508.html

[48] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Informat.*, vol. 36, pp. 55–81, Mar. 2019.

[49] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50759–50779, 2019.

[50] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, 2006, pp. 89–98.

[51] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[52] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* New York, NY, USA: Springer, 2011, pp. 568–588.

[53] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Proc. Int. Conf. Financial Cryptography Data Secur.* New York, NY, USA: Springer, 2015, pp. 315–332.

[54] M. Cheng, B. Moher, E. Napke, L. Lehtiniemi, J. Erskine, and T. Gaamangwe, "Medical device regulations and patient safety," in *Clinical Engineering Handbook*. Amsterdam, The Netherlands: Academic, 2019, pp. 353–356.
[55] J. Willis, "Docker and the three ways of DevOps," Docker, Inc., Palo Alto, CA, USA, White Paper, 2015. [Online]. Available: https://goto.docker.com/rs/929-FJL-178/images/20150731-wp_docker-3-ways-devops.pdf
[56] S. Edward, D. Czarnota, R. Tonic, and B. Perez, "Kubernetes security whitepaper," Trail Bits, New York, NY, USA, White Paper, 2019. [Online]. Available: https://github.com/kubernetes/community/blob/master/wg-security-audit/findings/Kubernetes%20White%20Paper.pdf
[57] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, Jun. 2013, doi: 10.1007/s13389-013-0057-3.

VANGELIS MALAMAS (Graduate Student Member, IEEE) received the degree in mathematics from the University of Patras, in 2008, and the M.Sc. degree in computer science from the University of Piraeus, in 2017, where he is currently pursuing the Ph.D. degree in distributed security and trust management technologies on the IoT with the Department of Informatics. He is a member of the Security Research Laboratory (SecLab). His research interests include blockchain, the IoT security, and cryptography.

PANAYIOTIS KOTZANIKOLAOU (Member, IEEE) received the degree in computer science and the Ph.D. degree in ICT security from the University of Piraeus, Greece, in 1998 and 2003, respectively. He is currently an Associate Professor of network security and privacy with the Department of Informatics, University of Piraeus, and the Director of the Security Research Laboratory (SecLab). He has participated in various national, European, and international research and development projects. He has published more than 70 articles in books, peer-reviewed journals, and conferences. His research interests include network security, communication privacy, applied cryptography, and critical infrastructure protection. He has served as a guest editor, a program committee member, and a reviewer for various international journals and conferences.

THOMAS K. DASAKLIS graduated from the Department of Industrial Management and Technology, University of Piraeus. He received the M.Sc. degree in supply chain management and the Ph.D. degree in emergency supply chain management and disaster response. His research interests include supply chain management, operational research, humanitarian logistics/disaster response, data analysis, and blockchain technology. He has participated in National and European Research projects and has published papers in several international journals and conference proceedings. He has worked for the European Commission (DG Humanitarian Aid and Civil Protection) and the University of Piraeus Research Centre. He has also worked in the private sector for three years as a Supply Chain Director. He is currently an Adjunct Academic Staff with Hellenic Open University. He is also a Seasonal Lecturer with the Department of Informatics, University of Piraeus.

MIKE BURMESTER (Senior Member, IEEE) is currently a Professor of computer science with Florida State University, where he has been a Faculty Member, since January 2000. He is the Director of the FSU Center for Security and Assurance in IT, an editor of four journals, and has published over 250 publications. His research was supported by NSF and NSA/DoD. His research interests include cyber security and information assurance, with particular focus on cyber physical systems, the IoT, network security, and cryptography.